

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

**Evolução, simbiose e hibridismo aplicados à
engenharia de sistemas inteligentes modulares:
Investigações em redes neurais, comitês de máquinas e
sistemas multiagentes**

Tese de Doutorado

Autor: **André Luís Vasconcelos Coelho**

Orientador: **Ivan Luiz Marques Ricarte, PhD**

Banca Examinadora: **Prof. Dr. Ivan Luiz Marques Ricarte (DCA-FEEC-Unicamp)**

Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho (ICMC-USP)

**Prof. Dr. Carlos Henrique Costa Ribeiro (Depto. de Teoria da
Computação-ITA)**

Prof. Dr. Márcio Luiz de Andrade Netto (DCA-FEEC-Unicamp)

Prof. Dr. Akebo Yamakami (DT-FEEC-Unicamp)

Prof. Dr. Fernando José Von Zuben (DCA-FEEC-Unicamp)

Tese submetida à Faculdade de Engenharia Elétrica e de
Computação da Universidade Estadual de Campinas, como
parte dos requisitos para a obtenção do Título de Doutor
em Engenharia Elétrica.

Área de concentração: Engenharia de Computação.

30 de julho de 2004

©

Este exemplar corresponde a redação final da tese
defendida por André Luís Vasconcelos Coelho
e aprovada pela Comissão
Julgada em 30/07/04

UNICAMP
BIBLIOTECA CENTRAL

NIDADE	PC
CHAMADA	
T/UNICAMP	
C65e	
EX	
OMBO, BC/	61122
ROC.	16-11-04
C ☐	D ☒
REÇO	14.00
DATA	14-11-04
CPD	

3181d 321571

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

C65e

Coelho, André Luís Vasconcelos

Evolução, simbiose e hidridismo aplicados à engenharia de sistemas inteligentes modulares: investigações em redes neurais, comitês de máquinas e sistemas multiagentes / André Luís Vasconcelos Coelho.--Campinas, SP: [s.n.], 2004.

Orientador: Ivan Luiz Marques Ricarte.

Tese (Doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Aprendizado do computador. 2. Agentes inteligentes (Software). 3. Inteligência artificial distribuída. 4. Algoritmos genéticos. 5. Redes neurais (Computação). 6. Simbiose. 7. Coevolução. 8. Hierarquias. 9. Cooperação. 10. Kernel, Função de. I. Ricarte, Ivan Luiz Marques. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

COELHO, A. L. V. – **Evolução, simbiose e hibridismo aplicados à engenharia de sistemas inteligentes modulares: Investigações em redes neurais, comitês de máquinas e sistemas multiagentes.** Campinas: FEEC/Unicamp, Maio 2004. Tese de Doutorado — Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, 331 pp.

Este tratado é de caráter essencialmente empírico e apresenta um estudo sistemático e abrangente em torno da aplicação de técnicas de Computação Evolutiva para a resolução adaptativa de problemas não-triviais relacionados à disciplina de engenharia de sistemas inteligentes modulares. Os vários experimentos conduzidos e reportados tiveram como motivação maior a avaliação das potencialidades exibidas por algoritmos evolutivos mais rebuscados, inspirados em modelos artificiais de simbiose e hibridismo, tendo como referencial um leque de questões/requisitos pertinentes a sistemas modulares levantados pelo autor. Os domínios-alvo de pesquisa investigados compreendem o desenho customizado de arquiteturas de redes neurais heterogêneas; o projeto automático e hierárquico de *ensembles* heterogêneas de redes neurais *feed-forward*; a seleção adaptativa de instâncias de máquinas de vetores-suporte para a composição de *ensembles* de SVMs; a configuração plena de arquiteturas de misturas de especialistas; e a organização emergente de times de agentes autônomos. Outros projetos correlatos ao tema são também brevemente descritos.

Palavras-chave: sistemas modulares, sistemas inteligentes, aprendizado de máquina, inteligência computacional, sistemas multiagentes, computação evolutiva, coevolução artificial, sistemas connexionistas, máquinas de vetores-suporte, comitês de máquinas, *ensembles*, misturas de especialistas, simbiose, hierarquia, hibridismo, emergência, organização, classificação de padrões, predição de séries temporais e futebol de robôs simulado.

Abstract

COELHO, A. L. V. – **Evolution, symbiosis and hybridism applied to modular intelligent systems engineering: Investigations on neural networks, committee machines, and multiagent systems.** Campinas: FEEC/Unicamp, Maio 2004. Dr. E. E. Thesis — Faculty of Electrical and Computer Engineering, State University of Campinas, pp. 331

This treaty is of an essentially empirical nature and presents a systematic and encompassing study about the application of Evolutionary Computation techniques towards the adaptive solving of non-trivial problems under the discipline of modular intelligent systems engineering. The series of experiments conducted and reported along the text had as main stimulus the assessment of the potentials behind some advanced evolutionary algorithms, inspired from artificial models of symbiosis and hybridism, with regard to an array of issues/requirements pertaining to modular systems raised by the author. The target domains of investigation include the tailored design of heterogeneous neural network architectures; the automatic and hierarchical development of heterogeneous ensembles of feedforward neural networks; the adaptive selection of support vector machine instances for setting up ensembles of SVMs; the entire configuration of mixture of experts architectures; and the emergent organization of teams made up of autonomous agents. Other correlated projects under the same research umbrella are also succinctly touched upon.

Keywords: modular systems, intelligent systems, machine learning, computational intelligence, multiagent systems, evolutionary computation, artificial coevolution, connectionist systems, support vector machines, committee machines, ensembles, mixtures of experts, symbiosis, hierarchy, hybridism, emergence, organization, pattern classification, time series forecasting, and simulated robot soccer.

Agradecimentos

Permito-me, neste espaço, a expressar a minha gratidão àqueles que, de alguma forma, contribuíram para a conclusão desta obra:

Aos membros do clã Coelho, pelo estímulo e abdicação constantes; por terem me indicado o caminho e me deixado seguir com as minhas próprias pernas, mas sempre me amparando nos momentos de dificuldade; e por me ensinarem, não com fórmulas matemáticas ou algoritmos computacionais, que tudo vale mais a pena quando feito com as nossas próprias mãos. A meu pai, Dr. Pojuca, minha mãe, D. Verinha, minhas irmãs, Ana Paula (Jaana e Johan) e Aline, e meus amigos, Jaap e Nuno, o meu eterno agradecimento.

À minha Bibinha, simplesmente por tudo. À sua família, pela amizade, suporte e momentos de descontração.

Ao Prof. Ivan Ricarte, pela liberdade a mim concedida quanto à escolha do tema, pelo apoio logístico, pela paciência e confiança devotadas, e pela disposição em ouvir e estimular minhas elocubrações.

Aos Profs. Fernando Von Zuben, Fernando Gomide e Márcio Netto, por terem compartilhado material bibliográfico de extrema importância; por terem me introduzido à tarefa desafiadora de julgamento de trabalhos técnicos de outros pares; pelas aulas ministradas; pela orientação quanto ao rumo a seguir no pós-doutorado; e pelo exemplo na conduta da pesquisa científica.

Ao Prof. Helmut Mayer, pelas conversas remotas, sempre muito proveitosas, em torno do tema da tese.

Aos membros da banca, pelas críticas e sugestões construtivas sobre o teor do trabalho.

Aos vários (ex-)colegas da FEEC, pelo ambiente, pelas discussões, pelo compartilhar dos sonhos de quem se inicia na vida acadêmica. Em especial, aos amigos Luiz Gonzaga Jr., Rodrigo Prado, Alencar Jr., Luís Fernando Faina, Alberto Raposo, Carlos Forster, Edgar Noda, Cláudio Muniz, Mário Ernesto Silva, Daniel Ojeda, Igor Walter, Daniel Weingaertner, Franklin Flores, Alessandro Bicho, Victor Tatai e Antônio Sérgio Gomes. A Clodoaldo Lima agradeço enormemente pelo convívio nas horas de pesquisa em grupo, pelo compartilhar das idéias, pela co-autoria dos vários trabalhos e por ter-me introduzido ao tema de SVMs e Misturas de Especialistas.

Aos entes Rodrigo Gribel e Maurício de Lima Jr., pelas discussões filosóficas sobre temas os mais variados, pelo estímulo ao término do trabalho e, sobretudo, pela amizade.

A APOGEEU, por ter-me permitido exercer o difícil mas estimulante papel de representante discente.

Ao CNPq, pelo aporte financeiro.

A Unicamp, pelo acolhimento deste filho de Deus ao longo dos últimos 12 anos.

À minha Biba Maria Nenê Jujubolinha Peloponesa Profiterólis Zugüe-zugüe Silva,
Com todo o carinho que só você pode me dar.

Sumário

SUMÁRIO	v
LISTA DE FIGURAS	ix
LISTA DE TABELAS	xii
GLOSSÁRIO DE ACRÔNIMOS	xv
1 Introdução	1
1.1 Modularidade aplicada a sistemas inteligentes	3
1.2 Objeto de estudo	7
1.2.1 Treinamento coevolucionário de redes neurais heterogêneas	9
1.2.2 Projeto automático de comitês de máquinas	10
1.2.3 Organização emergente de times de agentes autônomos	11
1.3 Organização do documento	11
2 Redes neurais artificiais com múltiplas camadas	13
2.1 Aplicações em estudo	14
2.2 Propriedades gerais	15
2.3 Aspectos estruturais em redes <i>feedforward</i>	18
2.3.1 O problema do dimensionamento de RNAs	20
2.3.2 Seleção de atributos	22
2.3.3 Capacidade de uma rede MLP e configuração de seus parâmetros	23
2.4 Treinamento de redes neurais com camadas múltiplas	26
2.4.1 Tratamento e particionamento dos dados	26

2.4.1.1	Pré-processamento dos dados	27
2.4.1.2	Particionamento dos dados	28
2.4.2	Metodologia de retropropagação do erro	29
2.4.3	Modos de aprendizado: incremental ou em batelada	33
2.4.4	Retropropagação do erro: Dificuldades e deficiências	34
2.5	Aprendizado <i>versus</i> generalização	34
2.6	Considerações finais	37
3	Máquinas de vetores-suporte	39
3.1	Formalização de máquinas de vetores-suporte	40
3.2	Pontos adicionais	47
3.3	Observações finais	51
4	Comitês de máquinas	53
4.1	<i>Ensembles</i> de estimadores	57
4.1.1	Justificativas teóricas	60
4.1.2	Diversidade em <i>ensembles</i>	66
4.1.3	Criação de componentes	70
4.1.4	Seleção de componentes	77
4.1.5	Combinação de componentes	79
4.2	Misturas de especialistas	83
4.2.1	Interpretação probabilística de MEs	86
4.2.2	Treinamento de misturas de especialistas via EM	88
4.2.2.1	Passo-E	89
4.2.2.2	Passo-M	90
4.2.3	Pontos adicionais	91
4.3	Sinopse do capítulo	95
5	Coordenação em sistemas com múltiplos agentes	97
5.1	Inteligência artificial distribuída — Aspectos importantes	100
5.2	Coordenação em sistemas multiagentes	102

5.2.1	Caracterização do problema de coordenação	103
5.2.2	Políticas, protocolos e mecanismos de coordenação	105
5.2.2.1	Planejamento	107
5.2.2.2	Reatividade	107
5.2.2.3	Negociação	107
5.2.2.4	Mecanismos de mercado	108
5.2.2.5	Organizações	108
5.2.3	Análise de algumas linhas de atuação	109
5.3	Síntese do capítulo	113
6	Computação evolutiva e sua aplicação em sistemas modulares	115
6.1	O que caracteriza um algoritmo evolutivo?	116
6.2	Pontos críticos	122
6.3	Aspectos teóricos em AGs	123
6.4	Componentes de algoritmos evolutivos	124
6.5	Tópicos avançados	128
6.6	Algoritmos evolutivos híbridos	131
6.7	Simbiose e coevolução: Do biológico ao artificial	136
6.7.1	Coevolução artificial competitiva	140
6.7.2	Coevolução artificial cooperativa	142
6.8	Computação evolutiva e sistemas emergentes	149
6.9	Aplicações de algoritmos evolutivos em sistemas inteligentes modulares	150
6.10	Últimas ponderações	163
7	Uma arquitetura coevolucionária, hierárquica e híbrida para <i>ensembles</i> heterogêneas de redes neurais artificiais	165
7.1	Motivação e escopo de investigação	166
7.2	COHENN: Propriedades e arquitetura	168
7.2.1	Funcionalidades	169
7.2.2	Passos do processo coevolucionário e caracterização das populações	174
7.3	Resultados experimentais	182

7.3.1	Análise de eficiência e freqüencial	184
7.3.2	Análise de eficácia	193
7.4	Recapitulação e outras perspectivas	201
8	Outros projetos em engenharia de sistemas inteligentes modulares	203
8.1	Redes neurais coevolutivas	203
8.1.1	Considerações adicionais	216
8.2	Seleção automática de modelos de SVM	218
8.2.1	Considerações adicionais	231
8.3	Misturas de especialistas evolutivas	231
8.3.1	Considerações adicionais	243
8.4	Organização emergente de times de agentes artificiais	249
8.4.1	Comentários adicionais	266
8.5	Trabalhos correlatos	266
8.6	Encerramento	269
9	Conclusão	271
9.1	Análise das contribuições	271
9.2	Perspectivas futuras	275
A	Aprendizado de máquina: Terminologia básica, tratamento dos dados e análise de desempenho para modelos supervisionados	279
A.1	Terminologia básica	280
A.2	Tratamento dos dados	284
A.3	Comparação entre algoritmos	286
B	Métodos de combinação em sistemas com múltiplos classificadores	289
B.1	Definições básicas da teoria da evidência de Dempster-Shafer	299
C	Descrição dos problemas de classificação	301
	Referências Bibliográficas	304

Lista de figuras

2.1	Modelo do neurônio MCP e arquitetura de uma rede MLP.	19
2.2	Série <i>forex</i> original, transformada e normalizada.	24
2.3	Passos da metodologia de retropropagação do erro	29
3.1	Modo de operação de um classificador com margem larga	41
3.2	Superfícies de decisão geradas por <i>kernels</i> distintos para um problema fictício	46
3.3	Arquitetura típica de uma instância de SVM.	48
4.1	Um exemplo de arquitetura multimodular híbrida.	57
4.2	Diagrama em blocos de um modelo típico de <i>ensemble</i>	58
4.3	Exemplo de operação de <i>Bagging</i> e <i>Boosting</i> sobre um conjunto de dados hipotético.	73
4.4	Esquemático do método AdaBoost com amostras ponderadas.	75
4.5	Esquemático do método de empilhamento com validação cruzada.	82
4.6	Arquitetura ME.	83
4.7	Rede de crenças associada a MEs.	87
5.1	Uma taxonomia de coordenação em SMAs.	104
5.2	Alguns padrões organizacionais	109
6.1	Linhas de atuação em Computação Evolutiva.	119
6.2	Pontos de hibridização em um AG simples.	133
6.3	Efeito da busca local (aprendizado) sobre uma superfície de <i>fitness</i>	135

6.4	Arquitetura coevolucionária cooperativa proposta por Potter e De Jong.	147
6.5	Um arcabouço geral para caracterização de RNAs evolutivas.	152
6.6	Neuroevolução “canônica” e neuroevolução realizada por SANE.	156
7.1	Escopo de investigação de COHENN.	167
7.2	Modelo arquitetural hierárquico COHENN.	169
7.3	<i>Ensembles</i> de <i>ensembles</i> evolucionárias.	173
7.4	Processo coevolucionário por trás de COHENN.	175
7.5	Cromossomos dos três tipos de espécies.	176
7.6	Processo de convergência de COHENN para as nove primeiras instâncias de problemas.	186
7.7	Processo de convergência de COHENN para as nove últimas instâncias de problemas.	187
8.1	Arquitetura da rede neural heterogênea.	207
8.2	Processo de evolução hierárquica de redes neurais	208
8.3	Resultados mostrando a evolução do <i>fitness</i> da melhor rede e do <i>fitness</i> médio da população.	214
8.4	Arquétipo da abordagem HCGA_2.	217
8.5	Algoritmo para experimentos relacionados aos métodos PCSe1, VCSe1, and MMSe1.	225
8.6	Propriedades estruturais dos espaços de busca.	225
8.7	Algoritmo para experimentos relacionados ao método GASE1.	226
8.8	Uma execução típica de GASE1 para o problema BCD699.	227
8.9	Variação da taxa de erro para uma execução relativa ao problema BCD699.	228
8.10	Variação da taxa de erro para uma execução relativa ao problema HEP.	228
8.11	Arquitetura ME.	233
8.12	Pseudo-código para HGT-GAME.	237
8.13	Cromossomo em HGT-GAME.	237
8.14	Predição da série Mackey-Glass (HGT-GAME).	240
8.15	Erro de predição para série <i>forex</i> (HGT-GAME).	240
8.16	Crescimento e poda dinâmicos—Número médio de especialistas nas 10% melhores MEs (Forex).	241

8.17 Janela de predição para a série Laser (HGT-GAME).	241
8.18 Comparações entre melhores instâncias HGT-GAME treinadas com aprendizado híbrido lamarckiano e baldwiniano sobre a série Laser.	247
8.19 Comparações entre as saídas das redes <i>gating</i> para as melhores instâncias HGT-GAME treinadas com aprendizado híbrido lamarckiano e baldwiniano sobre a série Laser.	248
8.20 Posição inicial do jogo para o simulador de futebol de robôs.	252
8.21 Abordagem de memória compartilhada para coordenação multiagente.	254
8.22 Área de atuação de um jogador e coordenadas de discretização do campo.	255
8.23 Processo coevolutivo de criação emergente de times.	257
8.24 Modelo de objetos com relações entre classes Java.	258
8.25 Fluxo de execução da abordagem coevolucionária proposta.	258
8.26 Formações dos times.	261

Lista de tabelas

1.1	Diferenças entre paradigmas computacionais.	5
2.1	Descrição de funções (não-)lineares (e suas derivadas) tipicamente adotadas em redes MLP	21
2.2	Métricas de desempenho em problemas de previsão de séries temporais	31
3.1	Lista de possíveis <i>kernels</i> e seus parâmetros associados.	45
4.1	Algumas métricas para quantificar o grau de diversidade em uma <i>ensemble</i> de classificadores qualquer.	68
5.1	Algumas propriedades de agentes de <i>software</i>	99
6.1	Pontos fortes e fracos dos algoritmos evolutivos tradicionais.	129
6.2	Classificação das interações simbióticas de espécies biológicas.	137
6.3	Algumas definições para o termo “coevolução”.	139
6.4	Diferentes trabalhos no segmento de Coevolução Cooperativa.	142
7.1	Parâmetros de configuração comum a todas as populações.	182
7.2	Parâmetros de configuração da população de descrições de <i>ensembles</i>	182
7.3	Parâmetros de configuração da população de redes-componentes.	183
7.4	Parâmetros de configuração da população de redes-combinadoras.	183
7.5	Valores adotados para os parâmetros dos operadores fixos.	184
7.6	Tempo médio necessário para uma execução de COHENN, número médio de gerações necessárias para encontrar a estrutura de <i>ensemble</i> vencedora em potencial e número médio de componentes não-nulos das <i>ensemble</i> vencedoras finais do processo.	185
7.7	Taxa média de acerto exibida pelo oráculo.	188
7.8	Número médio de pesos sinápticos e número médio de neurônios escondidos pertencentes aos arranjos COHENN1.	190

7.9	Análise freqüencial (%) das funções de ativação compondo os arranjos COHENN1.	192
7.10	Operador de combinação fixo mais freqüentemente adotado por COHENN para compor as <i>ensembles</i> vencedoras finais relativas a cada instância de problema.	193
7.11	Operadores fixos mais utilizados por COHENN para a geração de suas <i>ensembles</i> vencedoras finais.	193
7.12	Taxas médias de acerto dos modelos gerados por COHENN sobre os conjuntos de treinamento e teste.	195
7.13	Taxas médias de acerto dos modelos de <i>ensembles</i> de <i>ensembles</i> evolucionárias gerados por COHENN sobre os conjuntos de treinamento e teste.	196
7.14	Taxas médias de acerto dos melhores modelos de redes-componentes gerados por COHENN sobre os conjuntos de treinamento e teste.	197
7.15	Comparação dos modelos produzidos por COHENN com aqueles oriundos das técnicas de <i>Boosting</i> , <i>Bagging</i> e <i>Stacking</i>	198
7.16	Comparação das taxas médias de erro exibidas por COHENN1 com aquelas exibidas por outras abordagens de AM para a instância de problema <i>Credit</i>	200
7.17	Comparação das taxas médias de erro exibidas por COHENN1 com aquelas exibidas por outras abordagens de AM para a instância de problema <i>Diabetes</i>	200
7.18	Comparação das taxas médias de erro exibidas por COHENN1 com aquelas exibidas por outras abordagens de AM para a instância de problema <i>Heart</i>	200
8.1	Conjunto de funções de ativação.	209
8.2	Parâmetros de configuração.	211
8.3	Número de neurônios de cada tipo para a melhor rede gerada.	212
8.4	Resultados comparativos entre HCGA e outros métodos construtivos.	213
8.5	Resultados para diferentes métodos de seleção sobre o problema BCD699 (dados de teste).	229
8.6	Resultados para diferentes métodos de seleção sobre o problema HEP (dados de teste).	229
8.7	Resultados para diferentes métodos de seleção sobre o problema MCR (dados de teste).	230
8.8	Configurações de parâmetros para os experimentos conduzidos com MEs engatilhadas.	240
8.9	Resultados médios para as séries estudadas.	242
8.10	Novos resultados para as séries estudadas.	246
8.11	Coordenadas das regiões do campo.	255
8.12	Parâmetros de configuração	260
8.13	Ordem de criação dos novos AGs e suas regiões associadas.	262

9.1	Análise de como os domínios de investigação da tese contemplaram os pontos levantados no capítulo introdutório.	272
A.1	Matriz de confusão de um classificador.	283
C.1	Conjunto de problemas de classificação.	301
C.2	Descrição dos problemas da Tab. C.1.	302

Glossário de acrônimos

ACE	<i>Action Estimation</i>
ADDEMUP	<i>Accurate and Diverse Ensemble-Maker giving United Predictions</i>
AE	Algoritmo Evolutivo
AEH	Algoritmo Evolutivo Híbrido
AG	Algoritmo Genético
AGE	<i>Action Group Estimation</i>
AM	Algoritmo Memético
AM	Aprendizado de Máquina
ANFIS	<i>Adaptive Network-based Fuzzy Inference Systems</i>
Arcing	<i>Adaptive Resampling and Combining</i>
Bagging	<i>Bootstrap Aggregation and Combining</i>
BKS	<i>Behavior Knowledge Space</i>
CE	Computação Evolutiva
CLO-EQM	Combinação Linear Ótima–Erro Quadrático Médio
CMNN	<i>Cooperative Modular Neural Networks</i>
CNET	<i>Contract Net</i>
COHENN	<i>Coevolutionary Heterogeneous Ensembles of Neural Networks</i>
CS	<i>Clustering & Selection</i>
DBNN	<i>Decision-Based Neural Networks</i>
DCS_LA	<i>Dynamic Classifier Selection based on Local Accuracy</i>

Dimensão VC	Dimensão “Vapnik-Chervonenkis”
DPI	Dilema-do-prisioneiro iterativo
DPV	Decomposição Polarização-Variância
DS	Dempster-Shafer
DT	<i>Decision Template</i>
E-ANFIS	<i>Ensembles of ANFIS</i>
E/S	Entrada/Saída
EB	Efeito Baldwin
EEs	Estratégias Evolucionárias
ELMI	<i>Evolutionary Learning with Mutual Information</i>
EM	<i>Expectation Maximization</i>
ENNCL	<i>Evolutionary Ensembles with Negative Correlation Learning</i>
EQM	Erro Quadrático Médio
ERBF	Função de base radial exponencial
ERC	<i>Evolutionary Resample and Combine</i>
ERNM	<i>Ensembles de Redes Neurais Multiestágios</i>
EVOL	<i>Ensembles Voting On-Line</i>
FFCs	Funções de <i>Fitness</i> Competitivas
GARCH	<i>Generalized Autoregressive Conditional Heteroskedasticity</i>
GASel	<i>GA-based Selection of Ensemble Components</i>
GASEN	<i>Genetic Algorithm based Selective Ensemble</i>
GEM	<i>Generalized EM</i>
HCGA	<i>Hierarchical Coevolutionary Genetic Algorithm</i>
HE-SVMs	<i>Heterogeneous Ensembles of SVMs</i>
HGT-GAME	<i>Hybrid Genetic Training of Gated MEs</i>
HTML	<i>Hypertext Markup Language</i>

IA	Inteligência Artificial
IAD	Inteligência Artificial Distribuída
IC	Inteligência Computacional
IRPS	Índice Relativo de Pesos Sinápticos
IRTE	Índice Relativo de Tempo de Execução
KKT	Karush-Kuhn-Tucker
KQML	<i>Knowledge Query and Manipulation Language</i>
L-MSVME	<i>Localized Mixture of Support Vector Machine Experts</i>
LDA	<i>Linear Discriminant Analysis</i>
LME	<i>Localized Mixture of Experts</i>
LS-SVM	<i>Least-Squares SVM</i>
MASCoNN	<i>Multiagent Systems-based Constructive Neural Network</i>
MAV	<i>Moving Average Variance</i>
MCP	Neurônio McCulloch-Pitts
MDL	<i>Minimum Description Length</i>
ME	Misturas de Especialistas
MEB	Método de <i>Ensemble</i> Básica
MEG	Método de <i>Ensemble</i> Generalizada
MLP	Rede Perceptron Multicamadas
MOCCGA	<i>Multi-Objective Cooperative Coevolutionary Genetic Algorithm</i>
MOGA	<i>Multi-Objective Genetic Algorithm</i>
NDPI	Dilema-do-prisioneiro iterativo com N jogadores
NFL	<i>No-Free-Lunch</i>
NPGA	<i>Niched-Pareto Genetic Algorithm</i>
NSGA	<i>Nondominated Sorting Genetic Algorithm</i>
OWA	<i>Ordered Weighted Averaging</i>

PAC	<i>Probably Approximately Correct</i>
PAES	<i>Pareto Archived Evolution Strategy</i>
PCA	Análise de componentes principais
PE	Programação Evolucionária
PG	Programação Genética
PPQ	Problema de Programação Quadrática
RBF	Função de base radial
RBFN	Rede Neural de Base Radial
RDP	Resolução Distribuída de Problemas
RFNC	Redes <i>Fuzzy</i> -Neurais Cooperativas
RNA	Rede Neural Artificial
SAC	Sistema Adaptativo Complexo
SANE	<i>Symbiotic Artificial Neuroevolution</i>
SCs	Sistemas Classificadores
SEQ	Soma dos Erros Quadráticos
SIH	Sistema Inteligente Híbrido
SMA	Sistema Multiagente
SMO	<i>Sequential Minimal Optimization</i>
SSGE	<i>Scale-Sensitive Gated Experts</i>
SUS	<i>Stochastic Universal Sampling</i>
SV	<i>Support Vector</i>
SVM	<i>Support Vector Machine</i>
XCS	<i>Special Classifier Systems</i>
ZCS	<i>Zeroth-level Classifier Systems</i>

Capítulo 1

Introdução

“O começo é a parte mais importante de uma obra.”
Platão

Modularidade⁽¹⁾ parece ser um conceito ubíquo na natureza. Em diferentes níveis de resolução, sejam estes temporais ou espaciais, é possível encontrar sistemas (de origem biótica ou abiótica) exibindo a propriedade de serem modulares, na medida em que são passíveis de subdivisão em partes identificáveis (e, muitas vezes, interdependentes), cada uma com sua própria estrutura e função.

Com efeito, todas as possíveis classes de organismos vivos podem ser decompostas em átomos, moléculas, sólidos e fluidos. De um outro ponto de vista, de caráter mais fisiológico, pode-se constatar que células aglutinam-se em tecidos, que, por sua vez, dão origem a órgãos, estes atuando como componentes essenciais para o funcionamento e interconexão dos diferentes sistemas do corpo humano (digestivo, respiratório, imunológico, endócrino...). De uma outra forma, seguindo uma vertente mais “ecológica”, indivíduos formam populações, as quais dão origem às diferentes espécies, que, ao atuarem de forma conjunta no meio ambiente, integram comunidades e, em mais alta escala, grandes ecossistemas. A observação de tais fatos sugere, fortemente, que o aspecto da modularidade esteja intrinsecamente associado a outros fenômenos ou conceitos de caráter também onipresente, quais sejam o da *recursividade*⁽²⁾ (modularidade como unidade atômica de si mesma) e o da *hierarquização* (módulos dispostos em patamares incrementais de complexidade ou granulosidade) [192].

Replicação é outro fenômeno correlato ao da modularidade, freqüentemente encontrados em combinação. Replicação implica o reuso de estruturas encapsuláveis e com funcionalidades bem definidas. Conforme esse princípio, uma vez que um módulo genérico foi desenvolvido e se “provou” a sua utilidade, este passa

⁽¹⁾ Ao longo deste documento, passa-se a usar uma notação diferenciada quando da introdução de um novo termo, cuja definição se torna importante dentro do contexto. Da mesma forma, utilizar-se-á esse recurso para fins de realce ou para se assinalar alguma palavra transcrita diretamente de outro idioma.

⁽²⁾ Como exemplo, a alta similaridade apresentada por estruturas fractais demonstra a possibilidade de se construir módulos iterativos a partir de si mesmos.

a ser replicado em larga escala [416], possibilitando o surgimento *a posteriori* de novas especializações [70]. Replicação também implica *redundância*, tornando-se, portanto, um mecanismo fundamental a ser empregado sempre que se almeja maiores níveis de robustez, isto é, tolerância a falhas. O recurso da replicação é suportado continuamente quando se leva em conta o processo evolucionário: arquiteturas parciais já codificadas em cadeias de cromossomos podem ser repetidamente utilizadas para fins de produção de sistemas mais intrincados. Como exemplo, analisando-se a estrutura corpórea de um ser humano, é fácil corroborar essa idéia, haja vista a duplicação de membros (pernas, braços, dedos) e órgãos (olhos, pulmões, rins), a organização em bloco de vértebras da coluna cervical, e a disposição maciça de pêlos por toda a superfície da pele. Do mesmo modo, o próprio genoma de todas as formas de vida da Terra é codificado em arranjos bem modulares, os genes, os quais, por sua vez, são composições sequenciais ordenadas formadas por um conjunto bem restrito de bases químicas (outros módulos de menor resolução), a saber, guanina, citosina, timina e adenina/uracila.

A importância dos conceitos de modularidade, recursividade, hierarquização e replicação torna-se ainda bem mais clara quando se almeja analisar o cérebro humano, reconhecidamente o mais elaborado sistema de processamento de informação [318, 401]. Essa “máquina” apresenta uma estrutura notavelmente modular, descritível em variados níveis, desde o neuronal até o lobular [342]. Por causa de sua arquitetura altamente organizada e integrada, o cérebro é capaz de executar, em um espaço mui compacto, uma miríade de funções computacionais. Camadas subseqüentes de neurônios, arranjadas em múltiplos estágios, podem formar representações altamente complexas. Além dessa disposição hierárquica, o cérebro exibe a capacidade de processamento independente de diferentes tipos e modalidades de informação, por meio da formação dinâmica de caminhos (*pathways*) alternativos entre suas regiões — a partir desse artifício é que se torna possível, por exemplo, a integração de interpretações dos distintos estímulos visuais de cor, forma e movimento, que são processados, paralelamente, em zonas anatomicamente separadas [394]. Outras características comumente enaltecidas na literatura são a presença de *inter-* e *intraconexões*, tanto de forma serial como *paralela*, de zonas diferentes do cérebro; e a grande *esparsividade* dessas conexões (em relação à possibilidade de total conectividade), provocando o surgimento de grupos (*clusters*) de neurônios [69].

Não somente sob um caráter puramente neurofisiológico, mas também de acordo com certas teorias neuropsiquiátricas [372], vem a ser fundamental avaliar fenômenos cognitivos complexos, como capacidade de percepção, memória, raciocínio e consciência, inerentes à mente humana e que parecem *emergir* a partir de atividades cerebrais de mais baixo nível (i. e., interações positivas/negativas intensas dos neurônios), consoante a perspectiva da modularidade. Dentro desse contexto, de acordo com a visão de J. Fodor [150], sistemas cognitivos são considerados modulares, uma vez que: *i*) demonstram alto grau de *especificidade*, exibindo padrões computacionais únicos, em conformidade com o domínio em que atuam (estímulos recebidos); *ii*) sua operação é de natureza *inviolável*, ou seja, não é passível de ser comprometida pela ocorrência de outros fatores, quer sejam estes frutos de outros processos cognitivos ou não (p. ex., a crença na existência de um objeto não o leva a ser, necessariamente, percebido ou sentido fisicamente); *iii*) exibem alta velocidade de processamento, já que são autocontidos (*encapsulados*) e não necessitam fazer referência a informações externas para completar suas operações; e *iv*) apresentam saídas bastante simples, facilmente interpretáveis

e com possibilidade de serem *agregadas* em representações mais complexas. Segundo a concepção de Fodor, um módulo seria todo “órgão mental, possivelmente em estado de evolução gradativa, voltado para tratar tipos específicos de informação que demonstrem relevância particular para a espécie [humana]” [138, 150].

Os exemplos acima ilustram quão poderoso e universal é o artifício da modularidade. Em geral, pode-se lançar mão de tal abstração para fins não somente descritivos (ou seja, puramente analíticos), mas também como ferramenta de *prescrição* dirigida ao projeto e síntese de sistemas de elevada complexidade. Seguindo essa linha de argumentação, a estratégia de repartir uma tarefa complicada em subproblemas de menor dificuldade, para depois integrar os vários produtos parciais numa solução final, consagrada pela alcunha de “*dividir para conquistar*”, é constantemente aplicada quando da concepção de novas técnicas científicas e de engenharia. Construir uma casa, decifrar códigos, resolver uma equação ou mesmo escrever uma longa tese como esta são empreitadas que lançam mão do princípio da *decomposição* em módulos.

Dentro da ótica computacional, projetar um sistema equivale a especificar os vários componentes e suas inter-relações, tentando otimizar, concomitantemente, tanto o grau de *separação* como o de *integração* dessas partes quando do atendimento dos objetivos gerais⁽³⁾. O projeto ideal, portanto, traduz-se naquele onde cada componente (com suas funções e comportamento bem definidos) seja passível de ser controlado de forma independente dos demais, sem que, com isso, a combinação efetiva das várias partes se torne comprometida. É nesse sentido que Scheurer [413] propõe uma conceituação mais formal de modularidade. Segundo o autor, “...um módulo é qualquer componente de um sistema que seja ‘bem distinguível’ dos demais, e cujo relacionamento com as outras partes seja o mais ‘simples’ e precisamente estabelecido quanto possível. Isso implica que, em um módulo (ideal), pode-se identificar duas partes: o seu *corpo* (elementos que o caracterizam como único) e a sua *interface* (todas as conexões, lógicas como físicas, com o restante do sistema)”. Exemplos de abstrações computacionais que seguem essa caracterização são muitos, incluindo os conceitos de objetos (componentes) de *software*, *plug-ins*, neurônios artificiais e agentes autônomos (esses últimos, alvos de estudo deste trabalho).

Na seqüência, será discutida, com maior grau de profundidade, a aplicação de abordagens modulares quando da elaboração e implementação de sistemas de *software* adaptativos (ou seja, capazes de se modificar e/ou “aprender”, ao longo do seu ciclo de vida, como resultado de sua própria experiência), sendo este o tema de estudo da presente tese.

1.1 Modularidade aplicada a sistemas inteligentes

Dentre as áreas de pesquisa computacional que mais fazem uso do conceito de modularidade, com intuito de modelagem ou desenvolvimento de técnicas flexíveis e adaptativas, destaca-se a da *Inteligência Artificial* (IA), cujo propósito maior está na caracterização e obtenção de *sistemas inteligentes*. Tais sistemas

⁽³⁾Essa é a filosofia por trás de um princípio fundamental da teoria de desenvolvimento de *software* conhecido como “*separation of concerns*” [413].

compreendem, em sua essência, um conjunto de métodos que tentam emular comportamentos considerados inteligentes como forma efetiva de solucionar problemas.

Conforme uma das primeiras definições, dada por McCarthy e Hayes [313], “uma máquina é dita inteligente se ela é capaz de resolver uma classe de problemas que requerem inteligência para serem solucionados por seres humanos”. Verifica-se, conforme essa aceção, uma forte tendência inicial de se buscar nas atividades comportamentais do ser humano a fonte de inspiração para o desenvolvimento de modelos e técnicas de sistemas inteligentes. A esse ramo da IA, orientado para o tratamento de estruturas *simbólicas* que representam e manipulam explicitamente o conhecimento acerca do domínio-alvo, passou-se a chamar de *IA Simbólica* [392]. *Sistemas Baseados em Conhecimento* e *Sistemas Especialistas* são exemplos notórios dentro desse contexto, apresentando como características primordiais: *i*) a necessidade de se capturar *a priori* todo conhecimento específico (*know-how*) acerca de como o problema deve ser resolvido; *ii*) o emprego de métodos de busca heurística (não-determinística) para a inferência de novos conhecimentos; *iii*) o requisito de se gerar saídas interpretáveis, indicando os passos seguidos até a obtenção das conclusões; e *iv*) um forte embasamento em estratégias *top-down* (“de cima para baixo”), onde o foco se dá, primeiramente, na idealização do sistema como um todo, para depois se tratar aspectos relacionados unicamente às suas partes.

Sob a perspectiva da modularidade, o paradigma de *agentes* de software [214, 402] passou a assumir um papel proeminente dentro da IA. Apesar de não se possuir uma definição clara das propriedades que um módulo deve ter para ser considerado um agente, tal abstração tornou-se bastante difundida mais recentemente, haja vista a necessidade incremental de se construir sistemas inteligentes mais robustos e de larga escala. A propriedade da *autonomia* viabiliza tais objetivos, já que se espera que um agente (com identidade própria e distinguível) seja capaz de realizar suas tarefas de forma independente e deliberada, por meio da atuação e percepção contínuas sobre/do ambiente onde está inserido (tipicamente, o problema em questão).

O emprego do conceito de modularidade também se dá, muitas vezes, na própria concepção da arquitetura de um agente [214]. Desse modo, atividades de planejamento, previsão, sensoriamento, indução (aprendizado) e controle motor passam a ser, por decisão de projeto, alocadas a (sub)módulos com funcionalidades específicas. Por outro lado, é também possível se confeccionar as partes de um agente sob a ótica das tarefas que este irá realizar (isto é, decomposição por tarefas/objetivos), tais como locomover-se, caçar, monitorar, comunicar-se, etc. Independentemente do arranjo empregado, cada bloco lida com uma porção separada do espaço de busca da solução do problema a ser alcançada pelo agente como um todo.

Com o estabelecimento da corrente “behaviourista” da IA, em meados da década de 1980, ficou difundida a idéia de que comportamento inteligente poderia ser reproduzido (ou sintetizado) via o uso maciço de “elementos de inteligência” (ou seja, módulos) relativamente simples, cada um alocado a uma área de conhecimento (*expertise*) particular [324]. Segundo essa abordagem, a complexidade aparente de ações comportamentais inteligentes *emerge* a partir de duas fontes, quais sejam, as *interações* (muitas vezes não-determinísticas) de múltiplas unidades executando assincronamente em paralelo e a inerente complexidade do ambiente onde estas atuam e reagem [64]. Tal linha de pensamento serviu de base para o desenvolvimento de sociedades artificiais compostas por inúmeros módulos autônomos, que exibissem inter-relações de *cooperação*

Tab. 1.1: Algumas diferenças entre paradigmas computacionais. (Fonte: Reggia [388].)

	Computação canônica	Computação bio-inspirada
<i>substrato</i>	silício	carbono, carbono + silício
<i>processamento</i>	seqüencial	maciçamente paralelo
<i>controle</i>	centralizado	descentralizado
<i>interações</i>	global	local
<i>fonte de comportamento</i>	programado	emergente (auto-organizado)
<i>componentes</i>	rápidos, confiáveis	lentos, ruidosos
<i>tolerância a falhas intrínseca</i>	não	sim
<i>construção</i>	projetada (<i>top-down</i>)	evoluída (<i>bottom-up</i>)
<i>flexibilidade</i>	rígida (<i>hard</i>)	adaptativa (<i>soft</i>)

e/ou *competição* e atuassem de acordo com *papéis* bem definidos, em consonância com regras ou protocolos de *coordenação*. Projetar e analisar tais sociedades (organizações) passou a ser o alvo de estudo da subárea conhecida como IA Distribuída (hoje, sob a denominação de *Sistemas Multiagentes*) [131, 232, 355, 475].

Como alternativa (ou complementação) às perspectivas anteriores, há uma tendência, também recente, de se expandir os horizontes de influência e se buscar em outros processos ou fenômenos biológicos novas metáforas para a resolução de problemas computacionais, compreendendo, tipicamente, abordagens *bottom-up* [154, 236, 276]. A iniciativa de se buscar imitar determinados processos naturais a partir de implementações em ambientes artificiais (virtuais) ajuda a entendê-los melhor, ao mesmo tempo que serve como base de inspiração para a concepção de novas metodologias de desenvolvimento de dispositivos computacionais inteligentes (ou seja, exibindo características de *aprendizado*, *adaptação*, *emergência* e *auto-organização*). Essa é a filosofia por trás de sistemas inteligentes ditos *bio-inspirados* [362, 431]. A Tab. 1.1 lista algumas das propriedades exibidas por esse recente paradigma computacional, comparando-o com modelos computacionais de natureza mais “canônica”.

Como parte desse esforço, surgiu uma nova corrente da IA mais voltada para a concepção de métodos de *Inteligência Computacional* (IC), sendo esta vinculada, em linhas gerais, à habilidade de se aprender a lidar com novas situações mediante um ou mais dos seguintes atributos da razão: associação, generalização, descoberta e abstração [153, 516]. Tradicionalmente, a IC abrange três campos de estudo inter-relacionados: Sistemas Conexionistas (mais conhecidos pela denominação de Redes Neurais Artificiais — RNAs) [143, 198, 318, 386], Sistemas baseados na teoria de Conjuntos e Lógica *Fuzzy* [367, 507] e Computação Evolutiva (CE) [24, 30, 152]. Mais recentemente, passou-se a considerar também como subáreas da IC a *Inteligência Coletiva* [56, 249] e os *Sistemas Imunológicos Artificiais* [107, 113].

Das subáreas acima, a que mais se aproxima da IA Simbólica diz respeito aos sistemas nebulosos (*fuzzy*), uma vez que estes também pressupõem a manipulação de conhecimento capturado diretamente de especialistas no problema-alvo (conhecimento esse representado, geralmente, via variáveis e regras lingüísticas

de alto nível, do tipo “Antecedente $\rightarrow$ Conseqüente”). Contudo, tais sistemas apresentam maior robustez e consistência quando da manipulação integrada de “grânulos” de informação em cenários de aplicação exibindo características de incerteza, não-exatidão, vagueza, ou mesmo conflito [365].

Por sua vez, RNAs compreendem sistemas formais [437] que se assemelham às estruturas neurais biológicas e que têm capacidade de processamento computacional adquirida via processos de *aprendizado indutivo e generalização*. Constituem modelos semi-paramétricos⁽⁴⁾ de inferência estatística [197], construídos de forma iterativa (mediante um procedimento ou *algoritmo de aprendizado*) pela interação direta com o problema-alvo (tipicamente representado via conjuntos de dados amostrais) [58]. Segundo tal definição, sistemas conexionistas (como também os algoritmos evolutivos, descritos a seguir) podem ser considerados como um dos paradigmas de suporte à subárea da IA conhecida como *Aprendizado de Máquina* — AM. Os principais objetivos dessa disciplina de estudo são *i)* o desenvolvimento de novas técnicas computacionais que venham a auxiliar no entendimento dos processos de aprendizado natural; e *ii)* a construção de máquinas artificiais capazes de adquirir conhecimento de forma automática, a partir de experiências acumuladas por meio da solução bem-sucedida de problemas anteriores [326, 330]. Dentro desse escopo da AM, encontram-se também embutidos os trabalhos na linha de pesquisa conhecida como *Learning from Data* [80, 491], onde os problemas de aprendizado a serem tratados reduzem-se àqueles relativos à manipulação de dados amostrais para fins de caracterização estatística e/ou extração de conhecimento. A maioria dos projetos apresentados neste documento encaixa-se nesse último contexto.

Finalmente, com respeito à Computação Evolutiva, pode-se dizer que esse é o ramo da IC centrado na construção de modelos computacionais, sob a forma de *Algoritmos Evolutivos* (AEs), baseados em um ou mais aspectos da teoria da evolução natural [105]. Tais algoritmos constituem procedimentos de natureza heurística e estocástica direcionados à resolução de problemas computacionais (sendo, nesse caso, considerados como métodos de busca, otimização e aprendizado) [178], à modelagem de processos evolucionários [24, 152, 207], ou mesmo à síntese de dispositivos artificiais exibindo propriedades/comportamentos típicos de organismos vivos (área conhecida como *Vida Artificial* [276]). Existem também extensões mais recentes de algoritmos evolutivos que tentam modelar relações ecológicas (simbióticas) entre espécies artificiais distintas com o intuito de melhorar o desempenho do processo adaptativo, tanto em termos de eficiência (ajuste automático e progressivo da convergência à solução final) como em termos de eficácia (qualidade da solução ou comportamento gerado). Dentre tais abordagens, destacam-se as de gênero *coevolucionário*, muitas das quais se calcando, explicitamente, no princípio da modularidade [104, 358]. Inserida nesse contexto está a *arquitetura coevolucionária cooperativa*, proposta recentemente por Potter e De Jong [238, 375, 377] e que serviu de base para uma parcela substancial dos projetos reportados no decorrer desta tese.

As três subáreas da IC acima caracterizadas também apresentam suas limitações. Por exemplo, no desenho de sistemas nebulosos, muitas vezes torna-se difícil a configuração, de modo otimizado, dos formatos das funções de pertinência que definem as variáveis linguísticas do problema, bem como das partes

⁽⁴⁾RNAs podem ser caracterizadas também como modelos não-paramétricos, desde que a especificação de sua arquitetura não seja feita *a priori*, mas sim no curso do próprio processo de aprendizado.

antecedentes das regras nebulosas, sendo que, tradicionalmente, tais decisões passam a ser estipuladas arbitrariamente pelo projetista do sistema [229, 293]. De uma outra forma, a definição manual *a priori* da arquitetura de uma RNA, em total acordo com os requisitos do problema em vista, pode se tornar uma atividade bastante ineficiente, mormente quando se lança mão de estratégias de tentativa e erro. Finalmente, com relação aos algoritmos evolutivos, a tarefa de se configurar uma função de *fitness* apropriada, que possa avaliar requisitos subjetivos impostos em um problema complexo, nem sempre é de fácil concepção ou mesmo representável via equações matemáticas.

De todo modo, como são técnicas bem flexíveis e que exibem características não-coincidentes (i. e., complementares), parece ser bem intuitivo agregá-las entre si (e com outras técnicas) a fim de se gerar soluções incrementalmente mais robustas e eficazes. Essa é a motivação por trás dos chamados *Sistemas Inteligentes Híbridos* (SIHs) [317], que se caracterizam pela *agregação* (combinação, fusão ou transformação) [180] de subsistemas individuais, de tal forma que a solução final apresente um potencial maior que a simples soma dos potenciais isolados de cada subsistema constituinte. São exemplos desses sistemas *i*) as diversas arquiteturas de *redes neurais nebulosas* (ou *neurofuzzy*) [148, 228, 229, 293], combinando a capacidade de aprendizado das RNAs com a representação explícita de conhecimento via regras *fuzzy*; *ii*) as abordagens *geno-fuzzy* [98, 99, 119, 242, 363], onde algoritmos evolutivos customizados são projetados para otimizar parâmetros de sistemas nebulosos complexos; *iii*) as abordagens *neuro-evolutivas* [20, 47, 108, 151, 334, 501], que tentam projetar/configurar arquiteturas de RNAs de forma automática via algoritmos de aprendizado (busca) de cunho evolutivo; e *iv*) os *algoritmos evolutivos híbridos* (também conhecidos como *algoritmos meméticos*) [7, 340, 400, 481], que tentam embutir conhecimento disponível acerca do problema-alvo quando da configuração dos operadores genéticos, combinando o poder de busca global, inerente a algoritmos evolutivos, com o poder de busca local, proporcionado, por exemplo, por métodos do tipo “subida da colina” (*hill climbing*) ou “descida do gradiente” (*gradient descent*). Sob essa perspectiva puramente *holística*, SIHs se apresentam como uma metodologia bastante propícia para o estudo formal e/ou empírico de novas formas de sistemas modulares adaptativos.

À luz dessas primeiras colocações, que explicitam o poder por trás do conceito de modularidade, bem como sua relevância para o desenvolvimento de máquinas inteligentes, passa-se, a seguir, a delimitar o escopo de investigação do presente trabalho, apresentando o seu objetivo e um resumo dos projetos desenvolvidos. Ao final do capítulo, é anunciado um esboço de como a tese está estruturada.

1.2 Objeto de estudo

Como ressaltado por Medsker [317], nem sempre a utilização de SIHs leva necessariamente a um ganho de desempenho do sistema como um todo quando comparado àquele obtido pelas técnicas aplicadas em separado. Pelo contrário, algumas vezes os resultados obtidos por um sistema híbrido são inferiores, segundo uma ou mais métricas de avaliação. Mesmo porque, como o processo de integração de técnicas deve considerar as peculiaridades de cada aplicação, não existe uma sistematização única para a implementação

automática de sistemas híbridos. Destarte, o que se pode concluir é que SIHs podem ser muito eficazes, desde que justificados, desenvolvidos e utilizados corretamente.

Em conformidade a essa premissa, e para se tirar real proveito do grande potencial por trás de abordagens modulares como ferramenta dirigida à engenharia de sistemas inteligentes [250], algumas questões-chave precisam ser averiguadas antecipadamente, durante a fase de análise do problema e avaliação dos métodos-candidatos à solução. Dentre tais indagações, este autor destaca:

Natureza do problema e do método-candidato à solução do problema:

1. O problema apresenta grau de **complexidade** elevado? Impõe requisitos de não-linearidade, não-continuidade, caoticidade, não-diferenciabilidade, não-estacionariedade ou de natureza não-determinística?
2. É factível decompor o problema em subtarefas? Se sim, de forma **temporal** e/ou **espacial**? De modo **funcional** ou **guiado pelos dados**? Com ou sem **sobreposição** entre as partes?
3. Existe alguma técnica **específica** para tratar o problema? Em havendo, há algum agravante que possa comprometer o desempenho dessa técnica? Caso não haja uma técnica específica ou esta apresente aspectos desfavoráveis que inviabilizem a sua aplicação no domínio em estudo, existe algum método de IA passível de ser empregado ou **customizado** para o mesmo fim, sendo que tal método, por sua vez, também permite ser decomposto em submétodos?
4. Há espaço para se empregar métodos de essência **construtivista**, pelos quais a arquitetura da solução seja delineada de forma progressiva, ou métodos de cunho **reducionista**, mediante os quais uma solução mais parcimoniosa seja gerada a partir de outras mais complexas? Há ainda a possibilidade de se recorrer a técnicas **complementares** que, quando combinadas, levem a uma solução adequada (mesmo que aproximada) para o problema (por exemplo, mesclar métodos construtivos e subtrativos)?

De um outro modo, quando da fase de planejamento da arquitetura da solução do problema, outros aspectos relevantes, sob a ótica do conceito de modularidade, necessitam também ser considerados, tais como:

Natureza da solução (sistema modular):

5. Como deve o sistema resultante ser **organizado**: verticalmente (hierarquia), horizontalmente (em um só nível) ou de ambos os modos? Quantos módulos usar (em cada nível)?
6. Como **selecionar**, dentre vários blocos-candidatos, aqueles que integrarão efetivamente o sistema?
7. Deve-se permitir **redundância** (mesma sub tarefa executada paralelamente) e/ou **complementaridade** (subtarefas interdependentes) entre blocos de um mesmo nível?

Natureza da interação dos módulos:

8. Como as partes (módulos) que compõem a solução do problema deverão interagir entre si? De forma **estática** ou **dinâmica**? De forma **cooperativa** ou **competitiva**?
9. Como **coordenar** a atuação dos módulos de forma a que se tenha **coerência** no comportamento do sistema como um todo?

Natureza dos módulos:

10. Deve-se recorrer a módulos **pré-especificados** ou **emergentes**? **Autônomos** ou **controlados**?
11. Deve-se aplicar módulos de funcionalidade/especificidade **homogênea** ou **heterogênea**? Módulos com alta ou baixa **granulosidade**?
12. Como determinar a verdadeira parcela de **contribuição** de cada módulo para a resolução eficaz do problema como um todo?

O objetivo maior por trás desta tese foi o de investigar, de forma sistemática, o potencial exibido por ferramentas de Computação Evolutiva quando aplicadas no desenvolvimento automático de sistemas inteligentes modulares, tendo como referencial os pontos supracitados. Nesse estudo, foram realizados vários experimentos, concentrando-se principalmente em três domínios-alvo: A) treinamento coevolucionário de redes neurais heterogêneas; B) projeto automático de comitês de máquinas; e C) emergência de organizações (times) em sistemas multiagentes. Cada um dos pontos acima mencionados foi alvo de consideração em, pelo menos, uma dessas três linhas de investigação. No que segue, passa-se a examinar, de forma sucinta, a motivação do estudo realizado em cada um desses domínios, relatando as atividades desenvolvidas. Uma análise retrospectiva dos projetos, comentando acerca da relevância das contribuições técnicas alcançadas, é postergada para o capítulo de conclusão.

1.2.1 Treinamento coevolucionário de redes neurais heterogêneas

Os modelos mais simples de RNAs *feedforward* assumem módulos simples (neurônios) com funções de ativação idênticas, quase sempre do tipo sigmoidal, de modo a se asseverar, de antemão, que haja saturação nas não-linearidades geradas pela rede — para mais detalhes, vide Cap. 2. Entretanto, para se garantir um incremento substancial no poder de representação do modelo final, sem que, para tanto, seja requerido um aumento proporcional na quantidade de recursos (nós e pesos) da rede, vem a ser pertinente permitir-se que diferentes formatos de funções de ativação sejam selecionados e combinados em concordância com os requisitos do problema. *Redes neurais heterogêneas* constituem uma variante de redes *feedforward* que segue de perto essa filosofia. Nesse domínio de estudo, uma primeira motivação foi a de caracterizar/explorar as propriedades exibidas por tais modelos neurais. Ademais, como alternativa à configuração manual (de tentativa e erro), passou-se a investigar mais detalhadamente o potencial de se projetar automaticamente instâncias dessas redes por meio da aplicação de um método coevolucionário hierárquico customizado (denominado HCGA),

orientado para utilizar otimizadamente os recursos computacionais disponíveis (neurônios e pesos sinápticos). Quando comparada com outras estratégias construtivas de RNAs em problemas de classificação, os resultados comprovaram a eficácia da abordagem HCGA, apontando a sua tendência constante em apresentar altas taxas de classificação correta.

1.2.2 Projeto automático de comitês de máquinas

Não é errado dizer que os modelos mais convencionais de redes neurais apresentam uma estrutura topológica de natureza *monolítica* [69]. Como consequência, tais modelos costumam apresentar problemas de desempenho quando têm que arcar com tarefas de maior complexidade, sendo, uma vez ou outra, “achacados” pelo problema da *maldição da dimensionalidade* [165]. Muitas propostas vêm sendo endereçadas a fim de se remediar essa situação, algumas delas centradas em conceitos da teoria de regularização [198], outras embasadas em uma recém-difundida metodologia de aprendizado conhecida como *máquinas de vetores-suporte* [418, 463], bem como outras recorrendo à modularidade como conceito primário. Todavia, o emprego por si só de técnicas modulares *não* significa garantia de melhoria sempre, já que outras dificuldades são trazidas à tona, tais como a de se estruturar de maneira correta os módulos e/ou o conjunto de dados, a de especificar de antemão as suas inter-relações e a de se indicar o modo de se decompor a tarefa de uma forma mais otimizada. A abordagem mais fácil continua a ser a de se evocar o conhecimento de especialistas no domínio de aplicação, o que nem sempre é factível ou mesmo eficiente — o tempo envolvido no projeto pode não ser admissível.

Como alternativa, aparecem as diversas abordagens de *comitês de máquinas* [426, 452], seguindo metodologias formais fundamentadas em conceitos de estatística, processamento de sinais (teoria da informação), teoria de sistemas (de controle) e teoria da decisão. Dentre tais abordagens se destacam (1) os vários métodos de *ensembles* de estimadores (para a combinação de modelos redundantes, treinados ou configurados de maneiras distintas) [191, 253, 262], incluindo aqui as diferentes versões de técnicas de reamostragem aleatória dos dados de treinamento (a fim de se garantir, *a priori*, maior diversidade na geração dos módulos, focando sobre regiões mais complicadas do espaço de entradas) [63, 128, 160]; e (2) as arquiteturas de *misturas de especialistas* (para o particionamento dinâmico das tarefas e alocação das subtarefas a módulos adequados) [187, 198, 226]. Neste domínio de estudo, almeja-se investigar a aplicação de algoritmos evolutivos mais elaborados para a construção automática de sistemas conexionistas multi-modulares⁽⁵⁾ [59, 81, 208, 259, 266, 281, 296, 298, 504]. Os modelos propostos envolvem desde o desenho e ajuste adaptativo dos módulos até a seleção e combinação destes, passando também pela escolha customizada dos atributos e dados amostrais para a fase de treinamento. Um conjunto extenso de experimentos conduzidos, envolvendo problemas de classificação e previsão de séries temporais, serviu de instrumento empírico para a avaliação de desempenho dos sistemas resultantes.

Nesse contexto, torna-se proeminente ressaltar a proposta do modelo arquitetural de gênero coevo-

⁽⁵⁾Aqui está-se referindo também a agregados de máquinas de vetores-suporte.

lucionário e hierárquico de nome COHENN, cujo objetivo primeiro é o de prover um tratamento uniforme das três etapas comumente envolvidas quando da configuração de *ensembles* de estimadores — vide discussão à Seção 4.1. Neste texto, maior destaque é dado à caracterização e avaliação dessa abordagem pelo fato de a mesma congregar, sob um mesmo arcabouço conceitual, uma série de decisões de projeto e implementação também adotadas, de forma não-integrada, em outros estudos da tese. Além disso, com COHENN, tornou-se possível investigar aspectos concernentes à grande parte das questões introduzidas no início desta seção. Como exemplo, mediante a execução de um protótipo desse modelo, pode-se avaliar como se dá a integração e a interação de módulos neurais exibindo diferentes níveis de granulosidade, envolvendo desde neurônios e arquiteturas de redes neurais simples até arranjos topológicos mais elaborados, como *ensembles* e *ensembles* de *ensembles*. Desse modo, COHENN pode ser encarado como o projeto-chefe em que se sustentam as contribuições deste trabalho.

1.2.3 Organização emergente de times de agentes autônomos

Este domínio apresenta uma perspectiva de investigação diferente da dos demais. O interesse aqui *não* se traduz em atender a índices de desempenho quantitativos, mas sim em avaliar a factibilidade de se reproduzir, artificialmente, padrões de comportamento coletivo similares àqueles apresentados por conjuntos de agentes autônomos naturais. A motivação maior passou a ser a de explorar a habilidade de algoritmos evolutivos simbióticos (i. e., coevolutivos) quando da definição não-manual de estratégias de coordenação em times virtuais de futebol de robô. Sendo assim, mediante simulações controladas, detectou-se o surgimento de padrões de organização espacial dos jogadores, como decorrência do fato de os papéis desses agentes terem sido definidos de forma complementar e automática via a indução progressiva de espécies artificiais no ecossistema de atuação (ou seja, o campo de futebol).

1.3 Organização do documento

Este texto reporta os diversos projetos realizados por este autor na área de sistemas inteligentes modulares, dando-se destaque àqueles empregando técnicas evolucionárias. Almejando-se tornar autocontida esta dissertação e fazendo-se bom uso do próprio princípio da modularidade, optou-se por organizá-la em nove capítulos e três apêndices, os quais são descritos brevemente a seguir.

O primeiro dos capítulos remete o leitor a pensar sobre a importância de se estudar aspectos de modularidade em sistemas computacionais inteligentes, traz um elenco de requisitos a serem observados quando da análise e concepção de sistemas inteligentes modulares, apresenta o objetivo da tese e uma descrição sucinta sobre os domínios dos projetos que a compõem.

A razão por trás dos Caps. 2 a 5 é a de apresentar um estudo contemplativo acerca dos diferentes tipos de sistemas inteligentes modulares abordados nos experimentos relatados nos Caps. 7 e 8. Sempre

que pertinente, ao longo da exposição, passa-se a situar os conceitos introduzidos sob a perspectiva dos projetos conduzidos. Redes neurais artificiais (Cap. 2), máquinas de vetores-suporte (Cap. 3) e comitês de máquinas (Cap. 4) encaixam-se dentro da grande área interdisciplinar de Aprendizado de Máquina, cujas noções (conceitos básicos e critérios de avaliação) são expostas no Apêndice A. Por outro lado, no Cap. 5, são anunciadas algumas das principais propriedades intrínsecas ao conceito de agente de *software*, será caracterizado o relevante problema de coordenação em sistemas compostos por vários desses módulos autônomos, e serão discutidas algumas das várias linhas de pesquisa destinadas à implementação de mecanismos de coordenação em ambientes computacionais, com maior ênfase dada àquelas de cunho cooperativo.

O Capítulo 6 destina-se a uma revisão geral dos principais conceitos e definições por trás de meta-heurísticas inspiradas em processos evolucionários, evidenciando as propriedades de abordagens coevolutivas e evolutivas híbridas (por sua relevância para este trabalho e por se tratarem de extensões mais recentes) e mencionando algumas das diversas aplicações de CE em sistemas inteligentes modulares.

Em seguida, passa-se a descrever, com mais detalhes, os projetos sumarizados na Seção 1.2. O sétimo capítulo versa, especificamente, sobre a abordagem coevolucionária híbrida dirigida ao desenvolvimento completo de *ensembles* heterogêneas de redes neurais. Nesse espaço, traz-se a especificação da arquitetura proposta e de seus vários componentes, e é reportada uma série de experimentos realizados, mediante um protótipo implementado da arquitetura, para fins de análise comparativa com outras abordagens mais “consagradas” existentes na literatura. Esse capítulo é suportado por dois outros adendos, Apêndices B e C, sendo o primeiro designado à descrição dos diversos métodos de combinação de estimadores implementados e o segundo destinado a uma breve exposição dos vários conjuntos de dados empregados como instâncias de problemas reais de classificação.

Por sua vez, o Cap. 8, estruturado na forma de artigos publicados, compreende os demais projetos de pesquisa conduzidos pelo autor na área de engenharia de sistemas inteligentes modulares. Cada publicação é introduzida por um breve resumo em português e seguida de alguns comentários adicionais pertinentes (p. ex., explicando detalhes extras). O final do capítulo é reservado a uma apresentação condensada de um compêndio de publicações correlatas, inseridas também no contexto de sistemas inteligentes.

O Capítulo 9 traz as conclusões sobre este tratado, enaltecendo as suas contribuições. Nesse espaço, são discutidas também algumas sugestões de trabalhos futuros na área de engenharia de sistemas inteligentes modulares que, possivelmente, venham a estender o conjunto de abordagens apresentadas ou aplicá-las em outros domínios de investigação.

Capítulo 2

Redes neurais artificiais com múltiplas camadas

“O cérebro é um órgão maravilhoso: começa a trabalhar de manhã, assim que você se levanta, e não pára até você chegar ao trabalho.”

Robert Frost

Tarefas complexas, exigindo habilidades de raciocínio e reconhecimento de padrões, tornam-se muitas vezes extremamente difíceis de serem automatizadas. Por outro lado, essa não é a norma quando se observa o rico repertório de atividades sensoriais e cognitivas à disposição dos animais na natureza. Reconhecimento visual, controle motor, memória e capacidade de abstração são operações realizadas, concorrentemente e de maneira muito eficiente, pela máquina cerebral. Nesse sentido, parece ser bastante “lucrativo” tentar entender e imitar, artificialmente, tais processos cognitivos e sensoriais, com o intuito de se conceber novas metáforas computacionais para a resolução de problemas não-triviais — como os de engenharia e de tomada de decisão. Essa é a filosofia embutida no desenvolvimento de sistemas conexionistas, dando origem às redes neurais artificiais.

Formalmente, uma RNA compreende uma estrutura em grafo direcionado e rotulado cujos nós (*neurônios*) realizam alguma atividade de processamento de informação [318, 437]. Às arestas do grafo, dá-se o nome de *conexões sinápticas*, ao passo que, aos rótulos, cujo papel é o de identificar níveis de excitação ou inibição entre os neurônios, dá-se o nome de *pesos*. Diferentes escolhas para os pesos (parâmetros do modelo neural) resultam em distintas funções matemáticas representáveis pelo sistema. Dada uma configuração inicial desses rótulos e dada a tarefa que se queira realizar, um *algoritmo de aprendizado* (indutor) deve ser adotado a fim de que se determine os valores corretos dos pesos (i. e., gere a solução do problema na forma de um *estimador*) num número finito de iterações. De uma forma sucinta, pode-se dizer que uma rede neural é caracterizada por sua *arquitetura*, envolvendo o modelo (tipo) dos neurônios e seu padrão de interconexão (número de camadas), e por sua *dinâmica*, definindo as suas propriedades operacionais (como aprender, associar e recuperar padrões).

A literatura relativa a sistemas conexionistas já é, atualmente, bastante vasta. Existem várias referências consagradas tratando dos diversos aspectos relativos ao tema, desde as de cunho introdutório (e mais computacional) [143, 243, 318, 386] até aquelas orientadas para descrições mais formais [53, 77, 198, 437]. Em seguida, serão examinados alguns dos principais tópicos em RNAs, dando-se ênfase a redes neurais *feedforward*. O objetivo é prover um embasamento teórico para as contribuições apresentadas ao longo da tese (Caps. 7 e 8).

2.1 Aplicações em estudo

Do ponto de vista da aplicação, as tarefas computáveis por uma RNA, como qualquer estimador, podem ser classificadas de acordo com o *paradigma de aprendizado* inerente ao seu processo de treinamento. Dentre tais paradigmas (ver Apêndice A), maior destaque será dado, no decorrer deste texto, àquele de gênero supervisionado. Dentre as classes de possíveis aplicações dentro desse contexto, os trabalhos expostos nos Caps. 7 e 8 se concentram em duas principais, quais sejam, *classificação de padrões* e *predição de eventos futuros*.

A primeira classe de aplicações refere-se ao problema de se atribuir um dado objeto a um dentre vários (digamos, M) grupos predeterminados, tendo como referencial o conjunto de atributos (características) que descrevem o objeto. Exemplos de tarefas que se enquadram nessa definição incluem reconhecimento de indivíduos via seus padrões biométricos (impressão digital, fala, face, íris...), avaliação de risco de crédito bancário, diagnósticos médicos sobre determinados quadros clínicos, classificação funcional de novas proteínas de acordo com sua estrutura molecular, discriminação de imagens de satélite para sensoriamento remoto, dentre outras⁽¹⁾. Nessa situação, há necessidade de se ter disponível, em primeira mão, um conjunto de amostras (pré-processadas) de treinamento, contendo pontos representativos de todas as M classes — um conhecimento prévio sobre o domínio do problema-alvo pode ser, às vezes, imprescindível aqui. Por intermédio desse conjunto de treinamento, um indutor inferirá regras de pertinência para cada classe, originando, por conseguinte, um *classificador* (ou *hipótese* [326, 330]) neural; este será usado, posteriormente, para a atribuição correta de novos objetos.

Por outro lado, a atividade de previsão de eventos futuros pode ser considerada como um caso especial de problema de aproximação de função (regressão), onde os valores produzidos pela função-geratriz F_g do processo que se quer modelar são representados via *séries temporais*: seqüências ordenadas de observações medidas ao longo do tempo, de forma contínua ou discreta ($\{s_1, s_2, \dots, s_t, \dots, s_N\}$) [36, 333, 470, 509]. O objetivo aqui passa a ser o de antever valores futuros de uma única grandeza, quando se está lidando com séries *univariadas*, ou de mais de uma grandeza simultaneamente, para séries *multivariadas*, levando-se em conta suas medidas passadas (ou de outras variáveis exógenas). Nesse cenário, distinguem-se estimativas de curto prazo (“um passo à frente”) e estimativas de longo prazo (“múltiplos passos à frente”). Para

⁽¹⁾O Apêndice C traz uma lista com várias instâncias de problemas de classificação, tendo sido estas empregadas nos experimentos reportados nos Caps. 7 e 8.

estimativas de longo prazo, existem duas estratégias comumente acolhidas quando do projeto do preditor (no caso, RNA) [471, 509]: (1) *método direto*, onde a saída do preditor já deve ser a estimativa para k passos à frente; e (2) *método recursivo*, onde o preditor, que produz uma estimativa para um único passo à frente, é usado de forma iterativa, ou seja, alguns valores preditos por ele serão usados na geração de outros. Outro conceito importante é o de *janela de predição*, que sinaliza o número e o índice das observações passadas (*lags*) a serem usadas como variáveis independentes (atributos) no modelo de regressão neural.

Weigend *et al.* [471] advogam que uma abordagem preditiva bem sucedida deve fundamentar-se ou no conhecimento das leis estatísticas (físicas) intrínsecas ao processo gerador dos dados, ou na descoberta de regularidades empíricas relevantes diretamente das observações, sendo esta a estratégia seguida quando do emprego de RNAs. Todavia, essa não é uma tarefa das mais fáceis, porquanto a maioria das séries temporais não-artificiais (dentre elas, as de natureza financeira ou referentes a fenômenos físicos, como as estudadas no artigo da Seção 8.3) exhibe particularidades de difícil trato: são inerentemente *ruidosas*, *não-estacionárias* e *deterministicamente caóticas* [6, 387, 470]. A primeira das propriedades acima diz respeito à carência de informação precisa acerca do comportamento exibido pelo processo/fenômeno no passado, de tal modo que nem sempre é factível se apreender fielmente a dependência existente entre observações futuras e passadas. A informação não detida pelo modelo passa a ser considerada, portanto, ruído. A característica de não-estacionariedade, por sua vez, traz como consequência o fato de que a distribuição de probabilidade dos valores pertencentes a essas séries temporais varia, geralmente, com o passar do tempo. Descobrir como se dá essa dinâmica de variação nem sempre é factível. Finalmente, por “deterministicamente caóticas”, quer-se dizer que tais séries exibem, tipicamente, aleatoriedade num horizonte curto, mas comportamento determinístico a longo prazo — a conhecida *hipótese do mercado eficiente* [387], por exemplo, se pauta nessa premissa.

2.2 Propriedades gerais

Diversas são as características distintivas exibidas por sistemas conexionistas que justificam o seu alto grau de aplicabilidade. Diferentemente de modelos estatísticos tradicionais [130, 165], por exemplo, redes neurais constituem modelos adaptativos semi- ou não-paramétricos derivados diretamente dos dados [77, 197]. Nesse sentido, é costumeiro se dizer que RNAs “permitem com que os dados falem por si próprios”⁽²⁾, já que não impõem *a priori* nenhuma condição sobre as propriedades funcionais a serem exibidas pelas amostras (tais como o formato da distribuição de probabilidade intrínseca ao processo gerador dos dados). Essa habilidade de *aprender com a própria experiência* se torna bastante profícua quando não se tem conhecimento prévio suficiente acerca do problema-alvo.

Por outro lado, a capacidade de *generalização* de uma RNA está associada à sua habilidade em dar respostas coerentes a situações não contempladas, ou mesmo vislumbradas, durante a fase de aprendizado.

⁽²⁾Do inglês, “let the data speak from themselves”.

Isso se traduz na faculdade de produzir saídas bem similares àsquelas desejadas para exemplos que não pertençam ao seu conjunto de treinamento (ver discussão na Seção 2.5).

Ademais, RNAs com múltiplas camadas são consideradas *aproximadores universais* de funções contínuas [39, 209, 210], com capacidade de aproximar, dentro da faixa de precisão que se deseja, qualquer mapeamento multivariável não-linear — desde que, para isso, tenha recursos (que se traduzem em parâmetros livres, sem interpretação) suficientes. Para esse fim, redes neurais dispõem de um farto repertório de formas funcionais bem flexíveis (ver Seção 2.4), algo bastante interessante quando se deseja capturar dependências não-lineares entre observações futuras e passadas em problemas de predição, por exemplo. A complexidade das funções representáveis por um modelo de RNA depende diretamente do número de camadas intermediárias compondo a sua topologia, bem como do número de neurônios em cada uma dessas camadas [437].

Em se tratando de problemas de classificação, pode-se dizer que redes neurais *feedforward* (Seção 2.3) identificam-se como extratores de novas características (*feature extractors*), como decorrência das transformações não-lineares realizadas por seus neurônios escondidos, em consonância com o *Teorema da separabilidade de padrões* [97]. Resumidamente, esse teorema garante que, ao se projetar não-linearmente um problema de classificação em um espaço de mais alta dimensionalidade, aumenta-se a probabilidade de os padrões de entrada se tornarem linearmente separáveis nesse novo espaço [198]. De um modo mais formal, seja f uma função mapeando $\mathbb{R}^D$ a $\mathbb{R}^h$, onde D designa o número de atributos de entrada e h um número inteiro positivo qualquer, então o teorema estabelece que qualquer conjunto de P vetores D -dimensionais escolhidos aleatoriamente é passível de ser separável por f (i. e., f -separável) com probabilidade dada por $\sum_{p=0}^{h-1} \frac{\binom{P-1}{p}}{2^{P-1}}$. Para um P fixo, essa probabilidade cresce à medida que h aumenta, sendo ≈ 1 quando $P \approx h$, ou seja, quando o número de neurônios escondidos é praticamente igual ao número de amostras de entrada [318].

Outro ponto a favor de modelos neurais, tendo em vista tarefas de classificação, está no fato de que RNAs *feedforward* podem prover, simultaneamente, estimativas diretas de *probabilidades a posteriori* para todas as M classes de um problema [294, 393, 468]. Assim, a saída y_j de uma dada instância de RNA passa a quantificar a probabilidade *a posteriori* associada à classe c_j como sendo igual a $p(c_j|\mathbf{x})$, que representa a probabilidade da classe correta ser c_j dado um vetor de entrada específico $\mathbf{x}$. Pela regra de Bayes [130], tais probabilidades podem ser determinadas mediante

$$p(c_j|\mathbf{x}) = \frac{p(\mathbf{x}|c_j)p(c_j)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|c_j)p(c_j)}{\sum_{c_j^* \in C} p(\mathbf{x}|c_j^*)p(c_j^*)},$$

onde C é o conjunto formado pelas M classes, $p(\mathbf{x}|c_j)$ é a probabilidade condicional (verossimilhança ou plausibilidade) de se ter $\mathbf{x}$ dado que sua classe é c_j , $p(c_j)$ é a probabilidade *a priori*⁽³⁾ da classe c_j , e $p(\mathbf{x}) = \sum_{c_j^* \in C} p(\mathbf{x}|c_j^*)p(c_j^*)$ é a probabilidade não-condicional da entrada.

Diz-se que uma RNA provê um erro bayesiano mínimo de classificação quando, após computar as

⁽³⁾ A probabilidade *a priori* de uma classe denota a probabilidade de se selecionar, aleatoriamente, uma nova amostra de entrada pertencente àquela classe, sendo normalmente obtida pela medição da frequência de ocorrência dos exemplos dessa classe em relação às amostras de todas as M classes possíveis.

probabilidades posteriores de todas as M classes, ela dá como resposta o índice da classe cujo valor da probabilidade *a posteriori* é máximo. Essa possibilidade de interpretação probabilística de RNAs implica várias vantagens, dentre as quais Richard e Lippmann [393] ressaltam: (i) ela simplifica a criação de *thresholds* de rejeição⁽⁴⁾; (ii) ela permite com que a saída seja usada para se minimizar, concomitantemente, funções de custo alternativas; (iii) ela possibilita compensar eventuais diferenças entre medidas de probabilidade *a priori* apresentadas por uma mesma classe em conjuntos diferentes de treinamento e teste; e (iv) ela provê meios efetivos de se combinar as saídas geradas por diferentes estimadores. Esse último resultado é de particular relevância para o estudo de *ensembles* de classificadores (Seção 4.1), servindo de base teórica para a concepção de diferentes métodos de combinação [497].

Sob uma outra perspectiva, vale salientar que modelos neurais com múltiplas camadas ostentam a propriedade de representação distribuída da solução, ou seja, o comportamento coletivo (sinérgico) da rede fica sendo o responsável pelo seu poder computacional, na medida em que nenhum neurônio isolado armazena informação específica e tem “ciência” do processamento em curso em toda a estrutura — notadamente, uma abordagem *bottom-up*. Tal característica operacional se contrapõe a abordagens algorítmicas tradicionais, calcadas em seqüências de instruções.

Por esse motivo, é comum considerar RNAs como sistemas do tipo “caixa-preta”, uma vez que sua representação interna dos conceitos apreendidos não é facilmente interpretável por humanos. A fim de se dirimir tal deficiência, existem vários procedimentos propostos na literatura voltados à extração do conhecimento embutido em instâncias treinadas de RNAs, sendo que a maioria deles gira em torno do uso de regras de mais alto nível (*IF-THEN*), quer sejam estas baseadas em funções booleanas (p. ex., o método proposto por Tsukimoto [454]) ou não (p. ex., o método baseado em regras *fuzzy*, proposto por Benítez *et al.* [50]). Em verdade, o acoplamento de RNAs com outros tipos de máquinas de indução, ditas simbólicas (como árvores de decisão e redes semânticas), em concordância com a filosofia dos SIHs (ver Seção 1.1), tem sido justificado como um caminho efetivo para se lidar com tais problemas de interpretação em RNAs [173, 447]. Como uma metodologia alternativa, também concordante com o hibridismo de técnicas, apresenta-se a abordagem modular MASCoNN [290] (vide Seção 8.5). Esta se baseia numa arquitetura composta por neurônios especiais, modelados como agentes autônomos, que ficam responsáveis pela coordenação dos demais (“neurônios trabalhadores”) para a resolução de um dado problema complexo — no caso, o estudo feito foca, na sua maior parte, em problemas de identificação, previsão e controle de sistemas não-lineares. Nesse *framework*, a interpretação do *modus operandi* da rede final fica mais facilitada, já que (a) é possível se armazenar, na base de conhecimento dos agentes coordenadores, os passos seguidos até a decomposição final do espaço de entrada entre os neurônios trabalhadores; (b) por ser uma abordagem construtiva (ver Seção 2.5), o número de neurônios trabalhadores tende a ser pequeno; e (c) as funções de ativação dos neurônios são unidimensionais (*ridge functions* [197, 271]), isto é, de fácil visualização.

Outra característica exibida por modelos neurais é a de tolerância a dados ruidosos, haja vista que,

⁽⁴⁾Diz-se que um estimador *rejeita* uma amostra quando não consegue produzir uma decisão não-ambígua para aquela amostra, ou seja, quando, dentre M classes, pelo menos duas delas têm iguais chances de serem corretas.

mediante seu processo de aprendizado descentralizado e iterativo, RNAs podem recuperar (e generalizar sobre) padrões intrínsecos existentes em dados incompletos, corrompidos, ou mesmo conflitantes. Mais ainda, o desempenho da rede não fica completamente comprometido quando parte de seus neurônios são removidos (ou falham continuamente), algo difícil de acontecer em abordagens algorítmicas sequenciais. Daí, é comum declarar-se que modelos neurais exibem propriedade de degradação suave (*graceful degradation*) de desempenho em caso de falhas/corrupção de processamento [143].

Finalmente, é interessante mencionar a respeito do *dilema estabilidade-plasticidade*, cuja essência pode ser resumida pela seguinte indagação: como se projetar uma máquina neural de modo que ela seja *plástica* (i. e., adaptável) o suficiente para aprender novos conceitos sempre que eles apareçam, ao mesmo tempo que *estável* o bastante para preservar todo o conhecimento apreendido anteriormente? De um outro modo: “como o sistema neural sabe ‘quando’ e ‘como’ chavear para seu modo estável, a fim de atingir estabilidade sem rigidez, e para seu modo plástico, almejando plasticidade sem caos?” [243]. Esse tipo de questão se torna deveras relevante quando se almeja empregar RNAs para fins de aprendizado contínuo (*lifelong learning*), onde há a possibilidade de se usar experiência anterior para reduzir a complexidade de aprendizado em novas tarefas. A abordagem MASCoNN, supracitada, por adotar elementos de processamento com propriedades mais elaboradas (agentes deliberativos), parece ser apropriada para trabalhos nessa linha, embora isso necessite ser mais bem investigado.

2.3 Aspectos estruturais em redes *feedforward*

Como mencionado anteriormente, RNAs compreendem estruturas em grafo cujos nós realizam processamento (não-)linear sobre suas entradas. Existem arquiteturas que permitem com que todos os nós se conectem a todos os outros e arquiteturas cujos nós estão dispostos em camadas. A Fig. 2.1(b) ilustra um arranjo especial das estruturas em camadas, caracterizado pela inexistência de ciclos conectando quaisquer nós. A esse tipo de arquitetura, dá-se o nome de redes *acíclicas*, cujas propriedades dinâmicas tendem a ser bem diferentes daquelas exibidas por topologias ditas *cíclicas*, dentre as quais se destacam as redes *recorrentes*⁽⁵⁾. Em verdade, a topologia esboçada na Fig. 2.1(b) é dita pertencer a uma subclasse das redes acíclicas, onde, além de não haver arcos conectando nós pertencentes à mesma camada, só é possível existirem conexões ligando nós em camadas adjacentes, ou seja, de um nó da camada l a nós da camada $l+1$ [318]. A essa classe especial de topologias, dá-se o nome de redes *feedforward*, cujas representantes mais notórias compreendem as redes Perceptron multicamadas (*Multilayer Perceptrons* — MLP)⁽⁶⁾, foco dos trabalhos envolvendo RNAs e comitês de máquina desta tese.

A arquitetura de uma rede MLP consiste de uma camada de entrada, uma ou mais camadas intermediárias e uma camada de saída. A camada de entrada, cujos neurônios $x_n, n = 1, \dots, D$ são chamados

⁽⁵⁾Redes recorrentes se caracterizam, principalmente, pela capacidade de emular memória e pela faculdade de representação/processamento de padrões espaço-temporais [143, 198].

⁽⁶⁾Redes neurais de base radial (RBFN) [77, 198] constituem outro subgrupo bem conhecido de redes *feedforward*.

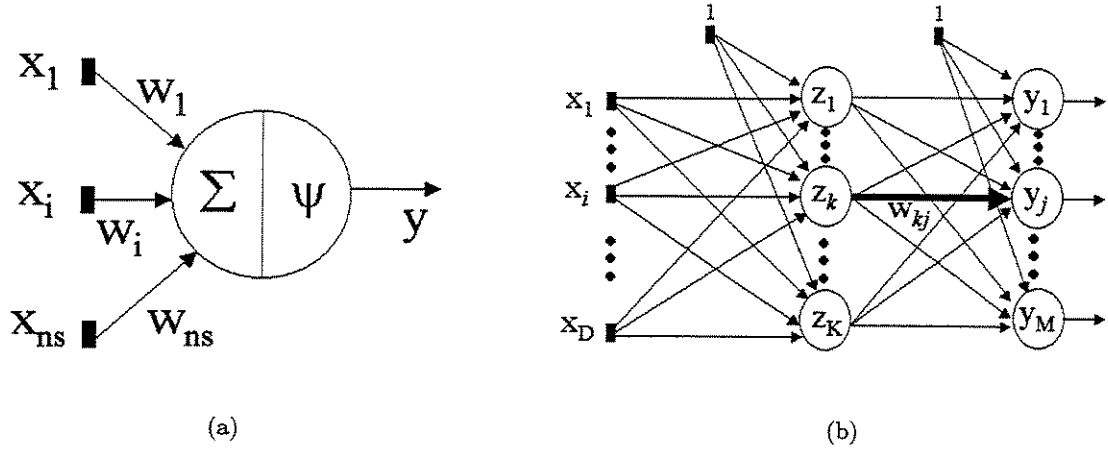


Fig. 2.1: (a) Modelo esquemático do neurônio MCP. (b) Arquitetura típica de uma rede *feedforward* com uma única camada escondida.

de unidades de entrada, difunde os valores (já pré-processados) dos atributos de cada amostra às demais camadas sem nenhuma modificação. As camadas intermediárias (ou escondidas) são responsáveis por extrair novas características e realizar o mapeamento dos dados para um espaço de maior dimensão (ver Seção 2.2). No esboço da Fig. 2.1(b), w_{kj} alude ao peso sináptico relativo à conexão ligando o nó z_k , da única camada intermediária existente, ao nó y_j , da camada de saída. Já as unidades indicadas por “1” sinalizam os neurônios de *bias* (polarização), que atuam simplesmente na geração dos *thresholds* dos neurônios escondidos e de saída. Por fim, a camada de saída transmite os sinais-resposta da rede associados ao exemplo de entrada corrente. Cada neurônio de saída, num problema de classificação, fica associado a uma classe em particular, produzindo a sua probabilidade *a posteriori*.

Formalmente, pode-se expressar o processamento *feedforward* da topologia da Fig. 2.1(b) via o seguinte equacionamento:

$$\begin{aligned} v_k &= \alpha_{0k} + \alpha_k^T \mathbf{x} \quad k = 1, \dots, K, \\ z_k &= \psi_k(v_k), \\ t_j &= \beta_{0j} + \beta_j^T \mathbf{z} \quad j = 1, \dots, M, \\ y_j &= g_j(t_j), \end{aligned} \tag{2.1}$$

onde $\{\alpha_k\}$ são coeficientes correspondentes aos pesos $w_{.k}$ entre a camada de entrada e a intermediária, $\{\beta_j\}$ são coeficientes correspondentes aos pesos $w_{.j}$ entre a camada intermediária e a de saída, $\mathbf{z} = (z_1, \dots, z_K)$ refere-se ao processamento (não-linear) realizado pelos nós da camada intermediária, e $\mathbf{y} = (y_1, \dots, y_M)$ diz respeito ao vetor de saídas de ativação (ver a seguir) dos neurônios da última camada.

O modelo típico de neurônio empregado em redes *feedforward* constitui uma simplificação do neurônio biológico e segue as diretrizes estabelecidas, na metade do século XX, pelos fisiologistas McCulloch e Pitts (MCP) [143, 198]. A sua descrição matemática resultou em um modelo com n_s terminais de entrada

$x_1, x_2, \dots, x_{n_s}$, representando os dendritos de um neurônio, e apenas um terminal de saída y , que representa o seu axônio. Para emular o comportamento das sinapses, os terminais de entrada do neurônio têm pesos acoplados $w_1, w_2, \dots, w_{n_s}$, cujos valores podem ser tanto negativos como positivos, dependendo das sinapses correspondentes serem *inibitórias* ou *excitatórias*. O efeito de uma sinapse particular i no neurônio pós-sináptico é simbolizado pelo produto $x_i \times w_i$, onde os pesos w_i atuam como “moduladores” dos sinais x_i . Uma descrição esquemática do modelo de neurônio MCP é esboçada na Fig. 2.1(a).

Um neurônio dispara sempre que a soma dos impulsos que ele receber ultrapassar o seu limiar (*threshold*) de excitação (este indicado, geralmente, por sua entrada de *bias*). O corpo do neurônio, por sua vez, é emulado por um mecanismo simples que faz a soma dos valores $x_i \times w_i$ recebidos por ele, ficando essa soma ponderada conhecida como *saída linear* ou *grau de ativação*. No modelo MCP, a *saída de ativação* do neurônio, y , é obtida mediante a aplicação de uma *função de ativação* sobre a saída linear, sendo que essa função pode assumir vários formatos, geralmente não-lineares para os nós da camada intermediária (ψ_k) e lineares para a camada de saída (g_j)⁽⁷⁾. Na descrição original do modelo MCP, a função de ativação foi projetada para ser a degrau (*step*); no entanto, qualquer uma das funções $\psi(v)$ listadas na Tab. 2.1 pode servir a esse propósito. Esse conjunto de funções contínuas, limitadas, monotonicamente não-decrescentes e diferenciáveis foi utilizado nos experimentos reportados no Cap. 7 e na Seção 8.1.

2.3.1 O problema do dimensionamento de RNAs

Uma questão deveras importante, no âmbito estrutural, é justamente a de se determinar, previamente, a topologia de uma rede *feedforward* mais adequada para se resolver uma dada tarefa. A capacidade de aproximação universal de RNAs *feedforward* é um resultado teórico de cunho meramente *existencial*, ou seja, não provê nenhuma orientação sobre como se deve obter a melhor instância de rede dadas as características de um problema-alvo. Fica somente assegurado que redes com uma única camada intermediária podem aproximar quaisquer mapeamentos, desde que o número de neurônios escondidos seja o suficiente [210].

Mais ainda, dadas duas arquiteturas A e B , caracterizadas pelo fato de que a primeira está totalmente contida na segunda, a ilação teórica de que o desempenho de B será *sempre* igual ou superior à de A é *não-verdadeira*, pois algoritmos de treinamento (ver próxima subseção) tendem a ser predispostos a dificuldades com mínimos locais. O dilema entre recorrer-se a redes pequenas ou a aquelas de maior escala origina-se no fato de que tanto uma classe quanto a outra exibem vantagens. Quando uma rede tem um número elevado de parâmetros livres, não somente o treinamento tende a ser mais rápido, como também os mínimos locais da superfície de erro tendem a ser mais facilmente evitados [72]. Existem estudos teóricos que comprovam que, quando o número de unidades escondidas se iguala ao número de amostras de treinamento (menos uma amostra), fica assegurado que a superfície de erro percorrida por um algoritmo de descida do

⁽⁷⁾Para problemas de regressão e previsão, a função identidade é a mais comumente adotada para a camada de saída. Já para problemas de classificação, a função *soft-max*, dada por $g_j(t_j) = \frac{\exp(t_j)}{\sum_{j^*=1}^M \exp(t_{j^*})}$, é a mais usada para fins de geração das probabilidades *a posteriori*.

Tab. 2.1: Descrição de funções (não-)lineares (e suas derivadas) tipicamente adotadas em redes MLP [143, 243, 318, 386].

	Função	Definição	Derivada
1	Linear	$\psi_l(v) = v$	$\psi'_l(v) = 1$
2	Tangente hiperbólica	$\psi_{th}(v) = \frac{1 - \exp(-2v)}{1 + \exp(-2v)}$	$\psi'_{th}(v) = 1 - (\psi_{th}(v))^2$
3	Sigmoidal bipolar	$\psi_{sbp}(v) = \frac{1 - \exp(-v)}{1 + \exp(-v)}$	$\psi'_{sbp}(v) = \frac{1 - (\psi_{sbp}(v))^2}{2}$
4	Sigmoidal binária	$\psi_{sbn}(v) = \frac{1}{1 + \exp(-v)}$	$\psi'_{sbn}(v) = \psi_{sbn}(v) [1 - \psi_{sbn}(v)]$
5	Seno	$\psi_{sen}(v) = \sin(v)$	$\psi'_{sen}(v) = \cos(v)$
6,7	Polinomial (ordem r)	$\psi_{poly}(v) = (v + \gamma)^r, \gamma = 1, r = \{2, 3\}$	$\psi'_{poly}(v) = r(v + \gamma)^{r-1}$
8	Degrau bipolar	$\psi_{step}(v) = \begin{cases} -1 & \text{se } v < 0, \\ 0 & \text{se } v = 0, \\ +1 & \text{se } v > 0 \end{cases}$	$\psi'_{step}(v) = \begin{cases} 0 & \text{se } v \neq 0, \\ 1 & \text{se } v = 0 \end{cases}$
9	Arcotangente	$\psi_{arctg}(v) = \frac{2 \arctan(v)}{\pi}$	$\psi'_{arctg}(v) = \frac{2}{\pi[1+v^2]}$
10	Absoluta inversa	$\psi_{absinv}(v) = \frac{v}{1+ v }$	$\psi'_{absinv}(v) = \frac{1}{(1+ v)^2}$
11	Logarítmica	$\psi_{log}(v) = \begin{cases} \log_{10}(1+v) & \text{se } v \geq 0, \\ \log_{10}(1-v) & \text{se } v < 0 \end{cases}$	$\psi'_{log}(v) = \begin{cases} \frac{1}{1+v} & \text{se } v \geq 0, \\ \frac{1}{1-v} & \text{se } v < 0 \end{cases}$
12	Cauchy	$\psi_{cau}(v) = \frac{1+v^2}{2\sigma}, \sigma \neq 0 (\text{variância})$	$\psi'_{cau}(v) = \frac{v}{\sigma}$
13	Multiquádrica	$\psi_{mq}(v) = \sqrt{\psi_{cau}(v)}, \sigma \neq 0$	$\psi'_{mq}(v) = \frac{v}{2\sigma\psi_{mq}(v)}$
14	Multiquádrica inversa	$\psi_{mqinv}(v) = \frac{1}{\psi_{mq}(v)}, \sigma \neq 0$	$\psi'_{mqinv}(v) = \frac{-v[\psi_{mqinv}(v)]^3}{2\sigma}$
15	Gaussiana	$\psi_{bell}(v) = \frac{\exp(-v^2)}{2\sigma}, \sigma \neq 0$	$\psi'_{bell}(v) = \frac{-v}{\sigma} \times \psi_{bell}(v)$
16	Wavelet	$\psi_{wav}(v) = -v \exp(\frac{-v^2}{2})$	$\psi'_{wav}(v) = -v\psi_{wav}(v) + \frac{\psi_{wav}(v)}{v}, v \neq 0$

gradiente *não* tem mínimos locais [506]. Redes sobredimensionadas podem também produzir regiões de decisão tão complexas quanto o problema requiera, além de se mostrarem mais robustas a falhas. Por outro lado, tanto trabalhos teóricos como experimentais demonstram que redes neurais com número restrito de parâmetros tendem a exibir melhor capacidade de generalização [42] (ver Seção 2.5), além de se mostrarem mais *parcimoniosas* quanto ao uso de recursos computacionais. Mais ainda, o conhecimento embutido em instâncias mais diminutas é presumivelmente mais fácil de ser interpretado, facilitando a extração de regras simples sobre como a rede opera.

2.3.2 Seleção de atributos

Uma outra questão de natureza estrutural diz respeito à escolha do número de entradas da rede, como meio de se definir a dimensão do problema. A qualidade do mapeamento realizado por uma rede *feedforward*, ou seja, das novas características extraídas por seus neurônios escondidos, vai depender diretamente da qualidade já embutida nos atributos que compõem as suas amostras de treinamento. O propósito das várias abordagens voltadas à *seleção de atributos* (ou *seleção de características*) [165, 270] é, portanto, o de se encontrar o menor subconjunto de atributos de entrada que ainda possa resultar em um desempenho preditivo satisfatório por parte do estimador gerado. Mais ainda, devido ao problema da *maldição da dimensionalidade* [130], é quase sempre benéfico se limitar o número de atributos de entrada de maneira a se ter ganhos computacionais. Dentre os métodos estatísticos mais populares, aplicados em reconhecimento de padrões, encontra-se a análise de componentes principais (*Principal Component Analysis* – PCA) [197], que segue a estratégia de reduzir a dimensão de entrada sem perder a informação intrínseca contida nos dados originais. Contudo, por se tratar de um técnica não-supervisionada, um dos problemas de PCA é o de não considerar a possível correlação existente entre dados de entrada e de saída. Isso abre caminho para se aplicar métodos de busca, tais como algoritmos evolutivos, para a seleção (ou mesmo geração) automática de atributos, levando-se em conta as nuances do problema em questão [259, 278, 279, 280].

Para problemas de predição de séries temporais, a escolha do número das variáveis independentes (*lagged variables*) assume uma importância ainda maior, segundo Zhang *et al.* [509], já que é somente por intermédio dessas variáveis que o modelo neural será capaz de identificar a estrutura de *autocorrelação* (linear/não-linear) existente nos dados. O fato é que atualmente *não* há nenhuma sistematização estatística padrão que sirva para calcular *exatamente* a ordem do modelo (número de *lags*), atendendo tanto às propriedades exibidas por uma série temporal qualquer como aos possíveis problemas de multicolinearidades existentes [387, 509]. A despeito da existência de um sem-número de testes estatísticos para detectar dependências não-lineares em séries temporais, a eficácia da maioria desses testes tende a ser fortemente dependente do tipo de modelo empregado (geralmente paramétrico) e nenhum desses testes tem se mostrado consistentemente superior em todas as situações.

Um artifício estatístico para se determinar a ordem k de um modelo auto-regressivo (AR) relativo a uma série $s = \{s_1, s_2, \dots, s_t, \dots, s_N\}$ qualquer é o de se estimar, iterativamente, os *coeficientes*

de *autocorrelação parcial* da série. Para um modelo linear genérico de ordem p ($p = 1, 2, \dots$), dado por $s_t = \phi_1 s_{t-1} + \phi_2 s_{t-2} + \dots + \phi_p s_{t-p} + a_t$, tal coeficiente de autocorrelação parcial, denotado aqui por $\hat{\phi}_{pp}$, será o mesmo que $\hat{\phi}_p$, ou seja, o último coeficiente do modelo. Plotando-se os vários coeficientes parciais, obtém-se a *função de autocorrelação parcial* da série [36]. Daí, avalia-se o gráfico resultante e verifica-se, aproximadamente, qual o último índice q para o qual a condição $|\hat{\phi}_{qq}| > \Phi$ seja ainda satisfeita, onde $\Phi = 2DP[\hat{\phi}_{qq}] \equiv 2/\sqrt{N}$ (intervalo de confiança)⁽⁸⁾. Tal índice sugerirá a ordem do modelo auto-regressivo que melhor se ajusta aos dados. Esse artifício foi usado, por exemplo, para se determinar a janela de previsão para a série financeira *forex* [16] estudada no artigo descrito na Seção 8.3⁽⁹⁾. A Fig. 2.2 ilustra (i) o formato da série original; (ii) a série transformada, obtida mediante um processo de suavização que leva em consideração o número de *lags*; (iii) a série normalizada em $[-1, 1]$; e (iv) o comportamento oscilatório da função de autocorrelação parcial associada a essa série. Esse comportamento é típico de séries passíveis de serem ajustadas por modelos auto-regressivos [36].

2.3.3 Capacidade de uma rede MLP e configuração de seus parâmetros

Outro fato relevante é que tanto o número de neurônios escondidos, como o número de amostras de treinamento, como os valores iniciais dos pesos têm influência (direta ou indireta) na *capacidade* a ser exibida pelo conjunto de hiperplanos finais gerados pela rede. O termo “capacidade” aqui faz alusão ao Teorema de separabilidade de padrões comentado anteriormente. Nesse contexto, Cover [97] define a capacidade de um hiperplano como sendo o número de *dicotomias*⁽¹⁰⁾ que ele consegue realizar. De um modo formal, tendo-se N pontos em posição geral⁽¹¹⁾ num espaço euclidiano de dimensão D , então o número de dicotomias linearmente e homogeneamente separáveis, $C(N, D)$, é dado por [386]

$$C(N, D) = \begin{cases} 2^N & \text{se } N \leq D + 1 \\ 2 \sum_{k=0}^D \binom{N-1}{k} & \text{se } N > D + 1 \end{cases} \quad (2.2)$$

Dessa expressão, pode-se depreender que todas as dicotomias sobre $N \leq D + 1$ pontos em posição geral são linearmente separáveis em D dimensões. Por exemplo, todas as rotulagens de três pontos em duas dimensões são linearmente separáveis. Já para $N > D + 1$, somente $C(N, D)$ das 2^N possíveis dicotomias são linearmente separáveis.

Esse resultado matemático torna-se importante pois viabiliza a análise do grau de generalização a ser exibido por qualquer máquina de aprendizado fundamentada no uso de hiperplanos de separação (tais como RNAs e máquinas de vetores-suporte). Mais ainda, ele permite que sejam tiradas algumas conclusões

⁽⁸⁾ “DP” significa desvio-padrão.

⁽⁹⁾ Para as outras duas séries temporais não-lineares estudadas no referido artigo, foram usados números de atrasos comumente adotados na literatura [208, 508].

⁽¹⁰⁾ Uma dicotomia compreende uma partição de quaisquer N pontos em apenas duas classes.

⁽¹¹⁾ Um conjunto de vetores D -dimensionais é dito estar em *posição geral* sempre que todos os subconjuntos possíveis de $D + 1$ ou menos vetores sejam linearmente independentes. Se os N pontos *não* estão em posição geral, então $C(N, D)$ tende a ser bem menor do que o resultado mostrado em (2.2).

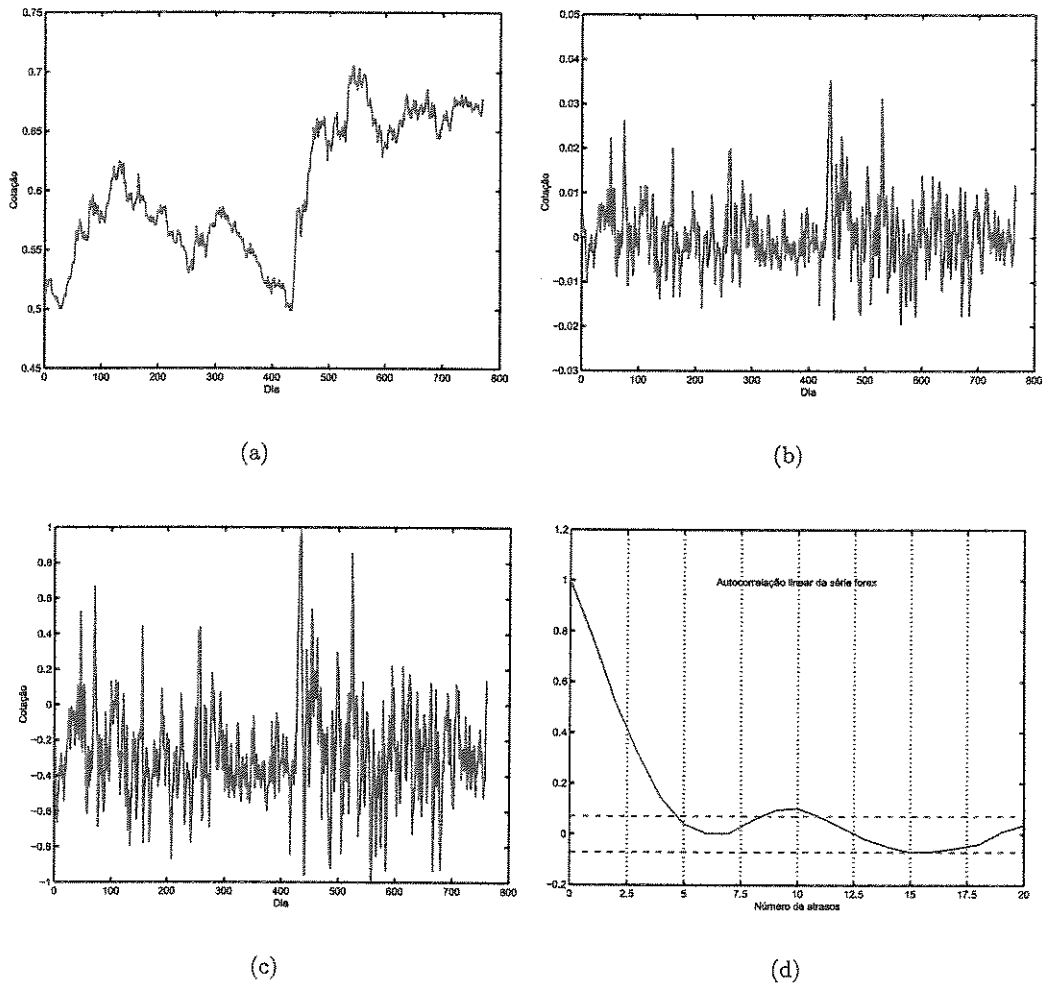


Fig. 2.2: (a) Série *forex* original [16]. (b) Série transformada. (c) Série normalizada. (d) Autocorrelação linear da série com indicação do limiar de definição do número de *lags* (no caso, $\Phi \approx 0,07$ e $q = 4$).

acerca do número de amostras de treinamento (N) a serem usadas em função da dimensionalidade (D) do problema. Por exemplo, em média, uma única unidade linear com D pesos e um *threshold* pode classificar corretamente até $2(D + 1)$ padrões binários aleatórios, sendo que a partir daí a sua probabilidade de erro tende a ultrapassar 50% [97]. Para valores elevados de N , somente uma minúscula fração dos possíveis mapeamentos são linearmente separáveis. Em termos de generalização, isso significa que um neurônio com *threshold* linear e D entradas pode implementar uma dicotomia sobre $D + 1$ ou menos padrões (em posição geral) e, quando D é elevado, esse neurônio passa a ter uma alta probabilidade de aprender qualquer dicotomia aleatória sobre até $2D$ padrões. Já para $N < 2D$, o grau de generalização da unidade passa a ser irrisório já que há uma alta probabilidade de que os pontos de treinamento sejam por si só linearmente dependentes — a rede formada por essas unidades seria irrestrita e a solução que ela guardaria provavelmente não iria generalizar sobre novos padrões. Em regressão linear, uma heurística comumente admitida, nesse sentido, é a de se requerir $N \geq 3D$ ou mais exemplos de treinamento para se evitar *overfitting* [386].

Baum e Haussler [42] trouxeram novas contribuições para esse exercício teórico ao estudarem a relação entre o número de exemplos necessários para se treinar uma rede com V neurônios e W pesos e o nível de precisão α ($0 < \alpha \leq 1$) requerido sobre o conjunto de teste. O resultado obtido é independente do algoritmo de aprendizado empregado e é calcado numa estimativa da dimensão VC da rede (ver Cap. 3). De acordo com Baum e Haussler, dado que uma rede pode ser treinada com

$$N \geq O\left(\frac{W}{1-\alpha} \log_2\left(\frac{V}{1-\alpha}\right)\right)$$

amostras aleatórias retiradas do mapeamento desejado, de tal forma que pelo menos $(100 \times \frac{1+\alpha}{2})\%$ dessas amostras sejam corretamente classificadas, então essa rede irá quase que certamente classificar corretamente $(100 \times \alpha)\%$ das amostras de teste retiradas da mesma distribuição (para $0 < 1-\alpha \leq 1/8$) [42, 318, 386]. Uma fórmula derivada dessa estabelece, de forma aproximada, o número de neurônios escondidos V_h em função de N , α e do número de entradas D e saídas M (supondo um problema de classificação com M classes):

$$V_h \leq \frac{N \times \alpha}{D + M}. \quad (2.3)$$

Essa relação teórica foi usada no projeto do Cap. 7 como um artifício para se estabelecer um limitante superior para o número de neurônios escondidos das diversas RNAs-componentes que irão integrar as *ensembles* coevolucionárias.

No que tange ao processo de inicialização dos pesos em redes *feedforward*, este geralmente se dá mediante a escolha de valores aleatórios pequenos (com distribuição uniforme ou gaussiana) dentro de uma faixa preestabelecida. Aleatoriedade aqui é necessária para efeito de se quebrar a simetria da rede, ou seja, para se garantir que os nós computem funções diferentes. Caso todos os nós de uma dada camada l tivessem vetores de pesos idênticos, então todos os nós responderiam de forma idêntica e l poderia ser substituída por um único neurônio. Já a imposição de se gerar valores pequenos decorre da necessidade de se evitar a saturação imediata da rede, visto que pesos com elevada magnitude podem amplificar as entradas de tal modo que os graus de ativação dos neurônios atinjam regiões de saturação das suas não-linearidades. Com

isso, o processo de treinamento tenderá a ser mui lento, haja vista que as derivadas dos erros em relação aos pesos tenderão a ser demasiadamente pequenas. Uma heurística comumente usada estabelece que a faixa de valores iniciais para o vetor de pesos de um neurônio particular seja dada por $[-A/\sqrt{n_s}, A/\sqrt{n_s}]$, onde A é um número qualquer entre 2 e 3 e n_s denota o número de sinapses (entradas) incidindo sobre esse neurônio [386].

2.4 Treinamento de redes neurais com camadas múltiplas

Com relação à análise de complexidade do processo de aprendizado em redes neurais, Judd [437] demonstrou que o *problema de carregamento* (*loading problem*) de uma dada instância de RNA é, em geral, NP-completo, ou seja, o tempo de treinamento cresce exponencialmente com o tamanho da rede, no pior caso. Blum e Rivest [55] foram mais adiante, provando que o problema já é NP-completo para redes contendo tão-somente três neurônios. O problema de carregamento pode ser definido como sendo a tarefa de se configurar os valores dos pesos de uma RNA de tal sorte que ela possa “imitar” o “comportamento” dos dados de entrada. De acordo com Kwok e Yeung [271], o problema pode ser especificado mediante seus requisitos de entrada e saída:

Entrada Uma arquitetura de rede e um conjunto de treinamento.

Saída A determinação dos pesos da rede de sorte que cada exemplo no conjunto de treinamento seja mapeado à saída desejada, ou uma mensagem de que isso não é possível.

Entretanto, esse inconveniente “teórico” não é tão severo na prática, já que, como também ficou demonstrado por Blum e Rivest, um problema difícil de ser “carregado” por uma instância de RNA pode não ser para outra. Além disso, existem outros resultados teóricos que sinalizam que algoritmos com capacidade de adicionar novos neurônios e pesos ao longo do treinamento da rede podem resolver, em tempo polinomial, qualquer problema de aprendizado que tenha sido resolvido, também em tempo polinomial, por outro algoritmo (seqüencial) qualquer. Essa capacidade de “aprendizado universal” abre uma prerrogativa interessante para o estudo de métodos construtivos, tema da Seção 2.5.

2.4.1 Tratamento e particionamento dos dados

Antes de discorrer sobre como se dá o treinamento em redes MLP, cabe aqui discutir, brevemente, alguns aspectos relativos ao pré-processamento e particionamento dos dados. Tal discussão é extensiva às demais abordagens de Aprendizado de Máquina, analisadas nos Caps. 3 e 4.

2.4.1.1 Pré-processamento dos dados

Primeiramente, é interessante salientar a necessidade de uma etapa de *normalização* dos atributos (de entrada e saída) que compõem as amostras, antes que estas sejam submetidas ao indutor. Isso é justificável, particularmente em redes neurais, a fim de se evitar dois problemas: (1) elevadas diferenças entre os intervalos (domínios) das entradas da rede podem diminuir a importância de alguns atributos-chave e ressaltar a de outros não tão relevantes, levando a rede a produzir um mapeamento distorcido; e (2) sinais com magnitude elevada podem ocasionar uma saturação rápida nas funções de ativação dos neurônios, gerando *parálisia* durante o treinamento da rede (*network paralysis*). Geralmente, os dados são re-escalados entre $[0, 1]$ (intervalo binário) ou $[-1, 1]$ (intervalo bipolar), de tal maneira a exibirem média zero e desvio-padrão unitário⁽¹²⁾. De todo modo, independentemente de como é feita a normalização, é necessário que, quando da fase de análise de desempenho do estimador gerado, seja mencionado se a medida de erro ou precisão foi calculada sobre os dados originais ou normalizados. Isso não tem muito impacto em se tratando de problemas de classificação, mas assume um papel proeminente em problemas de previsão. Nesse sentido, é bom frisar, de antemão, que os resultados reportados na Seção 8.3 foram obtidos sobre as séries *normalizadas*.

Dentre os quatro tipos de normalização de dados apresentados no Apêndice A, aplica-se, tipicamente, o método de *normalização intra-canal* (não-isotrópico) em tarefas de classificação, enquanto, em problemas de predição de séries univariadas, o método de *normalização externa* é o mais factível de ser implementado. Tanto em uma situação como na outra, tipicamente, os atributos são linearmente transformados conforme a regra $x_t = \frac{(b-a)(x_o - x_{min})}{x_{max} - x_{min}} + a$, onde x_o e x_t referem-se às séries original e transformada, x_{min} e x_{max} são os valores mínimo e máximo da série original, e a e b definem o intervalo de normalização ($[0, 1]$ ou $[-1, 1]$).

Além da etapa de normalização, muitas vezes é necessário lidar com dados faltantes e com atributos não-numéricos (discretos). No primeiro caso, a saída comumente adotada é a de substituir o atributo faltante em uma dada amostra pelo valor médio referente àquele atributo sobre todo o conjunto de amostras. No segundo caso, geralmente se atribui a cada um dos possíveis valores categóricos de um atributo uma nova dimensão (binária ou bipolar) de entrada à rede. Por exemplo, seja λ um atributo não-numérico assumindo quatro possíveis valores, $\lambda_1, \dots, \lambda_4$, então, para se indicar o fato de que uma amostra qualquer possui o atributo λ com valor λ_4 , basta apresentar, juntamente com os outros atributos (numéricos ou discretos) dessa amostra, o vetor $(-1, -1, -1, 1)$ à rede (considerando codificação bipolar). Uma desvantagem dessa estratégia está na possibilidade de “explosão” do número de entradas da rede, para quando o número de atributos discretos ou número de valores que estes assumam sejam bem elevados.

No que concerne unicamente à tarefa de previsão de eventos temporais, é comum serem realizados dois outros processos de tratamento de dados. O primeiro diz respeito à eliminação de *tendências* (inclinações do nível da série) com o intuito de restabelecer a *estacionariedade* da série, isto é, garantir que ela se desenvolva aleatoriamente no tempo ao redor de uma média constante, exibindo alguma forma de equilíbrio estável. A transformação mais comumente acolhida é a de se tomar *diferenças* sucessivas da série original, ou seja,

⁽¹²⁾Nos experimentos apresentados ao longo da tese, optou-se pelo segundo tipo de intervalo de escalamento, salvo indicação em contrário.

$\Delta^n s_t = \Delta[\Delta^{n-1} s_t] = \underbrace{\Delta \cdots \Delta}_{n-1}[\Delta s_t]$, onde $\Delta s_t = s_t - s_{t-1}$. O segundo processo de transformação dos dados ocupa-se com a presença de *outliers*, observações discrepantes que são, muitas vezes, fruto de erros de medição ou discretização. Tais valores aberrantes são normalmente considerados como ruído, devendo ser eliminados do conjunto de dados.

Finalmente, é interessante comentar um pouco sobre como se deve interpretar as saídas produzidas por uma instância de RNA. Em tarefas de regressão ou previsão, o número de saídas está geralmente atrelado ao número de variáveis dependentes do mapeamento que se quer modelar, sendo que a saída correspondente a uma dada variável pode estar normalizada (em $[0, 1]$ ou $[-1, 1]$) ou pode assumir um dentre vários valores reais de uma faixa de operação. Já para problemas de classificação, normalmente faz-se uso do formato de codificação conhecido como “1-of- M ”, em que, para cada um dos M possíveis rótulos de classe, fica alocada uma dada unidade de saída. Aquela unidade com maior saída de ativação indicará o rótulo da classe vencedora [318].

2.4.1.2 Particionamento dos dados

Concernente ao processo de particionamento dos dados, a estratégia mais comumente admitida, como forma de avaliar o erro de generalização do estimador, é a de reservar parte aleatória dos dados para fins de teste de desempenho. Esse tipo de particionamento caracteriza o método de validação *holdout*, que apresenta como desvantagem a redução da quantidade de dados disponíveis para fins de treinamento. Essa estratégia de particionamento foi utilizada nos experimentos discutidos nas Seções 8.1 e 8.3.

Como alternativa ao método anterior, apresenta-se a estratégia de *validação cruzada* (*cross-validation*). Esta permite que todas as amostras disponíveis sejam usadas pelo indutor na geração do melhor estimador, ao preço de incorrer em vários ciclos de treinamento, aumentando-se, conseqüentemente, o tempo computacional. De maneira sucinta, na estratégia de *validação cruzada com k partições*, dividem-se, aleatoriamente, os dados em k subconjuntos de tamanho (aproximadamente) igual. Daí, executa-se k vezes o processo de treinamento, deixando de lado, em cada uma das iterações, uma partição específica, que atuará como conjunto de teste e avaliará o desempenho de generalização do estimador. Essa estratégia de particionamento foi adotada nos experimentos reportados no Cap. 7 e na Seção 8.2.

Como se vê, a metodologia de validação cruzada é bem diferente da de *holdout*, mesmo quando esta apresenta um terceiro conjunto para validação do modelo ao longo do treinamento (técnica de *early stopping* [386]). O Apêndice A traz mais considerações acerca dessas abordagens de particionamento dos dados, tendo como mira a avaliação de desempenho de estimadores.

1. Apresenta-se uma amostra de treinamento, que é propagada pelos nós internos até serem geradas as saídas da rede (passo *feedforward*);
2. Comparam-se as saídas obtidas com aquelas desejadas e calcula-se o erro de acordo com a função-custo;
3. Calculam-se as derivadas do erro com relação aos pesos;
4. Ajustam-se os pesos de tal forma a minimizar o erro;
5. Repetem-se os passos anteriores até que um critério de parada (número de iterações ou erro dentro de uma faixa aceitável) seja atingido.

Fig. 2.3: Passos da metodologia de retropropagação do erro [198, 243, 386].

2.4.2 Metodologia de retropropagação do erro

Em se tratando de uma RNA *feedforward*, cuja arquitetura já tenha sido definida, o processo de aprendizado se traduz em um problema de minimização não-linear e irrestrito dirigido à adaptação contínua dos parâmetros livres do modelo em consonância com algum (ou mais de um) critério de desempenho, sendo este o responsável por avaliar a qualidade da solução a ser gerada — tipicamente, uma medida de erro, distância ou similaridade. Dentre os possíveis métodos aplicáveis nesse contexto, o algoritmo supervisionado *Backpropagation* (de retropropagação) [76, 437] pode ser considerado como, de longe, o mais estudado, implementado e estendido.

Em verdade, o termo *Backpropagation* compreende uma metodologia envolvendo dois passos distintos mas interdependentes [386]. O primeiro passo descreve um método para calcular as derivadas do erro de treinamento com respeito aos pesos mediante o recurso da “derivação pela regra da cadeia”. O segundo passo refere-se ao algoritmo de treinamento propriamente dito, que usa as derivadas geradas no passo anterior para minimizar o erro, sendo esse processo de otimização equivalente ao método da *descida do gradiente* (*gradient descent*). O grande avanço promovido aqui foi o de demonstrar que redes contendo múltiplas funções não-lineares, dispostas em camadas e com propriedade de *diferenciabilidade*, poderiam realizar cálculos bem avançados, exibindo capacidade de aprendizado e generalização. Os passos principais por trás dessa metodologia são exibidos na Fig. 2.3 e descritos mais formalmente a seguir.

Nesse procedimento, a função de erro mede o custo das diferenças entre as saídas da rede e os valores desejados. Existem várias métricas que podem ser empregadas para esse fim, mas a escolha geralmente é feita em função das características da superfície de erro associada e/ou em função dos requisitos da aplicação. Para problemas de regressão e classificação, as funções-custo mais comumente utilizadas são a *Soma dos*

Erros Quadráticos (SEQ), dada por

$$E_{SEQ} = \sum_i E_i = \sum_i \left[\frac{1}{2} \sum_j (d_{ij} - y_{ij})^2 \right] = \frac{1}{2} \sum_i \sum_j (d_{ij} - y_{ij})^2, \quad (2.4)$$

e o *Erro Quadrático Médio* (EQM), denotado por

$$E_{EQM} = \frac{1}{NM} E_{SEQ} = \frac{1}{2NM} \sum_i \sum_j (d_{ij} - y_{ij})^2, \quad (2.5)$$

onde i faz a varredura sobre as amostras do conjunto de treinamento, j é o índice dos neurônios de saída da rede, e d_{ij} e y_{ij} são, respectivamente, a saída desejada e a obtida para a j -ésima saída em relação ao i -ésimo exemplo. O fator “1/2” é usado somente para facilitar o cálculo das derivadas no terceiro passo. Vantagens dessas métricas incluem a facilidade de diferenciação e o fato de que o custo do erro passa a depender diretamente da sua magnitude. Para problemas de classificação, é comum lançar-se mão também da *Função de Entropia Cruzada* (*cross-entropy*), definida como

$$E_{cruz} = \sum_i \sum_j d_{ij} \ln(y_{ij}) + (1 - d_{ij}) \ln(1 - y_{ij}), \quad (2.6)$$

onde, nesse caso, a saída y_{ij} é interpretada como sendo a probabilidade do exemplo i pertencer à classe j e $d_{ij} \in \{0, 1\}$.

Já para problemas de previsão, é comum fazer uso também de outras métricas que tentam capturar outros aspectos estatísticos relacionados à dinâmica temporal das séries. Armstrong e Collopy [21] e Refenes [387] provêem uma discussão qualitativa de várias dessas métricas para análise das previsões geradas. Por exemplo, o *Coefficiente de informação* (ou Coeficiente de Theil), T_r , oferece uma boa medida de cotejo entre o desempenho do preditor gerado e o desempenho de um preditor trivial (*naïve predictor*) baseado na hipótese do caminho aleatório (*random walk hypothesis*)⁽¹³⁾. Qualquer preditor cujo valor de T_r é *menor* que a unidade estará fazendo mais do que repetir o valor da observação atual da série. Já a *Reversão média* compara o preditor gerado com outro preditor trivial, este baseado na média histórica da série. Tal estatística T_μ é, às vezes, chamada de *medida da variância relativa média* já que ela calcula o erro quadrático médio normalizado pela variância da série. A Tabela 2.2 traz a definição matemática de algumas dessas funções-custo, sendo estas adotadas nos experimentos conduzidos na Seção 8.3.

Além de termos primários que medem diretamente os erros de aproximação, a função-custo é frequentemente incrementada com outras parcelas refletindo objetivos e preferências que não são mensuráveis diretamente pelas diferenças entre saídas desejadas e obtidas. Termos de penalização, por exemplo, são normalmente utilizados a fim de se gerar superfícies de erro mais suaves (via *regularização* dos valores assumidos pelos pesos) ou para fins de otimização estrutural (*pruning* — ver Seção 2.5). Dentro dessa linha, é comum também utilizar-se de outros critérios, advindos da área de Teoria da Informação, com o intuito de se gerar soluções mais parcimoniosas, de acordo com o *Princípio do menor tamanho de descrição* (*Minimum Description Length* — MDL) [246]. Critérios de informação de Akaike, de Schwarz-Rissanem e de

⁽¹³⁾Tal hipótese afirma que a melhor estimativa para o próximo passo é o valor corrente da série.

Tab. 2.2: Métricas de desempenho em problemas de previsão de séries temporais ($e_t = d_t - y_t$ e $\mu_s = \frac{\sum s_t}{N}$).

Métrica	Equação
<i>Erro Absoluto Médio (EAM)</i>	$E_{EAM} = \frac{\sum e_t }{N}$
<i>Raiz Quadrada do Erro Quadrático Médio (RQ_EQM)</i>	$E_{RQ\_EQM} = \frac{\sqrt{\sum (e_t)^2}}{N}$
<i>Raiz Quadrada Normalizada do Erro Quadrático Médio (RQN_EQM)</i>	$E_{RQN\_EQM} = \sqrt{\sum \frac{(e_t)^2}{(\mu_d - y_t)^2}}$
<i>Erro Percentual Absoluto Médio (EPAM)</i>	$E_{EPAM} = \frac{1}{N} \sum \left \frac{e_t}{y_t} \right (100)$
<i>Coefficiente de Theil (T_r)</i>	$T_r = \sqrt{\frac{\sum (e_t)^2}{\sum (d_t - d_{t-1})^2}}$
<i>Reversão média (T_μ)</i>	$T_\mu = \sqrt{\frac{\sum (e_t)^2}{\sum (\mu_d - d_{t+1})^2}}$

Banshali-Downham [197, 387, 505] compreendem variações sobre métricas “puras” (como o EQM) destinadas a dimensionar o grau de complexidade do estimador.

Na etapa de *feedforward* (passo 1. da Fig. 2.3), cada u -ésimo nó da rede, quer seja escondido, quer seja de saída, computa uma soma ponderada de suas entradas (i. e., grau de ativação), $a_u = \sum_{m < u} w_{mu} y_m$ — $m < u$ indicando que m é um nó da camada precedente à de u —, a ser usada, posteriormente, como meio de gerar a sua saída de ativação $y_u = f_u(a_u)$. Caso u represente um neurônio intermediário k , então a_u e f_u equivalem a v_k e ψ_k , respectivamente, ao passo que, quando u faz alusão a um neurônio de saída j , então a_u e f_u passam a equivaler a t_j e g_j , respectivamente — vide Eq. (2.1).

Uma vez calculadas as saídas e, posteriormente, os erros de estimação associados, a próxima etapa (ver Fig. 2.3) do *Backpropagation* passa a ser a de se calcular as derivadas. Recorrendo-se à função-custo E_{SEQ} (Eq. (2.4))⁽¹⁴⁾, e como o erro total é a soma dos erros individuais para cada amostra, isto é, $E_{SEQ} = \sum_i E_i$, pode-se escrever que

$$\frac{\partial E}{\partial w_{mu}} = \sum_i \frac{\partial E_i}{\partial w_{mu}}.$$

O fundamental aqui passa a ser descobrir como se decompõe essa expressão de maneira eficiente computacionalmente. Isso pode ser feito, como mencionado anteriormente, apelando-se à “artimanha” da regra da cadeia. Supondo, num primeiro momento, que o nó u seja de saída, a aplicação da regra da cadeia dá origem a:

$$\frac{\partial E_i}{\partial w_{mu}} = \sum_{j=u} \frac{\partial E_i}{\partial t_{ij}} \frac{\partial t_{ij}}{\partial w_{mu}}, \quad (2.7)$$

onde j continua sendo índice sobre todos os nós de saída e t_{ij} denota o grau de ativação do neurônio j quando da propagação do i -ésimo caso de treinamento. Para todo nó j , pode-se definir o parâmetro $\delta_j \equiv \partial E_i / \partial t_{ij}$,

⁽¹⁴⁾Outras funções poderiam ser aplicadas nesse desenvolvimento matemático.

o qual mede a parcela de contribuição de t_{ij} para o erro total da rede, como sendo

$$\delta_j = \frac{\partial E_i}{\partial y_{ij}} \frac{\partial y_{ij}}{\partial t_{ij}}.$$

Por se tratar de um neurônio de saída, o valor de δ_j pode ser obtido diretamente, fazendo $\delta_j = -(d_{ij} - y_{ij})g'_j$, uma vez que $\partial E_i / \partial y_{ij} = -(d_{ij} - y_{ij})$ e a segunda derivada, $\partial y_{ij} / \partial t_{ij} = g'(t_{ij}) \equiv g'_j$, diz respeito à inclinação da função de ativação no valor de seu grau de ativação atual.

Por outro lado, quando o nó u não é de saída, a aplicação da regra da cadeia passa a produzir:

$$\frac{\partial E_i}{\partial w_{mu}} = \sum_k \frac{\partial E_i}{\partial v_{ik}} \frac{\partial v_{ik}}{\partial w_{mu}}, \quad (2.8)$$

sendo que, agora, não existe a restrição $k = u$ sob o somatório. Da mesma forma que para os neurônios de saída, pode-se também definir valores $\delta_k \equiv \partial E_i / \partial v_{ik}$ para os neurônios pertencentes a uma ou mais camadas intermediárias da rede. Todavia, tais valores só podem ser obtidos de forma indireta, posto que a contribuição desses nós para o erro total da rede relativo ao exemplo corrente se dá somente mediante os sinais que eles propagam para cada nó l da camada sobrejacente (podendo ser esta a de saída). Essa observação pode ser formalizada como:

$$\delta_k = \sum_l \frac{\partial E_i}{\partial v_{il}} \frac{\partial v_{il}}{\partial v_{ik}}.$$

Pode-se notar que o primeiro fator dentro do somatório se traduz no valor δ_l do l -ésimo nó, o que leva a redefinir δ_k como sendo

$$\delta_k = \sum_l \delta_l \frac{\partial v_{il}}{\partial v_{ik}}.$$

Já o segundo fator pode ser obtido atentando-se para o fato de que, se o nó k se conecta diretamente ao nó l , então $\partial v_{il} / \partial v_{ik} = \psi'_k \times w_{kl}$; caso isso não ocorra, esse fator é nulo. Disso, conclui-se que

$$\delta_k = \psi'_k \sum_l w_{kl} \delta_l.$$

Ou seja, δ_k compreende uma soma ponderada sobre os valores δ_l dos nós l com os quais o nó k se conecta. Porque δ_l deve ser obtido antes de δ_k , o processo de cálculo das derivadas deve começar de trás para frente, originando-se, daí, o nome da metodologia. Ademais, como δ_k depende dos valores ψ'_k , tal fato justifica a necessidade de se empregar não-linearidades diferenciáveis (Tab. 2.1), tais como a função sigmoideal, cuja derivada pode ser escrita como uma função simples da saída de ativação do nó, ou seja, $\psi'_{sbn}(v_{ik}) = y_{ik}(1 - y_{ik})$, onde $y_{ik} = \psi_{sbn}(v_{ik})$.

Obtendo-se os “deltas” para todos os nós (excetuando-se os de entrada), torna-se trivial calcular as derivadas $\partial E_i / \partial \mathbf{w}$ com relação ao vetor de pesos da rede. Levando em conta que t_{ij} é uma soma linear simples das entradas do neurônio de saída j , então o segundo fator presente na Eq. (2.7) pode ser denotado como $\partial t_{ij} / \partial w_{mu} = y_m$, onde y_m é a saída do m -ésimo nó. O mesmo resultado se aplica para a Eq. (2.8). Por

consequente, a derivada do erro total relativo à i -ésima amostra com respeito ao peso w_{mu} pode ser escrita como $\partial E_i / \partial w_{mu} = \delta_u y_m$, isto é, o produto entre o valor de correção do u -ésimo nó, seja escondido ou de saída, e a saída não-linear do m -ésimo nó.

De posse das derivadas dos erros (passo 3 da Fig. 2.3), passa-se à aplicação do método de otimização para o ajuste dos parâmetros da rede. Por definição, o gradiente do erro E aponta para a direção de máximo incremento de E , daí, de modo a minimizar E , os pesos passam a ser ajustados no sentido inverso, levando a

$$\Delta w_{mu} = -\eta \frac{\partial E}{\partial w_{mu}},$$

onde a *taxa de aprendizado* η é uma constante de valor pequeno ($0,1 < \eta < 1$), geralmente configurada previamente e responsável pela magnitude do ajuste aos pesos.

2.4.3 Modos de aprendizado: incremental ou em batelada

Um aspecto que precisa ser observado quando da aplicação da metodologia de retropropagação do erro é a frequência com que os pesos são atualizados. Quanto a esse aspecto, dois *modos de aprendizado* são comumente empregados, *incremental* (*online*) e *em batelada* (*batch*).

Na abordagem incremental (padrão-a-padrão), os pesos da rede são atualizados após a apresentação de cada caso de treinamento. Geralmente, as amostras são apresentadas em uma ordem aleatória, que deve mudar constantemente, com o propósito de se evitar efeitos cíclicos. Essa abordagem é considerada estável sempre que a taxa de aprendizado é pequena. Por outro lado, quando taxas elevadas são utilizadas, a rede tende a tornar-se bastante instável. Uma vantagem desse modo de aprendizado é que ele tende a ser mais rápido, principalmente se o conjunto de treinamento for grande e redundante, haja vista requerer menos memória. Uma possível desvantagem — pelo menos, sob a perspectiva de análise — é que, nesse modo, o algoritmo *Backpropagation* deixa de ser uma aproximação do método da descida do gradiente, uma vez que as derivadas do erro relativo a um simples padrão passam a ser vistas como estimativas ruidosas do verdadeiro gradiente [386].

Na abordagem em batelada (ou lote), por sua vez, os pesos são atualizados após todos os casos terem sido apresentados, o que corresponde a uma época de treinamento. Nesse caso, todos os desvios $\frac{\partial E_i}{\partial w}$ são somados de modo a se gerar uma única correção para os pesos. Esse modo de aprendizado tende a ser mais estável do que o anterior, mas pode ser lento caso o conjunto de treinamento seja grande e redundante. Essa abordagem apresenta uma estimativa mais precisa do vetor gradiente, ao custo da necessidade de mais memória. Todos os experimentos com RNAs reportados nesta tese (inclusive aqueles envolvendo comitês de máquina) adotam o modo de aprendizado em batelada.

2.4.4 Retropropagação do erro: Dificuldades e deficiências

A metodologia de retropropagação do erro apresenta uma série de pontos negativos, levando-se em consideração aspectos de eficiência e eficácia. Tais deficiências incluem (i) a necessidade de se configurar previamente parâmetros de controle cujas ações, em conjunto, podem se mostrar contraproducentes; (ii) a dificuldade de se obter soluções estáveis em poucas iterações (problema de convergência lenta); (iii) a idéia de se corrigir os pesos caminhando de trás para frente (*backwards*) não parece ser biologicamente plausível; e (iv) a tendência de se gerar sobreajuste (*overfitting*) aos dados. Mais ainda, como o processo de correção dos pesos é feito de forma acoplada (ou seja, todos os rótulos são alterados ao mesmo tempo), independentemente se em batelada ou de forma incremental, o algoritmo *Backpropagation* tem propensões a apresentar sinais de *interferência catastrófica* [314]. Resumidamente, interferência catastrófica ocorre quando novos padrões apreendidos pela rede apagam subitamente e completamente — i. e., “catastroficamente” — a memória relativa a padrões anteriores, caracterizando-se como um problema típico do dilema “estabilidade-plasticidade” discutido na Seção 2.2.

O desempenho computacional do método é afetado pela necessidade de se realizar cálculos extensos: a quantidade de cálculos é da ordem do número de pesos e o tempo requerido para se calcular as derivadas do erro e se atualizar os pesos, para uma dada amostra de treinamento, é proporcional ao tamanho da rede. Algumas técnicas podem ser utilizadas tanto para acelerar o processo de convergência como para reduzir a incidência de mínimos locais. Dentre elas, destacam-se o uso de taxas de aprendizado adaptativas e o uso de um termo adicional na correção dos pesos, conhecido como *momentum* [58, 318, 386]. A idéia por trás da inclusão de uma taxa de *momentum* é a de se corrigir os pesos com base numa combinação entre o ajuste atual e aquele realizado no passo anterior. Isso tende a aumentar a velocidade de aprendizado (aceleração) e a reduzir o perigo de instabilidade (oscilação de pesos em valores de ravinhas). Porém, introduz uma “sobrecarga” a mais para o projetista (item (i) acima).

A tendência do *Backpropagation* em gerar *overfitting* prejudica sobremaneira o modelo neural resultante, porquanto, devido à grande *capacidade* desse modelo, fruto do elevado número de neurônios e conexões sinápticas geralmente requeridos, o estimador tende a exibir problemas de generalização [58]. Como resultado, a rede terá propensão a memorizar os dados de treinamento (inclusive, o ruído implícito), produzindo saídas de má qualidade para dados desconhecidos (de teste). Esse dilema aprendizado-generalização será mais bem discutido a seguir.

2.5 Aprendizado *versus* generalização

A relação entre aprendizado e generalização é, talvez, o mais importante tópico de estudo em RNAs [42, 58, 386], sendo que o primeiro conceito diz respeito à habilidade de se modelar, adaptativamente, o comportamento intrínseco aos dados, enquanto o segundo alude à capacidade do estimador de continuar a prever eficazmente sobre outros dados gerados pelo mesmo processo (ver Seção 2.2). O fato é

que, ao se treinar uma RNA sobre um conjunto de dados particular, o erro gerado (erro de treinamento) será somente uma estimativa do erro verdadeiro inerente ao estimador. Os formatos das duas funções de erro (estimada e verdadeira) tendem a ser similares em larga-escala, mas um pouco diferentes localmente, de modo tal que uma alteração (sobreajuste) dos parâmetros que venha a minimizar uma delas não irá, necessariamente, minimizar a outra, acarretando prejuízo de desempenho. Alguns aspectos relativos ao processo de aprendizado como um todo, tais como a menção prévia do tamanho e do particionamento do conjunto de dados, a seleção dos atributos, e o dimensionamento do número de parâmetros livres do modelo final, têm impacto direto nessa questão.

Tratando especificamente do problema de dimensionamento do tamanho da rede de acordo com a tarefa em vista, duas classes de abordagens complementares têm sido investigadas. A primeira classe, envolvendo abordagens *construtivas* [271, 361], recorre à estratégia de se iniciar com uma instância de topologia pequena (tipicamente, com um ou nenhum neurônio) para então ir adicionando, gradualmente, novas unidades de processamento, ou mesmo camadas destas, até que o processo convirja para uma topologia apropriada — de acordo com um critério de parada envolvendo tanto estimativas do erro de treinamento como do erro de generalização. Portanto, a resposta dessas redes, ditas *ontogenéticas*, se inicia em uma situação de *underfitting* e caminha em direção ao *overfitting*. Espera-se, daí, que a solução de melhor generalização seja encontrada na transição entre os dois extremos. Exemplos de tais algoritmos de crescimento (*growing algorithms*) abrangem o algoritmo “novo-rico” (*Upstart*), o algoritmo do telhado (*Tiling*), o algoritmo da pirâmide (*Pyramid*), o algoritmo da torre (*Tower*), o algoritmo *DistAl*, e o célebre algoritmo por correlação em cascata (*CasCor*). Esses algoritmos serviram de base de comparação para a abordagem coevolucionária anunciada na Seção 8.1, e são descritos e avaliados nos trabalhos de Ribeiro e Vasconcelos [390, 464].

A segunda classe de abordagens, conhecidas como *reducionistas* (ou subtrativas) e envolvendo os *métodos de poda* [385], inicia o processo de busca (aprendizado) estrutural com uma arquitetura sobredimensionada para depois ir removendo, gradativamente, recursos (nós/conexões) redundantes. Esses procedimentos, voltados ao ajuste da complexidade (regularização) estrutural da rede (como o *método de decaimento dos pesos* [204]), combinam as vantagens mencionadas na Seção 2.3, quais sejam, alta velocidade e poucos mínimos locais, para redes fartas de parâmetros, e maior capacidade de generalização, para redes de menor dimensão [72].

Algoritmos construtivos apresentam algumas vantagens quando cotejados com métodos de poda. Por exemplo, para redes ontogenéticas, a especificação do número inicial de neurônios se torna trivial, ao passo que, para métodos de poda, isso não é necessariamente verdadeiro, embora já existam trabalhos teóricos determinando limitantes superiores para o número de neurônios escondidos [42, 72]. Além do mais, algoritmos construtivos sempre buscam por soluções mais parcimoniosas primeiro, mostrando-se, na média, mais econômicos, computacionalmente falando, do que métodos de poda. Porém, algoritmos construtivos apresentam também deficiências. Por exemplo, por empregarem normalmente estratégias gulosas para a geração de novas estruturas, muitos algoritmos construtivos tendem a produzir soluções subótimas.

É sabido também que tanto algoritmos construtivos como subtrativos não garantem convergência

para as funções-geratrizes F_g dos dados, pelo fato de que ambos tendem a sofrer das mesmas dificuldades de treinamento inerentes a redes MLP. Além disso, esses algoritmos requerem ajustes de parâmetros de treinamento adicionais, aos quais os modelos resultantes tendem a ser bastante sensíveis. Porém, por se tratarem de metodologias complementares, parece ser bem promissor combiná-las de algum modo. Assim, existem propostas que estipulam o uso inicial de um método construtivo, que fica livre para introduzir recursos além do necessário, para depois se aplicar um método de poda, que fica responsável por eliminar recursos redundantes. Por outro lado, existem abordagens onde os dois tipos de procedimentos competem simultaneamente entre si, onde um processo tenta adicionar e o outro tenta remover recursos. Encaixam-se aqui as técnicas evolutivas de treinamento de RNAs (Seção 6.9), em particular a HCGA (Subseção 8.1). As abordagens COHENN (Capítulo 7) e HGT-GAME (Seção 8.3) também lançam mão desse artifício quando do tratamento da especificação automática de sistemas com múltiplos módulos de RNAs.

De um ponto de vista mais teórico, Geman *et al.* [171] provêem uma análise formal minuciosa do relacionamento entre aprendizado e generalização em RNAs, tomando como referência os conceitos de polarização (*bias*) e variância. Segundo esse *dilema Polarização-Variância*, um modelo predefinido que seja menos dependente dos dados (ou mesmo um modelo subparametrizado) tende a representar erroneamente a função-geratriz, incorrendo em maior polarização para um determinado tipo de resposta. Ao mesmo tempo, um modelo direcionado pelos dados (ou sobreparametrizado) tende a exibir uma maior variabilidade nas suas respostas.

Sem perda de generalidade, como forma de introduzir formalmente esse raciocínio, considere um problema de classificação binária em que a saída desejada $y_d \in \{0, 1\}$ esteja relacionada ao vetor de entradas $\mathbf{x}$ por $y_d = F_g(\mathbf{x}) + \varepsilon$, onde ε designa uma variável aleatória de média nula e $F_g(\mathbf{x})$ indica a função-geratriz, expressa como a esperança condicional de y_d dado o padrão $\mathbf{x}$, ou seja, $F_g(\mathbf{x}) = E(y_d|\mathbf{x})$. De posse de um conjunto de treinamento D_N de tamanho N , o objetivo final é o de se gerar uma estimativa $f(\mathbf{x}; D_N)$ de $F_g(\mathbf{x})$ tal que, com ela, o erro de estimação total seja minimizado. Considerando o EQM como medida de desempenho, então tal colocação se traduz em

$$\begin{aligned} \text{EQM} &= E[(y_d - f(\mathbf{x}; D_N))^2] \\ &= E[(y_d - F_g(\mathbf{x}))^2] + E[(f(\mathbf{x}; D_N) - F_g(\mathbf{x}))^2]. \end{aligned}$$

Como a função-erro depende da escolha do conjunto D_N , uma mudança nesse conjunto (ou no seu número de amostras) pode resultar em uma alteração na função gerada f e, portanto, no erro de estimação. Considerando a possível aleatoriedade na escolha dos pontos do conjunto D_N , o erro de predição total do modelo pode ser escrito como

$$E_{\mathcal{D}} \{E[(y_d - f(\mathbf{x}; D_N))^2]\} = E[(y_d - F_g(\mathbf{x}))^2] + E_{\mathcal{D}}[(f(\mathbf{x}; D_N) - F_g(\mathbf{x}))^2], \quad (2.9)$$

sendo que $E_{\mathcal{D}}$ denota a esperança sobre todos os possíveis subconjuntos formados por N amostras quaisquer oriundas da população de pontos $\mathcal{D}$ de onde D_N se originou. Como o primeiro termo no lado direito de (2.9), $E[(y_d - F_g(\mathbf{x}))^2] = E[\varepsilon^2]$, independe tanto do conjunto de treinamento como da função estimativa, ele

reflete o erro irreduzível de estimação devido ao ruído intrínseco aos dados. Já o segundo termo oferece uma medida natural do grau de eficácia de $f(\mathbf{x}; D_N)$ como estimador de y_d . Essa parcela pode ser decomposta mais ainda [171], dando origem a

$$E_{\mathcal{D}}[(f(\mathbf{x}; D_N) - E(y_d|\mathbf{x}))^2] = \{E_{\mathcal{D}}[f(\mathbf{x}; D_N)] - E(y_d|\mathbf{x})\}^2 + E_{\mathcal{D}}\{(f(\mathbf{x}; D_N) - E_{\mathcal{D}}[f(\mathbf{x}; D_N)])^2\}. \quad (2.10)$$

Essa é a famosa *Decomposição Polarização-Variância* (DPV) do erro de estimação de uma máquina de aprendizado qualquer. O primeiro termo do lado direito da Eq. (2.10) diz respeito ao quadrado da polarização ($B_{\mathcal{D}}(f(\mathbf{x}))$). Ele provê uma noção quantitativa de quão distante da função-geratriz F_g está a média das funções-estimativas geradas sobre todos os possíveis conjuntos de dados de tamanho N . Já o segundo termo, que simboliza a variância do modelo ($V_{\mathcal{D}}(f(\mathbf{x}))$), fornece uma medida quantitativa da sensibilidade da função estimada ao seu particular conjunto de treinamento D_N . Como polarização e variância são incompatíveis, para um dado conjunto de dados fixo, à proporção que se tenta reduzir um dos fatores, estar-se-á incrementando o outro. Daí, uma boa abordagem de treinamento deve propiciar um equilíbrio justo entre esses fatores, levando a ganhos de generalização para a máquina (neural) resultante.

2.6 Considerações finais

Neste capítulo, foi proporcionada uma visão geral sobre alguns dos pontos principais de estudo em redes neurais multicamadas, discorrendo sobre suas propriedades, sobre aspectos estruturais e operacionais (treinamento), e sobre o dilema “aprendizado-generalização”. Em seguida, duas das linhas mais recentes de pesquisa em AM, máquinas de vetores-suporte e comitês de máquinas, que tentam lidar com o problema de generalização em RNAs sob perspectivas diferentes, serão tratadas.

Capítulo 3

Máquinas de vetores-suporte

“Aqueles que não conseguem se recordar do passado estão condenados a repeti-lo.”
Santayana

Recentemente, um novo tipo de metodologia de aprendizado, com características operacionais semelhantes às aquelas exibidas por sistemas conexionistas, foi proposto por Vapnik e colaboradores como forma de se implementar conceitos teóricos da área de aprendizado estatístico [57, 96, 462, 463]. O indutor por trás dessa metodologia, cujo estimador associado passou a ser chamado pela alcunha de *Máquina de Vetores-Suporte* (doravante SVM, advindo do inglês *Support Vector Machine*), se distingue dos modelos conexionistas em alguns aspectos fundamentais. Em vez de seguir o *Princípio da minimização do risco empírico*, por exemplo, SVMs são concordantes com o *Princípio da minimização do risco estrutural*, que procura minimizar um limitante superior do erro de generalização no lugar de otimizar somente o erro de treinamento. O princípio de indução por trás dessa nova estratégia de aprendizado apóia-se na hipótese de que a taxa de erro de uma máquina de aprendizado sobre o conjunto de teste (i. e., o erro de generalização) seja limitada pela soma entre o erro de treinamento e um termo de intervalo de confiança dependente da *dimensão de Vapnik-Chervonenkis* — *dimensão VC*⁽¹⁾. É com base nesse princípio que uma dada instância de máquina de vetores-suporte consegue produzir uma estrutura otimizada de rede (resultando, eventualmente, em melhor desempenho de generalização quando comparada a modelos tradicionais de redes neurais), mesmo não incorporando nenhuma informação acerca do problema-alvo.

Outro mérito exibido por esse novo método de aprendizado de máquina é que o treinamento de classificadores de vetores-suporte se resume a resolver um problema de programação quadrática (PPQ) com

⁽¹⁾A dimensão VC corresponde a uma métrica formal indicativa do grau de complexidade inerente a uma *classe* de funções, tendo em vista o conjunto de dados sob análise [5, 197, 463]. Quanto mais *complexa* for a classe de funções (i. e., maior a sua dimensão VC), menor será a sua capacidade de generalização intrínseca em relação ao processo gerador dos dados. Desse modo, pode-se lançar mão de tal critério estatístico para fins de seleção de modelos e/ou análise de desempenho, em substituição, por exemplo, ao método de validação cruzada (ver Apêndice A). Sontag [439] discute métodos diferentes de como estimar tal escalar em se tratando especificamente de RNAs, ao passo que Semolini [418] provê uma boa introdução teórica ao tema.

restrições lineares. Isso representa um grande avanço frente a algoritmos convencionais de treinamento de RNAs (que empregam otimização não-linear), já que a solução (configuração da máquina) obtida é, ao mesmo tempo, *única, ótima e não acometida de mínimos locais* [68].

Basicamente, a metodologia SVM pode ser resumida em dois passos sequenciais [198]: (1) mapear não-linearmente um vetor de entradas em um espaço de características de mais alta dimensão (que seja “escondido” tanto da entrada como da saída); e (2) construir um hiperplano ótimo que separe as características extraídas no passo anterior. O primeiro passo segue as diretrizes impetradas pelo Teorema de separabilidade de padrões [97, 318] sobre a separabilidade de padrões (ver Seção 2.2), justificando o emprego de funções com características especiais, denominadas *funções-kernel* [414, 415]. A tarefa básica de tais funções é o de se extrair novas características de modo computacionalmente eficiente, a fim de se amenizar os problemas relativos à “maldição da dimensionalidade” [130, 197].

A seguir, passa-se à formalização passo-a-passo dos conceitos supracitados, culminando com a apresentação da arquitetura típica de uma máquina de vetores-suporte. Posteriormente, alguns aspectos adicionais serão brevemente enaltecidos.

3.1 Formalização de máquinas de vetores-suporte

Supondo que o conjunto de treinamento consista de N pares $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$, com $\mathbf{x}_i \in \mathbb{R}^D$ e $y_i \in \{-1, 1\}$, pode-se definir um hiperplano de separação linear entre as classes como sendo

$$\{\mathbf{x} : f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = 0\}, \quad (3.1)$$

onde $\mathbf{w}$ é um vetor com norma unitária, ou seja, $\|\mathbf{w}\| = 1$. Uma regra de classificação induzida por $f(\mathbf{x})$ é dada por

$$G(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b), \quad (3.2)$$

uma vez que $f(\mathbf{x})$ fornece a distância entre um dado ponto $\mathbf{x}$ e o hiperplano definido em (3.1). Considerando, primeiramente, que as classes são linearmente separáveis, pode-se projetar uma função $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$, com $y_i f(\mathbf{x}_i) > 0, \forall i$, de tal modo que o hiperplano resultante apresente a maior *margem de separação* entre os pontos de treinamento pertencentes a ambas as classes. O problema de otimização a seguir captura tal conceito:

$$\max_{\mathbf{w}, b, \|\mathbf{w}\|=1} C$$

$$\text{sujeito a } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq C, \quad i = 1, \dots, N. \quad (3.3)$$

A banda (margem) ilustrada na Fig. 3.1 tem $2C$ unidades de largura, sendo que seu ponto médio (sob a fronteira de decisão) está distante, pelo menos, C unidades das amostras de cada classe. Geralmente,

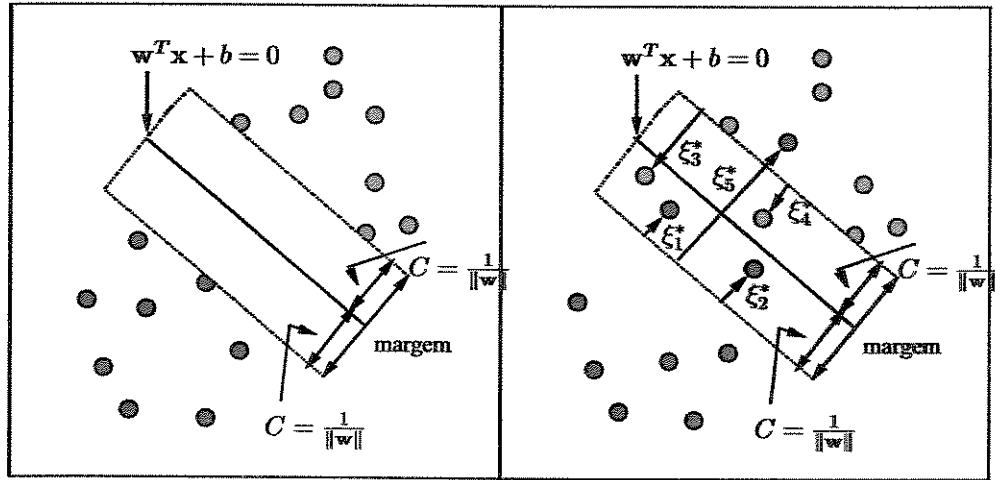


Fig. 3.1: Modo de operação de um classificador com margem larga. O painel à esquerda mostra o caso de separação linear. A fronteira de decisão é indicada pela linha sólida, enquanto as linhas tracejadas delimitam a largura da margem, no caso $2C = \frac{2}{\|w\|}$. O painel direito ilustra o caso de sobreposição entre as classes. Os pontos rotulados com ξ_j^* estão em lados errados da margem, com uma distância dada por $\xi_j^* = C\xi_j$; pontos localizados em lados corretos da margem têm $\xi_j^* = 0$. A margem é maximizada atendendo-se à restrição $\sum \xi_i \leq \text{constante}$. Disso, $\sum \xi_j^*$ denota a distância total relativa aos pontos localizados em lados errados da margem. (Fonte: Hastie *et al.* [197].)

sob a perspectiva de SVMs, o problema (3.3) passa a ser reescrito sob uma outra roupagem, exibindo critério quadrático e restrições de desigualdade lineares [100]. A nova formulação é obtida modificando-se as N restrições em (3.3), dando origem a $y_i(w^T x_i + b) \geq C\|w\|$, de tal modo que a condição $\|w\| = 1$ seja abolida. Assim, uma vez que, para quaisquer w e b atendendo às desigualdades acima, qualquer múltiplo positivo da solução também irá satisfazê-las, pode-se fixar, arbitrariamente, que $C = \frac{1}{\|w\|}$. Com essa transformação, chega-se ao problema equivalente, dado a seguir:

$$\begin{aligned} \min_{w, b} \quad & \|w\| \\ \text{sujeito a} \quad & y_i(w^T x_i + b) \geq 1, \quad i = 1, \dots, N. \end{aligned} \quad (3.4)$$

Supondo agora que as classes apresentem sobreposições no espaço de características (ou seja, não são linearmente separáveis), pode-se continuar a maximizar C desde que seja permitida a existência de alguns pontos “malcomportados”, isto é, aparecendo em lados opostos da margem. Apelando-se para o artifício de se empregar *variáveis de folga*⁽²⁾ $\xi = [\xi_1, \xi_2, \dots, \xi_N]^T$, há dois modos naturais de se alterar as restrições em (3.3):

$$y_i(w^T x_i + b) \geq C - \xi_i,$$

⁽²⁾Do inglês, “*slack variables*”.

ou

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq C(1 - \xi_i),$$

$\forall i, \xi_i \geq 0, \sum_{i=1}^N \xi_i \leq \text{constante}$. Os dois caminhos levam a diferentes soluções, sendo que a segunda opção conduz à definição “canônica” de um classificador de vetores-suporte [197].

Com a introdução das variáveis de folga, a formalização para o caso de classes linearmente não-separáveis se torna trivial como se segue. O valor de ξ_i na restrição $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq C(1 - \xi_i)$ oferece, proporcionalmente, uma medida quantitativa de quão errônea (i. e., quão distante do lado certo da margem) é a predição $f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i + b$. Daí, estabelecendo-se um limite à soma $\sum \xi_i$, pode-se estipular a quantidade proporcional de predições não-corretas definidas pela fronteira. Atentando-se ao fato de que uma classificação errada ocorre quando $\xi_i > 1$, ao se restringir $\sum \xi_i$ a um valor Λ qualquer, está-se, conseqüentemente, delimitando que o número total de má classificações no treinamento seja, no máximo, Λ .

Da mesma forma como feito para o caso anterior, pode-se recorrer a $C = \frac{1}{\|\mathbf{w}\|}$ e escrever (3.4) na forma adaptada

$$\begin{aligned} & \min_{\mathbf{w}, b} \|\mathbf{w}\| \\ \text{sujeito a } & \begin{cases} y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \forall i, \\ \xi_i \geq 0, \sum \xi_i \leq \text{constante.} \end{cases} \end{aligned} \quad (3.5)$$

A Fig. 3.1 traz um esboço pictórico do processo de decisão descrito acima. Dada a natureza das restrições impostas em (3.5), pode-se constatar que os pontos residindo dentro da região da classe a que pertencem não desempenham nenhum papel quando da definição da fronteira de decisão. Isso é uma propriedade bastante atraente que diferencia os classificadores de margens largas de outros de natureza estatística, tais como aqueles produzidos por Análise Discriminante Linear (de *Linear Discriminant Analysis* — LDA) [197], cuja fronteira associada passa a ser determinada pela covariância das distribuições das classes e pelas posições de seus centróides — não havendo, portanto, nenhuma discriminação entre as amostras.

O problema de otimização quadrática formulado em (3.5) é de natureza convexa, sendo que a sua solução pode ser obtida mediante a aplicação de multiplicadores de Lagrange ($\alpha = [\alpha_1, \alpha_2, \dots, \alpha_N]^T$ e $\gamma = [\gamma_1, \gamma_2, \dots, \gamma_N]^T$). Computacionalmente, é conveniente expressar (3.5) numa forma equivalente:

$$\begin{aligned} & \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + \mu \sum_{i=1}^N \xi_i \\ \text{sujeito a } & \xi_i \geq 0, \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i, \end{aligned} \quad (3.6)$$

onde μ substitui a constante em (3.5), sendo que, para o caso de total separabilidade entre as classes, $\mu = \infty$. Nessa função-custo, a primeira parcela ($\frac{1}{2} \|\mathbf{w}\|^2$) diz respeito a um termo de *regularização*, responsável por

mensurar a complexidade (e, por conseguinte, a capacidade — ver Subseção 2.3.3) do modelo a ser gerado. O segundo termo, por sua vez, acena para o risco empírico (que indica a qualidade do mapeamento sobre os dados de treinamento), sendo que $\sum_i \xi_i$ é tipicamente substituído por uma *função de perda* apropriada (vide Seção 3.2). O parâmetro μ , conhecido como *constante de regularização*, determina uma relação de compromisso entre o termo de regularização e o risco empírico, tendo seu valor previamente configurado.

A função lagrangeana primal correspondente à formulação (3.6) é

$$L_{CP}(\mathbf{w}, b, \xi_i, \alpha_i, \gamma_i) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \mu \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i \{ [\mathbf{w}^T \phi(\mathbf{x}_i) + b] y_i - (1 - \xi_i) \} - \sum_{i=1}^N \gamma_i \xi_i, \quad (3.7)$$

sendo que, daqui em diante, $\phi(\cdot)$ simboliza um vetor de funções $\varphi_t(\cdot)$ (lineares ou não), não explicitamente construídas pelo projetista, que mapeiam o espaço de entradas em um espaço de maior dimensão projetado por características extraídas — o seu papel será discutido mais adiante. Fixando as derivadas de (3.7) em zero, obtém-se

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \phi(\mathbf{x}_i), \quad (3.8)$$

$$0 = \sum_{i=1}^N \alpha_i y_i = \sum_{i=1}^N \nu_i, \quad (3.9)$$

$$\alpha_i = \mu - \gamma_i, \forall i, \quad (3.10)$$

juntamente com as restrições de positividade $\alpha_i, \gamma_i, \xi_i \geq 0, \forall i$. Substituindo (3.8)-(3.10) em (3.7), chega-se à função-objetivo dual

$$L_{CD}(\alpha_i) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j), \quad (3.11)$$

que provê um limitante inferior para a função-objetivo descrita em (3.6). A idéia é maximizar L_{CD} sujeitando-se a $0 \leq \alpha_i \leq \mu$ e $\sum_{i=1}^N \alpha_i y_i = 0$. Além de (3.8)-(3.10), existem outras restrições impostas pelas condições de Karush-Kuhn-Tucker (KKT) [100], originando

$$\alpha_i \{ y_i [\mathbf{w}^T \phi(\mathbf{x}_i) + b] - (1 - \xi_i) \} = 0, \quad (3.12)$$

$$\gamma_i \xi_i = 0, \quad (3.13)$$

$$y_i [\mathbf{w}^T \phi(\mathbf{x}_i) + b] - (1 - \xi_i) \geq 0, \quad (3.14)$$

para $i = 1, \dots, N$. Juntas, as Equações (3.8)-(3.14) caracterizam unicamente a solução para ambos os problemas, primal e dual⁽³⁾.

⁽³⁾O PPQ por trás da maximização da função dual apresenta-se, computacionalmente, mais simples de ser resolvido do que aquele para a minimização da função primal, sendo o segundo geralmente preterido em favor do primeiro.

Da expressão (3.8) advém o fato de que a solução para o vetor $\mathbf{w}$ apresenta o seguinte formato:

$$\hat{\mathbf{w}} = \sum_{i=1}^N \hat{\alpha}_i y_i \phi(\mathbf{x}_i), \quad (3.15)$$

onde os coeficientes $\hat{\alpha}_i$ são não-nulos somente para aquelas observações para as quais as restrições em (3.14) sejam estritamente satisfeitas — devido a (3.12). Tais observações são chamadas de *vetores-suporte* (SV) [463], visto que, recorrendo-se somente a tais pontos, pode-se construir $\hat{\mathbf{w}}$ completamente. Dentre as amostras que compõem tal subconjunto especial, algumas (p. ex., q delas) irão residir sobre a borda da margem de separação (ou seja, $\xi_i = 0$) e, por conta das restrições (3.13) e (3.10), serão caracterizadas por $0 < \alpha_i < \mu$. As amostras restantes ($\xi_i > 0$) terão $\alpha_i = \mu$. Da expressão (3.12), pode-se averiguar que qualquer um dos q pontos sobre a borda ($\alpha_i > 0, \xi_i = 0$) pode ser usado a fim de se obter uma medida do *threshold* $\hat{b}$, sendo que essa medida, por razões de estabilidade, é tipicamente obtida via cálculo da média sobre todos os q possíveis valores [197].

De posse das soluções $\hat{\mathbf{w}}$ e $\hat{b}$, pode-se projetar a função de decisão como sendo dada por $\text{sign}[\hat{f}(\mathbf{x})] = \text{sign}[\hat{\mathbf{w}}^T \phi(\mathbf{x}) + \hat{b}]$, sendo que o parâmetro de regularização μ define o tamanho da margem de separação (e, por conseguinte, quais pontos serão vetores-suporte), controlando a qualidade do modelo gerado. Corroborando essa posição, existe um teorema estabelecido por Vapnik [463] que assegura que, por intermédio de uma escolha apropriada da margem de separação, pode-se controlar a dimensão VC (isto é, complexidade) do hiperplano ótimo a ser gerado (ver Haykin [198, Eq. (6.19)]). Com relação à calibragem apropriada desse parâmetro, Hastie *et al.* [197] ressaltam que “valores altos de μ tendem a desencorajar a existência de ξ_i positivos, ao mesmo tempo que promover a criação de uma superfície sobreajustada dentro do espaço de entradas original; em contrapartida, valores pequenos de μ tendem a induzir a valores pequenos de $\|\mathbf{w}\|$, propiciando, por sua vez, o projeto de uma fronteira de decisão mais suave”.

O classificador de vetores-suporte descrito até o momento projeta somente superfícies lineares de decisão sobre o espaço de entradas. Contudo, pode-se tornar o procedimento descrito acima mais flexível lançando-se mão de expansões-base não-lineares, tais como polinômios e *splines*. Esse é o papel por trás das funções-base $\varphi_t(\cdot)$ agrupadas em $\phi(\cdot)$ (ver (3.7)): transformar o espaço de características de modo a que se obtenha uma melhor separação entre os pontos de treinamento (seguindo o Teorema de separabilidade de padrões). Desse modo, pode-se gerar uma função não-linear de separação $\text{sign}[\hat{f}(\mathbf{x})]$ seguindo o mesmo procedimento anterior.

Uma máquina de vetores-suporte (SVM) é uma extensão dessa idéia, por meio da qual se permite que a dimensão do espaço ampliado atinja valores bem elevados, infinitos em alguns casos. Dessa asserção, poder-se-ia imaginar que (1) os cálculos computacionais resultantes se tornariam proibitivos; e (2) dado um número suficiente (finito) de funções-base, os dados se tornariam linearmente separáveis, acarretando, provavelmente, sobreajuste aos dados (e perda da capacidade de generalização). Todavia, a metodologia SVM lida com tais aspectos mediante uma nova representação para o problema de otimização (3.7), a qual envolve os atributos de entrada somente via produtos internos. A função lagrangeana dual (3.11) passa a

Tab. 3.1: Lista de possíveis *kernels* e valores *default* de seus parâmetros associados. Aqui, $\mathbf{x}, \mathbf{z}$ são duas amostras D -dimensionais quaisquer, sendo que x^d, z^d representam os valores para o seu d -ésimo atributo ($d = 1, \dots, D$).

Nome	Definição
Linear	$K_l(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$
Polinomial	$K_p(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z} + 1)^p$, para $p = 2$
Base radial (RBF)	$K_{rbf}(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\ \mathbf{x} - \mathbf{z}\ ^2}{2\sigma^2}\right)$, para $\sigma^2 = 1$
Base radial exponencial (ERBF)	$K_{erbf}(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\sqrt{\ \mathbf{x} - \mathbf{z}\ ^2}}{2\sigma^2}\right)$, para $\sigma^2 = 1$
Tangente hiperbólica	$K_{tgh}(\mathbf{x}, \mathbf{z}) = \tanh(a(\mathbf{x}^T \mathbf{z}) - c)$, para $a = 1, c = 0$
Fourier	$K_f(\mathbf{x}, \mathbf{z}) = \prod_{d=1}^D \left\{ \frac{\sin(\delta + \frac{1}{2})(x^d - z^d)}{\sin(\frac{1}{2}(x^d - z^d))} \right\}$, para $\delta = 1$
<i>Spline</i> linear	$K_{sp}(\mathbf{x}, \mathbf{z}) = \prod_{d=1}^D \left\{ 1 + x^d z^d + x^d z^d \min(x^d, z^d) - \frac{(x^d + z^d)}{2} (\min(x^d, z^d))^2 + \frac{1}{3} (\min(x^d, z^d))^3 \right\}$
<i>B_t-spline</i>	$K_{bsp}(\mathbf{x}, \mathbf{z}) = \prod_{d=1}^D \left\{ \sum_{r=0}^{2(t+1)} (-1)^r \binom{2(t+1)}{r} \times [\max(0, x^d - z^d + t + 1 - r)]^{(2t+1)} \right\}$, para $t = 1$
Anova- <i>spline</i>	$K_{asp}^p(\mathbf{x}, \mathbf{z}) = \frac{1}{p} \sum_{s=1}^p (-1)^{s+1} K_{asp}^{p-s}(\mathbf{x}, \mathbf{z}) \cdot \tilde{K}^s(\mathbf{x}, \mathbf{z})$, onde: $K_{asp}^0 = 1$, $\tilde{K}^s(\mathbf{x}, \mathbf{z}) = \sum_{d=1}^D [K_{sp}(x^d, z^d)]^s$, para $p = 2$

assumir, conseqüentemente, uma nova formulação:

$$L_{CD}(\alpha_i) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \quad (3.16)$$

onde $K(\cdot)$ deve ser (semi-)definida positiva, simétrica e interpretada como $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. Tal expansão (denominada *kernel*) constitui-se um caso particular do Teorema de Mercer [415], oriundo do estudo de análise funcional. Por meio desse teorema, pode-se certificar se uma função-candidata qualquer é realmente um *kernel* com produto interno definido em algum espaço multidimensional e, portanto, admissível para ser adotada em uma instância de SVM. Entretanto, tal teorema não oferece nenhuma pista sobre como se construir as funções $\phi_t(\mathbf{x})$, tarefa que pode ser, muitas vezes, não-trivial, segundo Haykin [198]. Sob a perspectiva de *kernels*, a solução final pode ser escrita como sendo

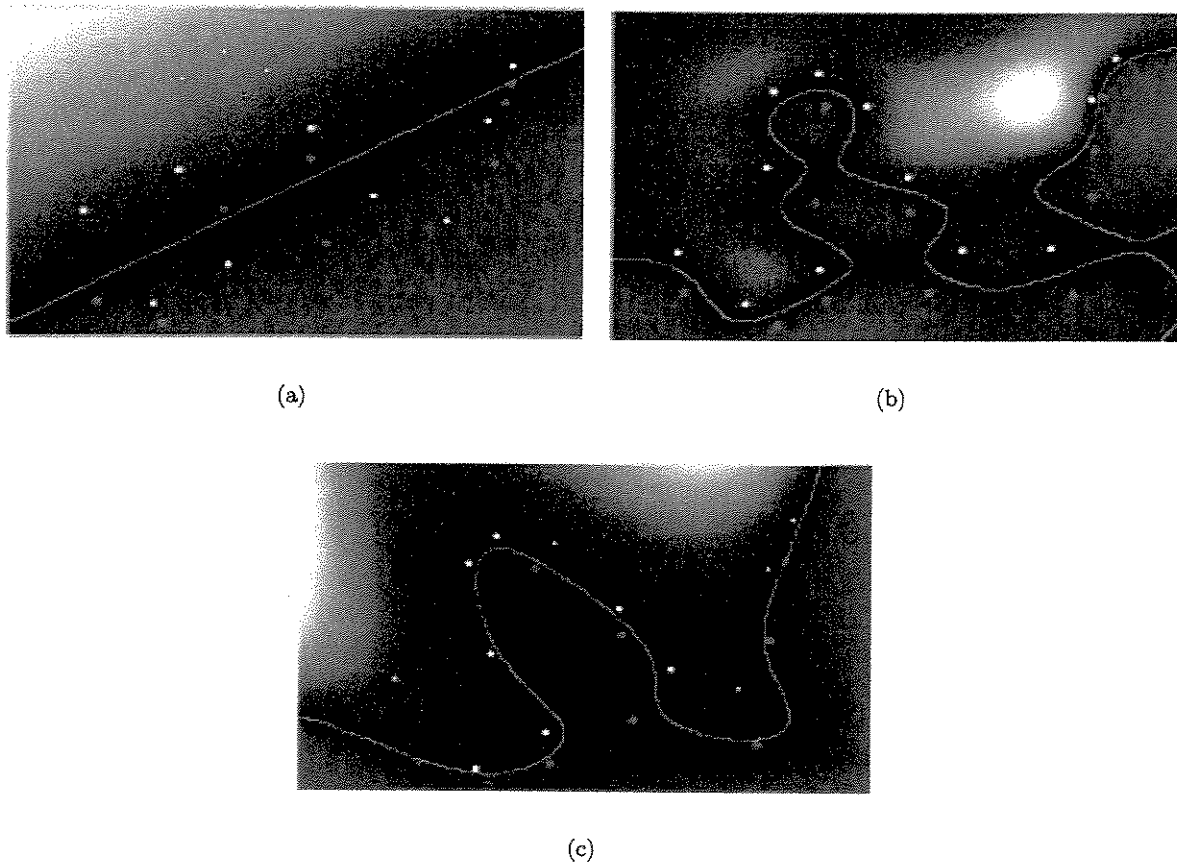


Fig. 3.2: Superfícies de decisão geradas por *kernels* distintos para um problema fictício com duas dimensões de entrada ($N = 24$ amostras, $\mu = 10.000$). Pontos em destaque correspondem aos vetores-suporte. (a) *kernel* linear: 18 SVs; (b) *kernel* RBF ($\sigma^2 = 1$): 23 SVs; e (c) *kernel* polinomial ($p = 5$): 15 SVs.

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^N \hat{\alpha}_i y_i K(\mathbf{x}, \mathbf{x}_i) + \hat{b}.$$

A Tabela 3.1 apresenta um conjunto de alternativas para $K(\cdot)$ (com valores-padrão para seus parâmetros associados) [186, 463], sendo essa lista adotada nos experimentos reportados na Seção 8.2. Por sua vez, a Fig. 3.2 traz um esboço de algumas superfícies de decisão associadas a *kernels* distintos, considerando um problema de classificação hipotético; fica evidente que *kernels* distintos podem implicar desempenhos distintos para uma mesma máquina de vetores-suporte (isto é, com valores fixos para os outros parâmetros).

Assumindo, como exemplo fictício, um espaço de entradas com duas amostras $\mathbf{X} = (X_1, X_2)$ e $\mathbf{X}' = (X'_1, X'_2)$, ao se fazer uso de um *kernel* polinomial de segunda ordem, pode-se chegar à seguinte

expansão [197]:

$$\begin{aligned} K(\mathbf{X}, \mathbf{X}') &= (1 + \langle \mathbf{X}, \mathbf{X}' \rangle)^2 \\ &= (1 + X_1 X'_1 + X_2 X'_2)^2 \\ &= 1 + 2X_1 X'_1 + 2X_2 X'_2 + (X_1 X'_1)^2 + (X_2 X'_2)^2 + 2X_1 X'_1 X_2 X'_2. \end{aligned}$$

É fácil verificar, aqui, a existência de seis possíveis funções-base, dadas por $\varphi_1(\mathbf{X}) = 1$, $\varphi_2(\mathbf{X}) = \sqrt{2}X_1$, $\varphi_3(\mathbf{X}) = \sqrt{2}X_2$, $\varphi_4(\mathbf{X}) = X_1^2$, $\varphi_5(\mathbf{X}) = X_2^2$ e $\varphi_6(\mathbf{X}) = \sqrt{2}X_1 X_2$, de tal modo que $K(\mathbf{X}, \mathbf{X}') = \langle \phi(\mathbf{X}), \phi(\mathbf{X}') \rangle$, sendo que $\phi(\cdot) = [\varphi_1(\cdot), \dots, \varphi_6(\cdot)]^T$.

Finalmente, a Fig. 3.3 ilustra a arquitetura típica de uma instância de SVM qualquer, já se levando em conta os vetores-suporte obtidos na fase de treinamento; fica aparente a similaridade estrutural com RNAs, embora, nesse contexto, os “módulos” não sejam neurônios, mas sim as funções-base encapsuladas em $\phi(\cdot)$.

3.2 Pontos adicionais

O processo de formalização descrito acima pode ser generalizado para o caso em que $y_i \in \mathbb{R}$, ou seja, quando se pretende configurar uma instância de SVM para problemas de regressão ou previsão [186, 327]. A estratégia continua sendo a mesma: mapear $\mathbf{x}$ em um espaço de alta dimensão, onde será feita a aproximação. Para tanto, o algoritmo tenta construir a função de decisão de tal modo que os pontos de treinamento residam a uma distância inferior ou igual a um valor $\epsilon > 0$, conhecido como *tamanho do tubo*, que estipula o grau de resolução do ajuste aos dados desejado pelo projetista. Nesse novo cenário, o papel da função de perda (ver discussão sucedendo à Eq. (3.6)) se torna mais proeminente. Tipicamente, recorre-se à *função ϵ -insensitiva* [100], dada por

$$V(e_i) = \begin{cases} 0 & \text{se } |e_i| < \epsilon \\ |e_i| - \epsilon & \text{caso contrário} \end{cases},$$

sendo $e_i = y_i - f(x_i)$, visto que ela habilita o uso de amostras esparsas⁽⁴⁾ para representar a função de decisão. Outros exemplos de função de perda incluem a *função ϵ -insensitiva quadrática*, similar à função-custo EQM em redes neurais, e a *função de Huber*, que se apresenta como um misto das duas anteriores (parte linear, parte quadrática) [197, 463].

Para se aplicar o algoritmo de aprendizado em um problema de regressão, especifica-se *a priori* o valor do tamanho do tubo ou assinala-se um limitante superior para a fração dos pontos de treinamento que possa

⁽⁴⁾Esparsividade tende a ser um requisito importante, em termos de desempenho computacional, para a aplicação dos métodos de resolução de problemas de programação quadrática, principalmente quando há necessidade de manipulação de matrizes de ordens elevadas.

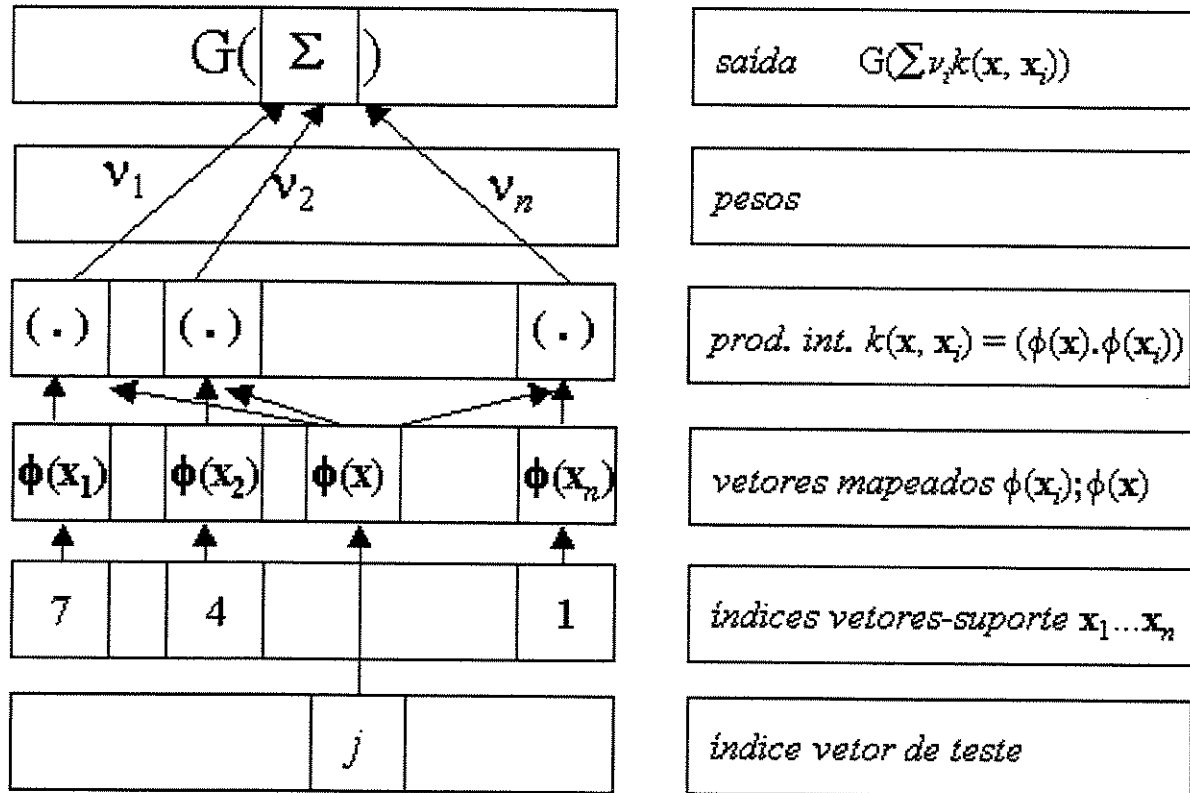


Fig. 3.3: Arquitetura típica de uma instância de SVM. O vetor de entradas x e os vetores-suporte x_i são mapeados (não-)linearmente (pelo vetor de funções-base ϕ) em um espaço F , onde os produtos internos são calculados. Lançando-se mão de um *kernel* K , essas duas camadas são, na prática, computadas num único passo. Os resultados são linearmente combinados por pesos ν_i , que são encontrados após se resolver o PPQ (em problemas de classificação, $\nu_i = y_i \alpha_i$; em problemas de regressão, $\nu_i = (\alpha_i^* - \alpha_i)^2$). A combinação linear é introduzida na função G — para classificação, $G(x) = \text{sign}(x + b)$, e para regressão, $G(x) = x + b$. (Fonte: Schölkopf [201].)

residir fora do tubo, de tal modo que o valor de ϵ seja resgatado automaticamente [415]. Com a introdução de novos multiplicadores de Lagrange ($\alpha^* = [\alpha_1^*, \alpha_2^*, \dots, \alpha_N^*]^T$), que devem satisfazer a novas restrições, pode-se formalizar o problema dual de programação quadrática para regressão como sendo [197, 462]:

$$L_{RD}(\alpha_i, \alpha_i^*) = \epsilon \sum_{i=1}^N (\alpha_i^* + \alpha_i) - \sum_{i=1}^N y_i (\alpha_i^* - \alpha_i) + \frac{1}{2} \sum_{i,j=1}^N (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j)$$

sujeito a $\sum_{i=1}^N (\alpha_i^* - \alpha_i) = 0, \quad 0 \leq \alpha_i, \alpha_i^* \leq \mu, \quad \alpha_i^* \alpha_i = 0 \quad \forall i,$ (3.17)

onde, agora, somente os pontos cujos coeficientes $(\alpha_i^* - \alpha_i)$ sejam não-nulos podem ser considerados como vetores-suporte. Geralmente, quanto maior o valor de ϵ , menor será o número de vetores-suporte e mais esparsa a representação da solução. Em contrapartida, valores altos para esse parâmetro podem depreciar a qualidade da aproximação (*fitting*).

Máquinas de vetores-suporte são insensitivas ao número de dimensões (atributos) de cada ponto amostral, ao passo que seu desempenho computacional é influenciado sobremaneira pelo número de amostras de treinamento. Isso por conta do modelo de otimização utilizado, que é quadrático em relação ao número de parâmetros livres (α_i). A fim de lidar com problemas de elevada complexidade, existem abordagens alternativas de implementação da etapa de otimização visando a acelerar o processo de convergência à solução final. Uma classe de abordagens faz uso de estruturas de dados sofisticadas, evitando a necessidade de se armazenar matrizes enormes em memória [201]. Outra classe de abordagens segue a estratégia de se decompor um grande PPQ em uma série de subproblemas (i. e., subconjuntos dos dados), sendo estes resolvidos iterativamente — os SVs são descobertos incrementalmente, ou seja, aqueles encontrados em uma geração são adicionados aos dados da próxima. Dentre esses métodos heurísticos, destacam-se o *chunking*, proposto por Vapnik [463], e a Otimização Mínima Seqüencial (SMO — *Sequential Minimal Optimization*), proposta por Platt [100]. Outra alternativa possível é o algoritmo *Kernel-Adatron* [404], que se trata de um método de otimização bastante rápido para aproximar a solução do processo de aprendizado dos vetores-suporte, uma vez que também é orientado a decomposições.

Recentemente, uma variação de SVMs, chamada de *Least-Squares SVMs* (LS-SVMs), foi introduzida na literatura com o intuito de possibilitar também aumentos de desempenho computacional, sendo tal extensão baseada no uso de um critério de erro quadrático juntamente com restrições de igualdade. A formulação para LS-SVMs constitui-se, pois, numa modificação simples da Eq. (3.6), originando:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + \mu \sum_{i=1}^N (\xi_i)^2$$

sujeito a $y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) = 1 - \xi_i, \quad i = 1, \dots, N.$

Esse problema é convexo e pode ser resolvido também via emprego de multiplicadores de Lagrange (α_i). Recorrendo-se às condições de KKT, obtém-se (i) $\alpha^T \mathbf{y} = 0$, onde $\mathbf{y} = [y_1, \dots, y_N]^T$; e (ii) $\mathbf{y}b + (\Omega + \mu^{-1}\mathbf{I})\alpha =$

$\mathbf{1}_N$, onde $\mathbf{1}_N$ é um vetor composto por N números “1” e $\Omega(i, j) = y_i y_j K(x_i, x_j), 1 \leq i, j \leq N$. Essa formulação e sua versão dual são apresentadas por Suykens *et al.* [446] e, porque as restrições são todas de igualdade, a solução final é obtida resolvendo-se um sistema de $N + 1$ equações lineares. Os requisitos de memória para LS-SVMs são geralmente mais brandos, uma vez que é necessário somente se armazenar a amostra de treinamento atual e uma representação das amostras anteriores. Desse modo, a variação LS-SVM tende a ser mais indicada para ser empregada junto ao modo de aprendizado *online*, ao passo que a metodologia SVM tradicional seria mais orientada para aprendizado em batelada. Uma desvantagem dessa extensão é que os multiplicadores de Lagrange tendem a ser todos não-nulos, carregando problemas de esparsividade nas matrizes de programação quadrática. Contudo, já existem novas versões de LS-SVMs que tentam eliminar muitos dos vetores-suporte menos relevantes de tal forma a se recuperar representações mais esparsas [446].

Para problemas que envolvam mais de duas classes, existem basicamente dois tipos de extensões para SVMs. Um grupo de abordagens (conhecidas como *métodos de decomposição*) segue a estratégia de se considerar as amostras referentes a todas as classes em um só problema de otimização, implicando, como consequência, o dispêndio de recursos computacionais em proporção ao número de classes. O segundo grupo de abordagens segue a filosofia de se construir, em uma primeira fase, vários classificadores binários para, então, combiná-los, de tal modo a se gerar uma única solução. Dentre tais extensões, destacam-se: (i) “um-contratodos”, em que a p -ésima instância de SVM fica responsável por discriminar as amostras pertencentes a p -ésima classe (ou seja, com rótulo “+1”) das demais (rótulo “-1”); (ii) “um-contrum”, em que, admitindo-se M classes, serão necessários $\frac{M \times (M-1)}{2}$ classificadores SVM, sendo que o pq -ésimo classificador fica responsável por distinguir as amostras da p -ésima classe daquelas pertencentes a q -ésima classe; e (iii) “grafo acíclico direcionado”, que corresponde a uma variação hierárquica do método anterior, com estrutura em árvore (cada nó interno corresponde a um classificador). Hsu e Lin [211] provêem um primeiro esforço prospectivo de comparação entre tais métodos, tendo em mente problemas não-triviais. Os experimentos reportados sugerem que os métodos baseados em classificadores binários, notadamente os de índices (i) e (iii) acima, são mais adequados para uso prático (eficiência *versus* eficácia).

Finalmente, é conveniente se ressaltar a importância por trás das funções-*kernel* tendo em vista a aplicabilidade de SVMs. A escolha de tais funções e de seus parâmetros associados tem influência direta sobre a forma funcional da superfície de decisão gerada e, por conseguinte, sobre o desempenho final da máquina (ver Fig. 3.2). Ratificando essa posição, já existem trabalhos teóricos [414] analisando a potencialidade de se derivar generalizações de algoritmos já consagrados para fins de caracterização de novos *kernels* — como exemplo, Schölkopf *et al.* [415] propuseram *kernel PCA* como uma extensão não-linear para análise de componentes principais. Há também trabalhos de cunho mais heurístico que tentam estipular dinamicamente os parâmetros dos *kernels*, de acordo com as nuances do problema em questão; encaixa-se aqui a abordagem *Kernel-Adatron*, citada anteriormente, que foi empregada com êxito, juntamente com técnicas de seleção de modelos [246], na adaptação dos parâmetros da função-*kernel* ao longo da fase de treinamento da máquina [404].

3.3 Observações finais

Neste capítulo, foi apresentada uma visão geral sobre a metodologia SVM, evidenciando: (i) sua fundamentação na teoria de aprendizado estatístico; (ii) aspectos teóricos relacionados ao processo de treinamento via programação quadrática, tanto para problemas de classificação como de regressão; (iii) a necessidade de configuração dos parâmetros de controle (μ e ϵ) e daqueles associados às funções-*kernel*; (iv) uma perspectiva modular da disposição estrutural de uma instância de SVM; e (v) alguns dos tópicos de estudo mais recentes (eficiência computacional do processo de otimização quadrática e problemas envolvendo múltiplas classes). O material apresentado aqui não pretende contemplar todos os aspectos vinculados ao tema, haja vista já existirem referências mais completas e atualizadas com esse propósito [100, 415, 462, 463], além de outras envolvendo extensões bem interessantes — p. ex., Semolini [418] conduziu vários experimentos de classificação, inclusive na área de bioinformática, combinando SVMs com técnicas de aprendizado transdutivo, em concordância com a abordagem proposta anteriormente por Joachims [235]. O objetivo principal aqui foi o de propiciar uma embasamento teórico para o projeto apresentado na Seção 8.2.

Com efeito, como contribuição ao estudo do papel exercido pelas funções-*kernel* sobre o desempenho final da máquina resultante, Lima *et al.* [284, 287, 289], seguindo uma linha de investigação mais empírica [78, 79], introduziram uma nova abordagem, baseada em comitês de máquinas (ver próximo capítulo), que é voltada para a seleção automática de diferentes instâncias de SVM atuando paralelamente sobre o mesmo problema. Trata-se de uma abordagem de cunho eminentemente *cooperativo*, já que estimula a combinação das funções-*kernel* mais adequadas à natureza da tarefa em questão. O artigo apresentado na Seção 8.2 estende essa abordagem na medida em que o processo de seleção e combinação das instâncias de SVM (e seus respectivos *kernels*) passa a ficar a cargo de um algoritmo evolutivo projetado especificamente para esse fim.

Capítulo 4

Comitês de máquinas

“As nossas necessidades nos unem, mas as nossas opiniões nos separam.”

Marquês de Maricá

Nos capítulos anteriores, foram descritas duas abordagens proeminentes dirigidas ao aprendizado de máquina supervisionado. Enquanto algoritmos de indução de RNAs *feedforward* (Cap. 2) dependem diretamente de ferramentas de otimização global e não-linear para fins de geração dos modelos neurais resultantes, a metodologia SVM (Cap. 3) é embasada fortemente em aspectos e definições formais, tais como dimensão VC e minimização do risco estrutural, advindos da teoria do aprendizado estatístico. Apesar de seus inúmeros pontos favoráveis, tais abordagens também apresentam algumas dificuldades inerentes. Como exemplo, o processo de otimização de MLPs tem, muitas vezes, sua eficácia comprometida por problemas relacionados à existência de vários mínimos locais ao longo da superfície de erro [318, 386], ao passo que o desempenho de máquinas de vetores-suporte tende a ser influenciado sobremaneira pela escolha de parâmetros de controle (tais como, constante de regularização e tamanho do tubo) e de parâmetros associados às funções-*kernel* adotadas [78, 186, 463].

Em vez de se fixar no estudo de estimadores operando isoladamente, como feito anteriormente, o conteúdo deste capítulo versa sobre diferentes maneiras de se compor vários desses estimadores — denominados *especialistas* e oriundos, provavelmente, de distintos métodos de aprendizado supervisionado — em um só arranjo estrutural. Uma primeira motivação por trás dessa estratégia de se trabalhar com *sistemas multiestimadores* (também chamados de *Comitês de Máquinas* [452]) é a de justamente eliminar a necessidade de se ter que configurar acertadamente parâmetros de controle associados aos indutores ou se estipular previamente qual deva ser a dimensão do modelo resultante. Outra forte justificativa é que os módulos que compõem tais sistemas têm propensão a apresentar menor complexidade, facilitando o seu entendimento por parte do projetista, bem como uma possível modificação ou extensão *a posteriori* [222, 425]. Um terceiro aspecto que também dá razão ao estudo e aplicação de comitês de máquinas está no aumento considerável da disponibilidade de ambientes computacionais mais potentes (como aqueles construídos sobre plataformas

distribuídas), o que permite a implementação e experimentação, cada vez mais frequentes, de arquiteturas compostas por múltiplos módulos de aprendizado executando em paralelo [461]. Ademais, a possibilidade de se ter dois ou mais especialistas (estes possuindo maior granulosidade quando comparados a neurônios ou funções-base) trabalhando juntos para a resolução de tarefas complicadas constitui uma alternativa atraente em se tratando do quesito de tolerância a falhas, na medida em que módulos corrompidos podem ser substituídos ou descartados sem prejuízo muito crítico para o sistema como um todo [27, 423].

Outro motivo, de maior apelo, diz respeito à possibilidade de se obter ganhos substanciais em termos da capacidade de *generalização* a ser exibida pela máquina agregada resultante [25, 191, 197, 297, 426]. Nesse contexto, o emprego com êxito de instâncias simples de preditores (p. ex., RNAs ou mesmo SVMs) parte do pressuposto de que tais modelos consigam capturar e processar satisfatoriamente todas as nuances e características do problema-alvo, independentemente de quais elas sejam. Para os casos em que tal suposição venha a falhar, por exemplo, quando os atributos de entrada passam a consistir de uma plethora de variáveis lingüísticas ou sintáticas, de natureza contínua ou discreta, incorrendo em problemas de “maldição da dimensionalidade” [130], recorrer às habilidades de um único preditor, por mais elaborado que este seja, pode não ser a estratégia mais adequada. Isso porque modelos sobredimensionados apresentam alto risco de produzirem mapeamentos sobreajustados aos dados, como discutido na Seção 2.5. Nesse sentido, extrair as diferentes habilidades de diferentes especialistas, cada um vendo o problema sob uma ótica particular, vem se mostrando, teórica e experimentalmente, ser uma opção de projeto mais atraente. Em geral, contrastando-se arquiteturas multimodulares (que retêm a capacidade de aproximação universal de funções) com aquelas de formato monolítico, verifica-se que as primeiras têm tendência a requerer um número mais reduzido de parâmetros livres (p. ex., número de nós e pesos em RNAs) a serem ajustados de modo a que se atinja o mesmo grau de acuidade (taxa de acerto) sobre dados de treinamento ou teste [198]. Tal fato vem a corroborar, portanto, a expectativa de que comitês de máquinas têm maior inclinação a exibir uma maior capacidade de generalização frente à metodologia tradicional de se usar o melhor modelo preditivo em mãos.

Por ser uma área de pesquisa relativamente recente em AM⁽¹⁾, não existe ainda um consenso definitivo com relação à melhor taxonomia a ser adotada para o tema [262, 461]. Todavia, é muito comum serem discriminadas na literatura duas classes de abordagens complementares [174, 283, 426]. A primeira classe, conhecida pela alcunha de *ensembles de estimadores* [123, 129, 191, 256, 297], segue a lógica de se *fundir* apropriadamente (via um módulo especial chamado de *combinador*) o conhecimento adquirido pelos vários módulos-especialistas, denominados *componentes*, para se chegar a uma decisão geral que seja supostamente de qualidade superior àquela alcançável por qualquer um dos membros do grupo em separado. A outra classe de abordagens assume o cognome genérico de *sistemas modulares* [25, 269, 433] e lança mão mais explicitamente do preceito de “dividir para conquistar” introduzido no início do Cap. 1. Desse modo, sob o prisma de sistemas modulares, uma tarefa complexa passa a ser decomposta em um certo número de subtarefas, cada uma delas alocada a um subgrupo de especialistas. Isso leva a exigência de que, para se

⁽¹⁾Embora se possa traçar um retrospecto desde os anos 1950/1960, com a divulgação da arquitetura Pandemonium por Selfridge [417] e da abordagem de combinação de Perceptrons de Nilsson [344], os primeiros trabalhos justificando, mais sistematicamente e com maior rigor formal, a idéia de combinação ou modularização de máquinas de aprendizado só começaram a despontar entre as décadas de 1980 e 1990 [49, 191, 222, 254, 384, 412, 497].

gerar a solução final, será necessário que se produza, primeiramente, as soluções para todas as subtarefas. Um exemplo de modelo hierárquico estipulando pelo particionamento dos módulos neurais de acordo com o rótulo da classe — i. e., caso se tenha M classes, deve-se empregar M especialistas, cada um treinado mediante o subconjunto de amostras relativo à sua classe — é aquele proposto por Kung & Taur, intitulado de DBNN (de *Decision-Based Neural Networks*) [268]. Uma outra forma muito comum de se “modularizar” a arquitetura do sistema é a de se definir, dinamicamente, regiões do espaço de entradas a serem atribuídas aos especialistas. Sendo o caso, a decisão do grupo para uma certa amostra será produzida por aquele(s) especialista(s) alocado(s) à região a que ela pertence. Essa estratégia de particionamento é adotada, por exemplo, pela abordagem de cunho estatístico conhecida como *mistura de especialistas* (ME) [226, 239] e pela abordagem CMNN (de *Cooperative Modular Neural Networks*), concebida por Auda & Kamel [25]. Essa última se distingue da primeira na medida em que a decisão sobre a seleção do especialista mais apto para lidar com a amostra corrente é feita pelos próprios especialistas, via um esquema de votação, e não por um único módulo à parte (rede *gating*).

Haykin [198] fornece uma classificação alternativa, diferenciando entre estruturas *estáticas* e *dinâmicas*. Para arquiteturas estáticas, as respostas de diversos especialistas devem ser combinadas por um mecanismo específico que *não* tem acesso aos sinais de entrada do sistema. O autor inclui nesse grupo os variados modelos de *ensembles* e os métodos estatísticos de reamostragem adaptativa de dados — o exemplo citado foi *Boosting* [160, 412]. Por outro lado, no que concerne às estruturas dinâmicas, o módulo de integração das saídas dos especialistas passa a fazer uso da informação de entrada de maneira a que possa selecionar os especialistas mais aptos para tratarem a entrada corrente e gerarem, conseqüentemente, a saída final do sistema. Incluídas aqui, pelo autor, foram a arquitetura ME e sua versão hierárquica. Essa categorização, em sua essência, não parece ser de todo universal, haja vista que não contempla outros tipos de sistemas modulares, ditos desacoplados [26], que não lançam mão de mecanismos de combinação das saídas dos especialistas — exemplos típicos encerram algumas versões do *framework* Pandemonium exploradas por Śmieja [433], em que o especialista (visto como agente) que apresentar o maior *grau de confiança* em relação à amostra atual será o escolhido, a abordagem DBNN [268] e os diferentes métodos de “hibridização e especialização” endereçados por Catfolis & Meert [73].

Um outro roteiro de classificação foi apresentado por Jain *et al.* [227], segundo o qual três categorias de arquiteturas multimodulares (em particular, multiclassificadores) podem ser depreendidas. Em estruturas *paralelas*, todos os preditores são invocados independentemente, sendo suas decisões combinadas por algum módulo destacado. Alguns métodos de *ensembles* como também os sistemas modulares descritos anteriormente passam a figurar nesse contexto. Por sua vez, em estruturas *multiestágios* (ou *em cascata*), os módulos passam a ser invocados em seqüência; o número de classes-candidatas para rotulagem de um dado padrão passa a ser gradualmente reduzido à medida que mais classificadores são convocados [205]. Caso se almeje eficiência computacional, deve-se recorrer primeiramente a preditores menos precisos mas com menor demanda por recursos, para depois lançar mão daqueles mais precisos mas com maior demanda por recursos. Finalmente, estruturas *hierárquicas* assumem a disposição em árvore, cujos nós passam a ser associados a estimadores treinados independentemente, algo interessante quando a tarefa de classificação/regressão dispõe

de um grande número de atributos — a versão hierárquica de MEs [239] pode se encaixar nessa classe.

Outros autores não fazem muita distinção entre as várias abordagens multiestimadores. Como exemplo, Kuncheva [262] e Valentini & Masulli [461] consideram os diversos arranjos possíveis, inclusive os sistemas modulares acima mencionados, como variações de *ensembles*. Por outro lado, Auda & Kamel [27] vêem *ensembles* de RNAs como um caso particular de sistemas conexionistas modulares. A abordagem EVOL [26], proposta pelos mesmos autores, ratifica essa posição, visto que emprega um modelo de votação *online* para *ensembles* de classificadores com o propósito de se conseguir, em realidade, a decomposição automática do espaço de características.

Um esforço mais consistente em caracterizar os inúmeros arranjos topológicos factíveis de sistemas multiestimadores foi empreendido recentemente por Sharkey, tendo como foco modelos conexionistas [427, 428]. Além da distinção entre *ensembles* e sistemas modulares, um segundo eixo de categorização promove a diferenciação entre arranjos *competitivos* e *cooperativos*. Em agregações da primeira estirpe, o objetivo se traduz em identificar o módulo mais apropriado para lidar com o caso atual, passando-lhe a responsabilidade por gerar a saída do grupo. Arranjos competitivos podem abranger tanto elementos típicos de *ensembles* como de sistemas modulares, dependendo se tais elementos representam ou não decomposições da tarefa. Em agregações cooperativas, por sua vez, as saídas produzidas por diferentes módulos são sempre combinadas, com ou sem ponderação. Arranjos dessa natureza são exemplificados tipicamente por *ensembles*, embora existam *frameworks* de sistemas modulares que também possam se enquadrar nesse contexto — a abordagem CMNN [25] é um caso. Uma terceira dimensão de categorização do esquema taxonômico resultante discrimina entre sistemas *bottom-up* e *top-down*, ou seja, entre sistemas que levam em conta as saídas dos módulos para a sua combinação e aqueles que não o fazem. Assim posto, estruturas cooperativas são inerentemente *bottom-up*, ao passo que o mesmo não pode ser dito sobre os arranjos competitivos — como exemplo, a abordagem ME é de natureza *top-down*, pois a seleção dos especialistas baseia-se em outra informação que não seja a saída dos módulos (no caso, o próprio sinal de entrada), enquanto a abordagem competitiva proposta por Baxt [43] é *bottom-up*, já que estipula que a saída do especialista selecionado só irá realmente ser interpretada como a saída do sistema caso ultrapasse um limiar empírico preestabelecido.

Independentemente do plano de categorização a ser adotado, o fato é que tanto estruturas típicas de *ensembles* como aquelas típicas de sistemas modulares não precisam ser pensadas como mutuamente exclusivas, podendo, pois, ser “hibridizadas” de forma a se gerar arquiteturas ainda mais complexas. A Fig. 4.1 traz uma disposição arquitetural hipotética compreendendo o uso concomitante de sistemas modulares e *ensembles* em diferentes níveis de resolução. No patamar superior, o *framework* consiste de uma combinação em *ensemble* de três componentes. No nível de sub tarefa, por sua vez, um dos componentes passa a encapsular, em realidade, um sistema modular de especialistas distintos. As soluções (decisões) produzidas pelos três componentes são geradas de maneiras dessemelhantes. A primeira solução é obtida com base em apenas um dos sensores. A segunda é computada a partir da combinação cooperativa da saída de três sensores, ao tempo que a terceira é fruto de uma associação modular baseada em decomposição por rótulos das classes — cada classe associada a um especialista, como estipulado pela abordagem DBNN. Não obstante esse esboço

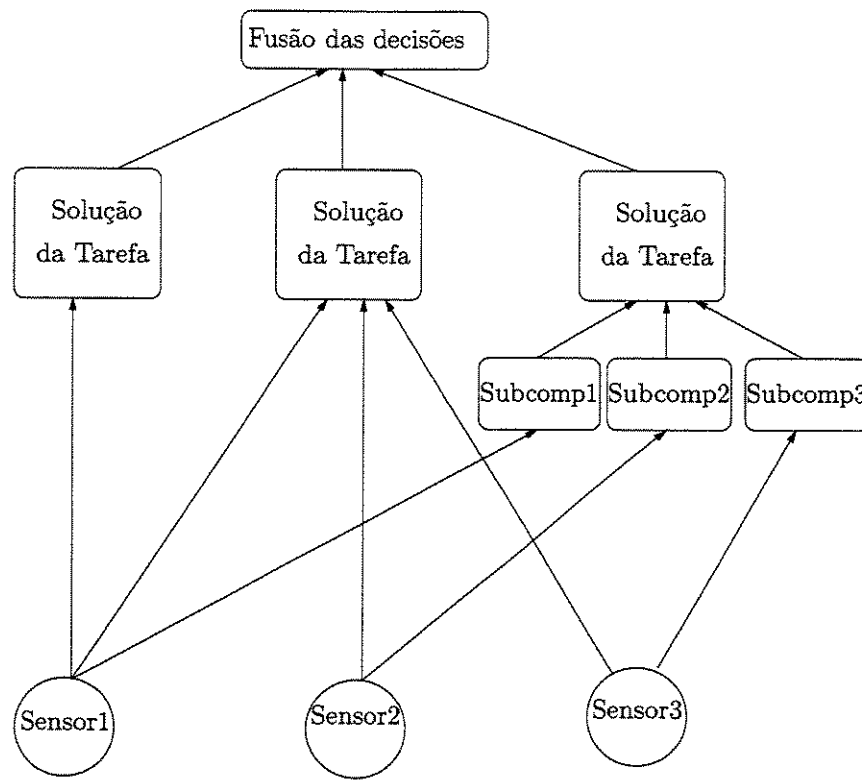


Fig. 4.1: Um exemplo de arquitetura multimodular híbrida lançando mão de *ensembles* e sistemas modulares em diferentes níveis de resolução. (Fonte: Sharkey [427].)

assuma a existência de três sensores díspares, um arranjo similar poderia ser concebido tomando por base um particionamento equivalente de um conjunto de dados qualquer ou a extração de diferentes atributos de tais dados [427].

À luz dessa exposição inicial, que transmite justificativas para o estudo do tema e tenta categorizar as inúmeras propostas que se inserem nesse *metiê*, passa-se na sequência a expor os pontos-chave por trás do paradigma de *ensembles* de estimadores (Seção 4.1) e as principais propriedades exibidas pela arquitetura de misturas de especialistas (Seção 4.2), tendo em mente se prover um ferramental teórico para os projetos relatados nos Caps. 7 e 8 deste documento.

4.1 *Ensembles* de estimadores

De um modo sintético, pode-se dizer que o termo “*ensemble*” faz referência aos diferentes métodos de construção, seleção e combinação de máquinas de aprendizado que geram soluções *redundantes* para a mesma tarefa, sendo tais máquinas geralmente configuradas sob condições dissimilares. Como apontado

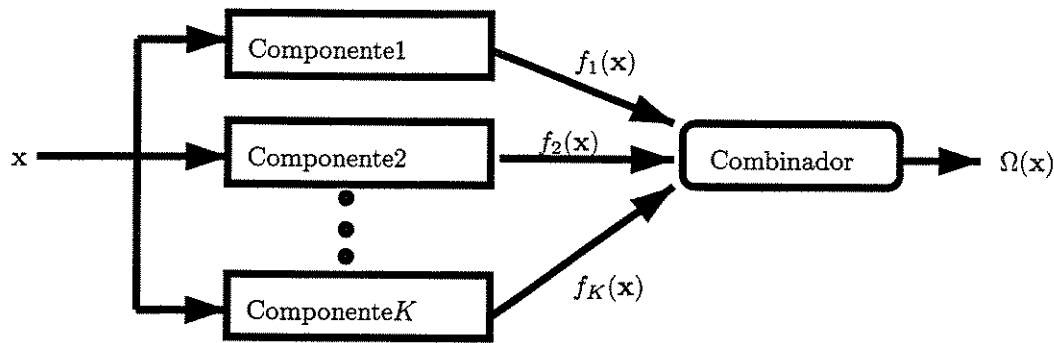


Fig. 4.2: Diagrama em blocos de um modelo típico de *ensemble* com K componentes. As saídas dos estimadores selecionados são agregadas usando um algoritmo de combinação adequado para gerar a saída do sistema.

por Sharkey [427], “a combinação de um conjunto de estimadores imperfeitos pode ser pensada como uma maneira [produtiva] de se administrar as suas reconhecidas limitações; é sabido que cada componente comete erros, mas eles todos são agregados de tal modo a se minimizar os efeitos desses erros”. A Fig. 4.2 traz um esboço de um arranjo típico de *ensemble*, evidenciando a existência do módulo combinador e o fato de que todos os K componentes (estes já selecionados, provavelmente de um universo maior) têm que arcar com o mesmo mapeamento de Entrada/Saída (E/S).

A idéia por si só de combinar dois ou mais estimadores (ou tomadores de decisão) com o intento de se obter ganhos sensíveis em termos de desempenho (segundo um critério qualquer) não é recente. Por exemplo, o modelo de júri Condorcet [114] foi inventado no final do séc. XVIII para estudar sob que condições uma democracia como um todo seria mais efetiva que qualquer uma de suas partes constituintes. Já na área de tomada de decisão estatística, foram desenvolvidos, em meados do século passado, vários métodos relacionados à teoria de consenso [49], dentre os quais se destacam aqueles dedicados a gerar uma distribuição de probabilidade única que reflita as múltiplas estimativas fornecidas por vários especialistas bayesianos. Ademais, na área de previsão de séries econométricas, tem sido cada vez mais advogada a tese de que melhores resultados podem ser alcançados mediante a combinação das previsões feitas por diferentes técnicas [83, 181]. Finalmente, na disciplina de Engenharia de *Software*, é comum se lançar mão de múltiplas versões redundantes do mesmo código com o intuito de se atingir níveis mais seguros de tolerância a falhas [134].

Há um número razoável de situações em que o uso de uma *ensemble* de estimadores seja justificável. Por exemplo, quando se pretende selecionar um subconjunto de atributos de entrada que melhor represente um dado problema [283]. Em geral, a principal motivação está na tentativa de se eliminar as deficiências de cada componente, de modo a que o grupo alcance níveis mais elevados de acuidade e generalização. Ratificando essa perspectiva, é pertinente mencionar que [123, 227, 461]:

- Normalmente, algoritmos de aprendizado tentam induzir a melhor hipótese (i. e., estimador) a partir

de um espaço de hipóteses $\mathcal{H}$ limitado, posto que, na maioria das vezes, somente poucas amostras estão disponíveis. Nesses casos, o indutor pode acabar encontrando hipóteses dessemelhantes exibindo, aparentemente, o mesmo grau de precisão sobre os dados da função-geratriz F_g que se quer modelar. Em vez de ter que se recorrer a critérios extras (tais como aqueles fundamentados no princípio MDL — ver Subseção 2.4.2) para se julgar qual hipótese escolher, parece ser mais fortuito aplicar algum método de combinação (por mais simples que este seja) sobre as várias hipóteses equipotentes encontradas de modo a se confeccionar uma aproximação mais adequada para F_g .

- Em muitos casos, a melhor hipótese para a função desconhecida F_g não está presente em $\mathcal{H}$. Porém, uma combinação de várias hipóteses extraídas de $\mathcal{H}$ pode *expandir* o espaço de funções representáveis pelo indutor, abarcando, inclusive, F_g .
- Outras vezes, mais de um conjunto de treinamento está disponível, cada um deles coletado em um tempo distinto ou por sensores espacialmente distribuídos. Outrossim, tais conjuntos de dados podem fazer uso de diferentes atributos ou representações (descrições) para o mesmo problema. Em vez de ter que reunir todos os dados para aplicar o indutor de uma só vez (modo em lote), pode ser mais eficiente (e, às vezes, necessário) gerar novas hipóteses *online*, sendo elas progressivamente agregadas ao sistema.
- No lugar de se descartar os vários modelos produzidos em favor de um só (“vencedor”), parece ser mais sensato, em termos de custo computacional, tentar *reusar* as habilidades de tais modelos (por menores que estas sejam) como suporte (*backup*) para as decisões do modelo vencedor.

Os vários esquemas possíveis de combinação de estimadores existentes na literatura [123, 262, 396, 426, 461] diferem basicamente entre si com relação às suas arquiteturas, às características de seus combinadores e ao modo como produzem⁽²⁾ e selecionam seus componentes. Embora muitos desses esquemas já tenham sua eficácia comprovada experimentalmente, Roli *et al.* [396] ressaltam que ainda *não* há à disposição dos projetistas receitas teóricas (ou heurísticas) claras e consistentes indicando como escolher o melhor esquema tendo em mente os requisitos do problema-alvo. De acordo com os autores, só se pode determinar o melhor esquema mediante experimentação sistemática e análise comparativa de desempenho⁽³⁾.

De todo modo, para se avaliar as vantagens de diferentes métodos de *ensemble*, torna-se relevante estabelecer critérios para qualificar as suas propriedades. Para tanto, Kuncheva & Jain [266] anunciaram seis aspectos-chave a serem adotados para caracterizar um arranjo de combinação qualquer: (i) o *número de estimadores* usados (já selecionados); (ii) a *natureza de cada componente* — a maioria dos esquemas utilizam estimadores provenientes de um mesmo indutor (RNAs, SVMs, árvores de decisão, etc.) [23, 129, 251, 297, 343, 425], ao passo que outros esquemas empregam estimadores oriundos de diferentes algoritmos de indução [162, 205, 403, 432, 494]; (iii) os *subconjuntos de atributos* (com ou sem sobreposição) usados

⁽²⁾Incluso está nesse quesito o modo como tais esquemas manipulam os dados de treinamento.

⁽³⁾Tal colocação estabelece uma forte prerrogativa para que sejam consideradas técnicas alternativas destinadas ao projeto automático de esquemas de combinação de estimadores, tais como aquelas calcadas no uso de algoritmos evolutivos (ver Seções 6.9 e 8.2, e Cap. 7).

para treinar os componentes — alguns esquemas treinam os preditores sobre todo o espaço de entradas, ao tempo que outros atribuem dimensões distintas a componentes distintos [259, 351]; (iv) os *subconjuntos de dados amostrais* (com ou sem sobreposição) usados para treinar os componentes — alguns esquemas treinam os preditores sobre todas as amostras de treinamento disponíveis, ao tempo que outros atribuem amostras distintas a componentes distintos [63, 129, 213]; (v) o *algoritmo de combinação* adotado para se coletar as saídas dos componentes (possivelmente, sob diferentes formatos de representação [9]) e gerar a decisão final do grupo [32, 40, 492, 497]; e (vi) a *estratégia de treinamento* aplicada a todo o grupo — indo desde o treinamento independente dos componentes e posterior aplicação do combinador, o qual, por sua vez, não requer nenhum ajuste [23, 61, 129, 191], até o treinamento simultâneo de todo o grupo [14, 26], passando pela alternativa de se treinar os componentes e, em seguida, o combinador [492].

Independentemente da abordagem de combinação que venha a adotar, o projetista deve atentar para o seguinte fato: uma vez que o principal motivo em se combinar estimadores está no incremento do desempenho da máquina resultante, *não* há vantagem alguma a ser ganha quando se lança mão de estimadores *idênticos*, ou seja, que apresentam o mesmo *padrão de generalização* [297, 354, 369, 397, 423]. Em princípio, várias máquinas podem divergir em termos de seus parâmetros livres, do tempo que levaram até a sua convergência, e de suas arquiteturas ou metodologias de aprendizado, e ainda assim constituírem essencialmente a *mesma solução*, uma vez que apresentam o mesmo padrão (distribuição) de erros quando avaliadas sob o mesmo conjunto de teste. Ademais, como salientado por Rogova [395], o melhor desempenho para uma determinada *ensemble* *não* é necessariamente obtido mediante a agregação dos estimadores com melhores desempenhos individuais dentre aqueles disponíveis. Como decorrência disso, o objetivo de qualquer procedimento de geração ou seleção dos membros de uma *ensemble* deve ser o de assegurar, de uma forma ou de outra, que tais membros sejam mais dissimilares entre si quanto o possível.

No que segue, alguns resultados teóricos legitimando o emprego de *ensembles* são brevemente introduzidos. Posteriormente, retoma-se a discussão sobre o requisito da diversidade dos padrões de falhas dos componentes, apontando diferentes maneiras de tentar avaliá-lo, mensurá-lo ou induzi-lo de antemão. Por fim, passa-se a comentar um pouco mais sobre aspectos relacionados às operações de criação, seleção e combinação de estimadores, tendo em vista a produção de *ensembles* efetivas. Nesse contexto, são apresentados, de maneira condensada, alguns desenvolvimentos matemáticos e técnicas relevantes que serviram de suporte teórico e de base de comparação para as abordagens COHENN (Cap. 7) e GASEL (Seção 8.2). O material anunciado nesta seção é complementado diretamente por aquele apresentado no Apêndice B, onde são descritos vários operadores de combinação de estimadores dirigidos a tarefas de classificação de padrões.

4.1.1 Justificativas teóricas

Muitas análises experimentais (a maioria delas centrada na tarefa de classificação de padrões) vêm confirmando a expectativa de que a estratégia de se combinar estimadores tem propensão a incorrer em ganhos substanciais em termos da taxa de reconhecimento final [41, 302, 306, 352]. Contudo, exis-

tem, proporcionalmente, poucas explanações teóricas justificando formalmente o porquê desses resultados empíricos [63, 197, 254, 384, 411]. Algumas delas são descritas abaixo.

A mais conhecida dessas explanações decorre diretamente do dilema Polarização-Variância [171], o qual estabelece uma decomposição clara do erro de estimação (classificação ou regressão) de qualquer estimador em termos da sua polarização e da sua variância aos dados. Como discutido na Seção 2.5, o aumento da capacidade de generalização de uma hipótese requer um balanço apropriado entre os requisitos conflitantes de menor variância e menor polarização. Isso se traduz numa relação de compromisso, visto que tentativas em se reduzir um dos fatores levam, inevitavelmente, ao incremento do outro. Vários exercícios matemáticos em torno da DPV vêm sendo realizados para *ensembles* sob diferentes perspectivas [61, 62, 123, 197, 256], embora as conclusões tiradas sejam similares ou complementares. A análise descrita a seguir baseia-se naquela exibida por Haykin [198] e originalmente proposta por Naftaly *et al.* [343], a qual considera a situação em que os especialistas (no caso, RNAs) têm configurações topológicas idênticas mas são treinados a partir de condições iniciais distintas para os seus parâmetros livres. Adota-se o operador “média simples” (MEB), como concebido por Perrone & Cooper [369] (ver à frente), para o papel de combinador, conquanto qualquer outro operador poderia ser usado.

Seja $\mathcal{I}$ o espaço incluindo todas as condições iniciais possíveis e seja $\Omega_I(\mathbf{x})$ a média dos K mapeamentos de E/S relativos aos especialistas da Fig. 4.2 sobre um número “representativo” de condições iniciais. Da definição de $\mathcal{D}$ dada na Seção 2.5, pode-se considerar esse espaço como sendo o produto entre $\mathcal{I}$ e um espaço residual, indicado por $\mathcal{D}^*$. Analogamente ao que foi feito anteriormente para $\mathcal{D}$ (ver Eq. (2.10)), pode-se escrever a seguinte decomposição para $\mathcal{I}$, omitindo-se a possível influência do conjunto de dados D_N :

$$E_{\mathcal{I}}[(\Omega_I(\mathbf{x}) - E(y_d|\mathbf{x}))^2] = B_{\mathcal{I}}(f(\mathbf{x})) + V_{\mathcal{I}}(f(\mathbf{x})),$$

onde $B_{\mathcal{I}}(f(\mathbf{x}))$ e $V_{\mathcal{I}}(f(\mathbf{x}))$ denotam, respectivamente, o quadrado da polarização sobre o espaço $\mathcal{I}$ e a sua variância correspondente, sendo dados por⁽⁴⁾:

$$B_{\mathcal{I}}(f(\mathbf{x})) = \{E_{\mathcal{I}}[\Omega_I(\mathbf{x})] - E(y_d|\mathbf{x})\}^2 \quad (4.1)$$

e

$$V_{\mathcal{I}}(f(\mathbf{x})) = E_{\mathcal{I}}\{(\Omega_I(\mathbf{x}) - E_{\mathcal{I}}[f(\mathbf{x})])^2\}. \quad (4.2)$$

Nesse desenvolvimento, $f(\cdot)$ indica um mapeamento genérico representando qualquer um dos K mapeamentos-componentes. Da mesma forma, pode-se escrever a DPV para o espaço residual $\mathcal{D}^*$:

$$E_{\mathcal{D}^*}[(\Omega_I(\mathbf{x}) - E(y_d|\mathbf{x}))^2] = B_{\mathcal{D}^*}(\Omega_I(\mathbf{x})) + V_{\mathcal{D}^*}(\Omega_I(\mathbf{x})),$$

onde $B_{\mathcal{D}^*}(\Omega_I(\mathbf{x}))$ e $V_{\mathcal{D}^*}(\Omega_I(\mathbf{x}))$ denotam, respectivamente, o quadrado da polarização sobre o espaço $\mathcal{D}^*$ e a sua variância correspondente, sendo, por sua vez, definidos como:

$$B_{\mathcal{D}^*}(\Omega_I(\mathbf{x})) = \{E_{\mathcal{D}^*}[\Omega_I(\mathbf{x})] - E(y_d|\mathbf{x})\}^2 \quad (4.3)$$

⁽⁴⁾O operador esperança $E_{\mathcal{I}}$ é definido sobre o espaço $\mathcal{I}$, enquanto a esperança $E_{\mathcal{D}^*}$ é definida sobre o espaço $\mathcal{D}^*$.

e

$$V_{\mathcal{D}^*}(\Omega_I(\mathbf{x})) = E_{\mathcal{D}^*}\{(\Omega_I(\mathbf{x}) - E_{\mathcal{D}^*}[\Omega_I(\mathbf{x})])^2\}. \quad (4.4)$$

A passagem mais importante nesse desenvolvimento é perceber que $E_{\mathcal{D}^*}[\Omega_I(\mathbf{x})] = E_{\mathcal{D}}[f(\mathbf{x})]$. Desse fato, pode-se reescrever (4.3) numa forma equivalente:

$$B_{\mathcal{D}^*}(\Omega_I(\mathbf{x})) = \{E_{\mathcal{D}}[f(\mathbf{x})] - E(y_d|\mathbf{x})\}^2 = B_{\mathcal{D}}(f(\mathbf{x})). \quad (4.5)$$

Por outro lado, uma vez que a variância de uma variável aleatória é igual ao valor quadrático médio dessa variável menos o quadrado da sua polarização, pode-se reescrever (4.4) como:

$$\begin{aligned} V_{\mathcal{D}^*}(\Omega_I(\mathbf{x})) &= E_{\mathcal{D}^*}[(\Omega_I(\mathbf{x}))^2] - \underbrace{\{E_{\mathcal{D}^*}[\Omega_I(\mathbf{x})]\}^2}_{\{E_{\mathcal{D}}[f(\mathbf{x})]\}^2} \\ &= E_{\mathcal{D}^*}[(\Omega_I(\mathbf{x}))^2] - \{E_{\mathcal{D}}[f(\mathbf{x})]\}^2. \end{aligned} \quad (4.6)$$

O mesmo artifício pode ser empregado sobre a segunda parcela (variância) da Eq. (2.10), o que produz:

$$V_{\mathcal{D}}(f(\mathbf{x})) = E_{\mathcal{D}}[(f(\mathbf{x}))^2] - \{E_{\mathcal{D}}[f(\mathbf{x})]\}^2. \quad (4.7)$$

Uma observação muito pertinente aqui é a de que o valor quadrático médio do mapeamento $f(\mathbf{x})$ sobre todo o espaço $\mathcal{D}$ é destinado a ser maior ou igual que o valor quadrático médio da média dos mapeamentos $(\Omega_I(\mathbf{x}))$ sobre o espaço residual $\mathcal{D}^*$. Formalmente:

$$E_{\mathcal{D}}[(f(\mathbf{x}))^2] \geq E_{\mathcal{D}^*}[(\Omega_I(\mathbf{x}))^2].$$

Com base nessa desigualdade, uma comparação entre (4.6) e (4.7) revela que

$$V_{\mathcal{D}^*}(\Omega_I(\mathbf{x})) \leq V_{\mathcal{D}}(f(\mathbf{x})). \quad (4.8)$$

Finalmente, a partir das Eqs. (4.5) e (4.8), decorre que (1) a polarização da *ensemble* (i. e., média dos mapeamentos, $\Omega_I(\mathbf{x})$) é *exatamente igual* àquela relativa a qualquer um dos mapeamentos-componentes ($f(\mathbf{x})$); e (2) a variância de $\Omega_I(\mathbf{x})$ é *menor* que aquela de $f(\mathbf{x})$. Tal resultado teórico aponta para o fato de que o aumento de desempenho que se pode obter com combinações em *ensemble* deve ser fruto direto de uma redução na variância em vez de uma redução na polarização⁽⁵⁾. Destarte, uma abordagem efetiva — quer seja baseada na variação das condições iniciais, quer seja baseada em outros fatores, como tamanho da arquitetura ou algoritmo de indução — é a de criar e/ou selecionar um conjunto de estimadores que exibam alta variância e pouca polarização, haja vista que a variância pode ser reduzida como efeito da combinação.

Dando suporte à premissa de que não há ganho algum em se combinar modelos apresentando comportamentos de falhas coincidentes, Krogh & Vedelsby [256] lançaram mão de uma DPV alternativa para

⁽⁵⁾ Conquanto tenha sido evidenciado também que algumas formas de empilhamento de estimadores (ver Subseção 4.1.5) e algumas versões de *Boosting* (ver Subseção 4.1.3) podem induzir à redução de polarização [283, 492].

demonstrar, formalmente, que o erro de generalização de uma *ensemble* (E_{ens}) pode ser visualizado como a diferença entre duas parcelas distintas: um termo medindo a média ponderada dos erros de generalização dos membros do grupo ($\bar{E}$) e um termo indicando a média ponderada das ambigüidades relativas aos membros ($\bar{A}$), esse último denominado *ambigüidade da ensemble*⁽⁶⁾. De uma forma matemática:

$$E_{ens} = \bar{E} - \bar{A}. \quad (4.9)$$

A equação acima deve ser interpretada da seguinte forma [256]: se a *ensemble* está fortemente polarizada, então a sua ambigüidade (i. e., diversidade) será pequena porque os especialistas implementam mapeamentos bastante similares e, portanto, concordam sobre todas as entradas, mesmo aquelas fora do conjunto de treinamento. Assim sendo, o erro de generalização será essencialmente igual à média ponderada dos erros de generalização dos componentes, isto é, ($E_{ens} \approx \bar{E}$). Por outro lado, caso haja alta variância, a ambigüidade será alta e o erro de generalização da *ensemble* será menor que o erro de generalização médio do grupo, ou seja, $E_{ens} < \bar{E}$.

Como alternativa ao resultado de Krogh & Vedelsby (Eq. (4.9)), Zhou *et al.* [513] chegaram a um equacionamento um pouco diferente que estabelece o erro de generalização de uma *ensemble* em função do grau de correlação entre seus membros:

$$E_{ens} = \sum_{k=1}^K \sum_{k^*=1}^K \alpha_k \alpha_{k^*} C_{kk^*}, \quad (4.10)$$

onde α_k indica o peso sobre a saída do k -ésimo componente — ver discussão adiante sobre *ensembles* generalizadas — e $C_{kk^*} = E[\epsilon_k(\mathbf{x}) \cdot \epsilon_{k^*}(\mathbf{x})]$ refere-se à matriz de correlação dos erros dos componentes, sendo que $\epsilon_k(\mathbf{x}) = f_k(\mathbf{x}) - y_d(\mathbf{x})$ e $E[\cdot]$ é o operador esperança matemática. Uma vez que a computação de C_{kk^*} envolve somente f_k e f_{k^*} , Zhou e colaboradores advogam que (4.10) seja mais fácil de ser adotada do que (4.9) em aplicações práticas.

Trilhando um outro caminho, Hansen & Salamon [191] verificaram formalmente que, considerando um problema de classificação binária, se as taxas de erro relativas aos K modelos disponíveis (representadas genericamente por p) são todas independentemente distribuídas e menores que 50%, então a probabilidade de que a saída do voto majoritário aplicado a esses modelos esteja errada (P_e) é menor que a menor das taxas de erro, o que pode ser confirmado analisando-se a área da distribuição binomial em que mais da metade das K estimativas esteja errada. De uma forma matemática [123]:

$$P_e = \sum_{k > \lceil K/2 \rceil}^K \binom{K}{k} p^k (1-p)^{K-k}.$$

Como $p < 1/2$, tal expressão é monotonicamente decrescente em K , o que faz com que, para $K \rightarrow \infty$, $p_e \rightarrow 0$. Esse resultado tem sido estudado (ainda que com menos rigor formal) há mais de três séculos no âmbito

⁽⁶⁾A ambigüidade relativa a um membro k , para uma amostra $\mathbf{x}$ qualquer, denota a diferença entre a saída produzida por k , digamos $f_k(\mathbf{x})$, e a média das saídas produzidas pelos K componentes, ou seja, $\Omega(\mathbf{x})$. Já a ambigüidade da *ensemble* captura numa só variável todas as correlações existentes entre os padrões de erro dos membros.

das ciências políticas e sociais, desde o estabelecimento do modelo de júri Condorcet [114] supracitado, e serviu de inspiração mais recentemente para a pesquisa em estimadores “fracos” (*weak*) [233, 254] ou “instáveis” (*instable*) [63], representando uma linha de investigação diametricamente oposta àquela voltada ao desenvolvimento de estimadores altamente precisos e específicos⁽⁷⁾.

Por outro lado, Tumer & Ghosh [455, 456] propiciaram uma análise quantitativa e bayesiana dos possíveis ganhos a serem obtidos com *ensembles* de RNAs em termos da taxa de reconhecimento. Os autores mostraram que, mediante o uso de combinadores lineares ou baseados em medidas estatísticas de maior ordem (i. e., momentos), pode-se reduzir a variância das fronteiras de decisão em mãos (relativas aos K componentes) em torno da fronteira de decisão ótima. Ademais, na ausência de polarização nas redes, a redução no erro adicional — devido às imperfeições dos componentes — àquele de cunho essencialmente bayesiano passa a ser diretamente proporcional à redução da variância. Uma combinação linear de K redes não-polarizadas, cujas distribuições de erro são independente e identicamente distribuídas (i.i.d.), pode reduzir a variância por um fator de K . Num primeiro momento, tal resultado parece ser significativo, pois, para quando $K \rightarrow \infty$, a variância tende a zero. Contudo, como apontado por Jain *et al.* [227], tal consideração em relação à manutenção da propriedade de i.i.d. para as K distribuições *não* é realística para valores elevados de K .

De maneira similar, Perrone & Cooper [369] demonstraram, com base na hipótese de independência e média nula para os desvios exibidos pelos componentes, que a média simples das saídas dos K componentes — que foi denominada Método de *Ensemble* Básica (MEB) —, ou seja,

$$\Omega_{MEB}(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^K f_k(\mathbf{x}),$$

pode reduzir o erro quadrático médio ($EQM[\Omega_{MEB}]$) por um fator de K , quando este é cotejado com a média dos EQMs dos K componentes. Novamente, avaliando para $K \rightarrow \infty$, chega-se que $EQM[\Omega_{MEB}] \rightarrow 0$, porém a mesma colocação feita anteriormente sobre a não-manutenção da propriedade de independência das distribuições de erros procede aqui também. Perrone & Cooper [369] também propuseram o Método de *Ensemble* Generalizada (MEG), que consiste da combinação linear ótima [196] no sentido de quadrados mínimos. Para esse método,

$$\Omega_{MEG}(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^K \alpha_k f_k(\mathbf{x}),$$

onde os pesos α_k 's passam a ser derivados a partir da matriz de correlação dos erros dos K módulos. Recorrendo-se à definição de C_{kk} dada acima, então o objetivo passa a ser o de encontrar os α_k 's que

⁽⁷⁾Segundo tal linha, *ensembles* não são recomendadas para serem aplicadas à combinação de modelos estáveis; por exemplo, modelos robustos de regressão logística [197].

minimizem (4.10). Para tanto, a escolha ótima para cada α_k deve ser

$$\alpha_k = \frac{\sum_{k^*=1}^K (C_{kk^*})^{-1}}{\sum_{k^{**}=1}^K \sum_{k^*=1}^K (C_{k^*k^{**}})^{-1}}.$$

Foi comprovado [369] que o EQM para esse método de combinação ($EQM[\Omega_{MEG}]$) tende a ser menor que aquele apresentado por uma *ensemble* básica como também por aquele apresentado pela melhor rede-componente. A aplicação com sucesso desse método depende de uma estimativa confiável da matriz $\mathbf{C} = (C_{kk^*})_{K \times K}$ e impõe a restrição de que $\mathbf{C}$ seja não-singular para ser invertível. Na prática, todavia, os erros pendem para serem altamente correlacionados, tornando as linhas de $\mathbf{C}$ linearmente dependentes, o que faz com que erros significativos de arredondamento sejam gerados quando da inversão dessa matriz [234].

Seguindo uma perspectiva um pouco diferente, Kittler *et al.* [253] desenvolveram um *framework* conceitual comum abarcando um grupo de esquemas de combinação em que classificadores individuais se utilizam de representações distintas de uma mesma amostra de entrada para estimarem as probabilidades *a posteriori* associadas a ela. O objetivo primal foi o de explicar em que circunstâncias e por qual motivo alguns esquemas de combinação têm inclinação para apresentar desempenho superior a outros — no caso, segundo os experimentos conduzidos pelos autores, a regra aditiva (i. e., média simples) foi a que obteve êxito mais consistentemente. Para esse propósito, os autores encaminharam uma análise de sensibilidade bem elaborada, em concordância com o *framework* teórico estabelecido. A explicação encontrada foi a de que a regra aditiva é menos sensível aos erros individuais dos classificadores quando da geração das suas probabilidades *a posteriori*. Além disso, ela parece ser mais apropriada para combinar diferentes estimativas das mesmas probabilidades, oriundas, por exemplo, de diferentes iniciações de um classificador neural. O combinador por produto, segundo os pesquisadores, passaria a ser mais adequado para trabalhar com probabilidades independentes e livres de erro, provenientes, por exemplo, de densidades estimadas a partir de conjuntos de atributos independentes.

Por fim, é válido mencionar acerca do trabalho conduzido por Schapire *et al.* [411] encaminhando uma tentativa de explicação formal para a eficácia apresentada por métodos de votação (principalmente quando aplicados a modelos gerados via reamostragem dos dados). Tal explicação fundamenta-se no conceito de *margem*, que mede a diferença entre o escore de suporte combinado (p. ex., soma das saídas de ativação, em se tratando de RNAs) para a classe verdadeira e o mais alto escore de suporte combinado dentre todos aqueles relativos as $M - 1$ demais classes (incorretas). Schapire *et al.* estabeleceram que o erro de generalização de uma combinação de estimadores qualquer é limitado pela soma entre a probabilidade de cauda da distribuição das margens sobre os pontos de treinamento e um termo que é função da complexidade de um classificador simples (componente) — em vez de ser da própria combinação em si. Os autores demonstraram que as técnicas de *Boosting* podem efetivamente aumentar a média e diminuir a variância da distribuição das margens, incorrendo em redução da primeira parcela do erro de generalização⁽⁸⁾.

⁽⁸⁾De acordo com Jain *et alia* [227], tal propriedade tem efeito similar àquele exibido por classificadores SVM, os

4.1.2 Diversidade em *ensembles*

Da discussão anterior, fica aparente que um dos principais determinantes para a eficácia de uma *ensemble* qualquer diz respeito à proporção em que seus componentes são não-correlacionados no que tange aos seus padrões de erro. Similarmente, a proporção em que seus componentes são correlacionados deverá influenciar no modo em que serão combinados [427]. Diversidade em uma *ensemble* pode ser induzida via diferentes estratégias, tais como variações na apresentação dos padrões de treinamento (vide técnicas de reamostragem dos dados discutidas na próxima subseção), variações relativas ao algoritmo de indução (parâmetros de controle, iniciações distintas para os parâmetros livres, ou mesmo indutores distintos) ou introdução de penalidades a especialistas correlacionados quando da sua configuração (como discutido abaixo).

Numa tentativa de caracterizar uma *ensemble* qualquer em consonância com o padrão de discordância entre as decisões de seus componentes, Sharkey *et al.* [423, 424] chegaram a identificar quatro níveis distintos de diversidade. No primeiro nível, não há erros coincidentes apresentados pelos componentes quando avaliados sobre um conjunto de validação à parte. No segundo, há alguns erros coincidentes, mas a maioria dos módulos está sempre correta — ou seja, a taxa de reconhecimento da *ensemble* continua sendo 100%. Já no terceiro nível, a maioria dos módulos nem sempre está correta, mas, para cada padrão, pelo menos um dos componentes prevê corretamente a saída. Por fim, no quarto nível, a maioria dos módulos nem sempre está correta e existem alguns pontos para os quais nenhum dos componentes produz a saída correta, entretanto, há ainda alguma diferença entre os padrões de erro exibidos pelos componentes que justifique a sua agregação. A vantagem em se lançar mão de categorizações como essa é a de permitir ao projetista tomar ações diferenciadas (p. ex., manter ou descartar os componentes) de acordo com o nível de diversidade.

Ademais, já existem na literatura alguns critérios para mensurar, quantitativamente, o nível de diversidade inerente a um arranjo de combinação qualquer. Exemplos de algumas das métricas sugeridas são comentados sucintamente na Tab. 4.1 à página 68. Vale frisar que não há receitas sugerindo quando uma métrica deve ser preterida em favor de outra ou quando uma métrica deve ter sua aplicação priorizada, de modo que a única forma de se avaliar a eficácia de uma métrica para um certo domínio de aplicação é via comparação experimental com outras [396]. Nesse sentido, Kuncheva & Whitaker [267] passaram a analisar, empiricamente, oito métricas de dissimilaridade entre membros de uma *ensemble* hipotética baseada em voto majoritário, as quais foram divididas em dois grupos: quatro métricas baseadas em médias de medidas par-a-par entre os K classificadores e quatro métricas baseadas em medidas sobre o grupo todo. A primeira conclusão interessante a que Kuncheva & Whitaker chegaram foi a de que a acuidade da *ensemble* foi maior para componentes diversos, mas ainda com alguma correlação, do que para componentes totalmente independentes entre si, isso para todos os critérios investigados. A segunda conclusão foi a de que todas as métricas estudadas são equiparadas no que concerne à sua capacidade em medir diversidade em combinações de classificadores — embora os autores recomendem o uso da estatística- Q , elaborada por eles, haja vista

quais, como se observou no Cap. 3, põem ênfase maior sobre aqueles pontos de treinamento próximos à margem. Claro que “margem”, para SVMs, assume uma conotação distinta daquela adotada por Schapire *et al.*, podendo ser visualizada como a área de sobreposição entre as densidades condicionais relativas as M classes existentes.

ela apresentar maior facilidade de interpretação e de cálculo.

Com o intuito de induzir maiores níveis de diversidade entre componentes de uma *ensemble*, dois algoritmos bastante similares foram propostos, separadamente, por Rosen [397] e por Liu & Yao [297], ambos fundamentados na idéia de se introduzir um *termo de penalidade* na função-custo de cada componente (no caso, RNA *feedforward*), em proporção ao grau de correlação de suas saídas para com as saídas produzidas pelos demais membros do grupo. A idéia aventada por Rosen [397] é que redes individuais devem tentar não somente minimizar o erro entre suas saídas e os valores-alvo mas também descorrelacionar os seus erros daqueles produzidos pelas outras redes treinadas anteriormente. Ou seja, a abordagem de Rosen é de natureza *seqüencial*. Já o algoritmo de “aprendizado por correlação negativa” de Liu & Yao [297] introduz o termo de penalidade de tal forma a que todos os membros da *ensemble* possam ser treinados *simultaneamente* e *iterativamente* sobre um mesmo conjunto de treinamento D_N . De acordo com o algoritmo, a função-custo E_k para uma rede-componente k passa a ser definida como:

$$\begin{aligned} E_k &= \frac{1}{N} \sum_{n=1}^N E_k(n) \\ &= \frac{1}{N} \sum_{n=1}^N \frac{1}{2} [f_k(n) - y_{d_n}]^2 + \frac{1}{N} \sum_{n=1}^N \lambda p_k(n), \end{aligned} \quad (4.11)$$

onde $f_k(n)$ e $E_k(n)$ são, respectivamente, a saída e o valor da função de erro para a k -ésima rede sobre o n -ésimo padrão de treinamento, ao passo que y_{d_n} indica a saída desejada para esse padrão. O primeiro termo no lado direito de (4.11) denota a função de risco empírico do componente k , enquanto $p_k(\cdot)$ alude a uma função de penalidade de correlação, cujo propósito é o de correlacionar negativamente o erro da k -ésima rede com os erros das demais redes-componentes. O parâmetro $0 \leq \lambda \leq 1$ controla a força exercida pela penalidade. A função $p_k(\cdot)$ assume o seguinte formato:

$$p_k(n) = [f_k(n) - \Omega(n)] \sum_{k^* \neq k} [f_{k^*}(n) - \Omega(n)], \quad (4.12)$$

onde $\Omega(n) = \frac{1}{K} \sum_{k=1}^K f_k(n)$ denota a saída da *ensemble* (via operador “média simples”).

A derivada parcial de $E_k(n)$ com respeito à saída do k -ésimo módulo para o n -ésimo padrão de treinamento pode ser calculada como:

$$\begin{aligned} \frac{\partial E_k(n)}{\partial f_k(n)} &= f_k(n) - y_{d_n} + \lambda \frac{\partial p_k(n)}{\partial f_k(n)} \\ &= f_k(n) - y_{d_n} + \lambda \sum_{k^* \neq k} [f_{k^*}(n) - \Omega(n)] \\ &= f_k(n) - y_{d_n} - \lambda [f_k(n) - \Omega(n)] \\ &= (1 - \lambda) [f_k(n) - y_{d_n}] + \lambda [\Omega(n) - y_{d_n}], \end{aligned} \quad (4.13)$$

onde considerou-se que $\Omega(n)$ apresenta um valor constante com relação a $f_k(n)$. A partir desse desenvolvimento, pode-se constatar que o aprendizado por correlação negativa pode ser visto como uma extensão simples do algoritmo *Backpropagation* (Subseção 2.4.2) operando no modo padrão-a-padrão — ou seja, a atualização dos pesos de todas as redes-componentes é feita simultaneamente, mediante (4.13), após a apresentação de cada amostra. Cada mapeamento individual f_k minimiza não somente a diferença entre $f_k(n)$ e y_{d_n} , mas também a diferença entre $\Omega(n)$ e y_{d_n} . Quando $\lambda = 0$, não há penalização e o algoritmo reduz-se ao treinamento independente das redes via *Backpropagation* como feito tradicionalmente. Já quando $\lambda = 1$, (4.13) reduz-se a $\frac{\partial E_k(n)}{\partial f_k(n)} = \Omega(n) - y_{d_n}$. Definindo o risco empírico da *ensemble* para o n -ésimo padrão como

$$E_{ens}(n) = \frac{1}{2} \left[\frac{1}{K} \sum_{k=1}^K f_k(n) - y_{d_n} \right]^2,$$

e após algumas manipulações algébricas, pode-se constatar, para o caso de $\lambda = 1$, que

$$\frac{\partial E_k(n)}{\partial f_k(n)} \propto \frac{\partial E_{ens}(n)}{\partial f_k(n)}.$$

Ou seja: a minimização do risco empírico da *ensemble* se dá pela minimização das funções-custo associadas às suas redes-componentes, promovendo uma espécie de decomposição da tarefa de aprendizado geral em subtarefas menores [297, 298].

Tab. 4.1: Algumas métricas para quantificar o grau de diversidade em uma *ensemble* de classificadores qualquer. São considerados M classes, N amostras de treinamento (validação) integrando o conjunto D_N e K componentes, cujos índices formam o conjunto $\Gamma = \{1, \dots, K\}$. Considera-se ainda que $a_k^j(\mathbf{x})$ seja o grau de suporte (ver definição à página 80) do k -ésimo componente para a j -ésima classe com relação à amostra $\mathbf{x}$. (Fontes: Kuncheva & Whitaker [267], Lee *et al.* [281] e Roli *et al.* [396].)

Nome	Descrição
Distância euclidiana entre as médias dos graus de suporte	A média dos graus de suporte para o k -ésimo componente pode ser calculada como: $\bar{a}_k = \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M a_k^j(\mathbf{x}_i)$. Dados dois estimadores k e k^* , então o grau de discordância entre eles pode ser obtido mediante o cálculo da distância euclidiana entre as médias dos seus graus de suporte, ou seja, $div_e(k, k^*) = \ \bar{a}_k - \bar{a}_{k^*}\ $. Quanto maior for o valor de $div_e(k, k^*)$, mais discordantes serão tais componentes. De posse desses valores (e descartando as distâncias entre componentes iguais), pode-se ter uma noção geral da diversidade da <i>ensemble</i> via cálculo da média desses valores, ou seja $div_e(\Omega) = \frac{2}{K(K-1)} \sum_{k=1}^{K-1} \sum_{k^*=k+1}^K div_e(k, k^*)$.

Continua na próxima página

Tab. 4.1: Métricas de diversidade em *ensembles*. (cont.)

Entropia cruzada	Interpretando os graus de suporte de um estimador como probabilidades <i>a posteriori</i>, pode-se lançar mão da entropia de Kullback-Leibler modificada (também conhecida como entropia cruzada) como medida direta de dissimilaridade entre dois componentes: $div_{cruz}(k, k^*) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^M \left(a_k^j(\mathbf{x}_i) \log\left(\frac{a_k^j(\mathbf{x}_i)}{a_{k^*}^j(\mathbf{x}_i)}\right) + a_{k^*}^j(\mathbf{x}_i) \log\left(\frac{a_{k^*}^j(\mathbf{x}_i)}{a_k^j(\mathbf{x}_i)}\right) \right)$. A diversidade da <i>ensemble</i> pode ser calculada como anteriormente: $div_{cruz}(\Omega) = \frac{2}{K(K-1)} \sum_{k=1}^{K-1} \sum_{k^*=k+1}^K div_{cruz}(k, k^*)$.
Correlação de Pearson	Neste caso, a discordância entre dois componentes, k e k^*, pode ser calculada como o inverso do coeficiente de correlação entre seus graus de suporte, ou seja, $div_{corr}(k, k^*) = \left(\frac{\sum_{j=1}^M (a_k^j - \bar{a}_k) \times (a_{k^*}^j - \bar{a}_{k^*})}{\sqrt{\sum_{j=1}^M (a_k^j - \bar{a}_k)^2 \times \sum_{j=1}^M (a_{k^*}^j - \bar{a}_{k^*})^2}} \right)^{-1}$, sendo $\bar{a}_k$ calculada como no item anterior. De posse desses valores, chega-se a uma medida do grau de diversidade em uma <i>ensemble</i> como $div_{corr}(\Omega) = \frac{2}{K(K-1)} \sum_{k=1}^{K-1} \sum_{k^*=k+1}^K div_{corr}(k, k^*)$.
Diversidade generalizada	Assumindo que $p(2.falham) = \sum_{k=1}^K \frac{k}{K} \frac{(k-1)}{(K-1)} p_k$ denota a probabilidade de dois classificadores selecionados aleatoriamente de Γ falharem (i. e., produzirem saída errônea) simultaneamente para uma amostra aleatória qualquer e que $p(1.falha) = \sum_{k=1}^K \frac{k}{K} p_k$ indica a probabilidade de falha de um componente qualquer para uma amostra qualquer, sendo que $p_k = p(Y = \frac{k}{K})$, onde Y é uma variável aleatória expressando a proporção de classificadores incorretos de Γ em uma amostra $\mathbf{x} \in D_N$ qualquer, então $div_{dg}(\Omega) = 1 - \frac{p(2.falham)}{p(1.falha)}$ ($div_{dg} \in [0, 1]$) provê uma medida direta de diversidade para a <i>ensemble</i> cujos componentes formam Γ.
Diversidade de falhas coincidentes	É uma modificação do caso anterior. A nova medida $div_{dfc}(\Omega)$ assumirá valor mínimo (zero) quando todos os classificadores estiverem sempre corretos ou quando todos os classificadores forem ou todos corretos ou todos errados alternadamente. O seu valor máximo ('1') ocorrerá quando, no máximo, um único classificador falhar para uma amostra aleatória qualquer. Disso: $div_{dfc}(\Omega) = \begin{cases} 0 & \text{se } p_0 = 1 \\ \frac{1}{1-p_0} \sum_{k=1}^K \frac{K-k}{K-1} p_k & \text{se } p_0 < 1 \end{cases}$

Continua na próxima página

Tab. 4.1: Métricas de diversidade em *ensembles*. (cont.)

Estatística- Q	Para cada classificador k , constrói-se o vetor $O_k = [O_{k1} \dots O_{kN}]$, cujo elemento O_{ki} assumirá valor igual a '1' quando k acertar o rótulo para $\mathbf{x}_i \in D_N$ e '0', caso o contrário. A estatística- Q avalia o grau de divergência entre dois classificadores k e k^* quaisquer mediante $Q(k, k^*) = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$, onde N^{ab} denota o número de elementos $\mathbf{x}_i$ de D_N para os quais $O_{ki} = a$ e $O_{k^*i} = b$ — pode-se verificar que $N = N^{00} + N^{01} + N^{10} + N^{11}$ e que Q varia em $[-1, 1]$. Classificadores que tendem a ser correlacionados exibirão valores positivos de Q , ao passo que aqueles que cometem erros em padrões distintos exibirão valores negativos dessa medida. Já para classificadores independentes, $Q = 0$. A diversidade da <i>ensemble</i> é dada pela média sobre todos os possíveis acoplamentos entre componentes, ou seja, $div_Q(\Omega) = \frac{2}{K(K-1)} \sum_{k=1}^{K-1} \sum_{k^*=k+1}^K Q(k, k^*)$.
Diversidade composta	Seja $p(k\_falha, k^*\_falha)$ a probabilidade conjunta de erro dos classificadores k e k^* , então, lançando mão da notação anterior, $div_{dc}(k, k^*) = 1 - p(k\_falha, k^*\_falha) = 1 - \frac{N^{00}}{N}$ provê uma medida direta de dissimilaridade entre tais componentes, ao tempo que $div_{dc}(\Omega) = \frac{2}{K(K-1)} \sum_{k=1}^{K-1} \sum_{k^*=k+1}^K div_{dc}(k, k^*)$ fornece uma medida do grau de diversidade intrínseco a <i>ensemble</i> cujos componentes integram Γ .
Medida de desacordo	Para dois classificadores $k, k^* \in \Gamma$, essa medida é dada pela razão entre o número de amostras para as quais um dos classificadores está correto e o outro errado e o número total de amostras de D_N , ou seja $div_{md}(k, k^*) = \frac{N^{01} + N^{10}}{N}$, usando a notação definida para a estatística- Q . Disso, $div_{md}(\Omega) = \frac{2}{K(K-1)} \sum_{k=1}^{K-1} \sum_{k^*=k+1}^K div_{md}(k, k^*)$ fornece uma medida do grau de diversidade para a <i>ensemble</i> .
Medida de acordo entre taxadores	Usando a notação definida para a estatística- Q , a medida κ par-a-par é calculada como $\kappa(k, k^*) = \frac{2(N^{11}N^{00} - N^{01}N^{10})}{(N^{11} + N^{10})(N^{01}N^{00}) + (N^{11} + N^{01})(N^{10}N^{00})}$, enquanto a medida do grau de diversidade da <i>ensemble</i> pode ser dada como $div_{\kappa}(\Omega) = \frac{2}{K(K-1)} \sum_{k=1}^{K-1} \sum_{k^*=k+1}^K \kappa(k, k^*)$.

4.1.3 Criação de componentes

Uma abordagem adequada para se criar *ensembles* efetivas é a de levar em consideração os méritos relativos exibidos pelos diferentes métodos já existentes de criação de componentes, tentando recorrer àquele(s) com maior potencial para garantir, de antemão, níveis elevados de diversidade, tendo em mente os requisitos da aplicação em foco [427]. Nesse âmbito, Wolpert [492] se referiu às “receitas” empíricas disponíveis à época sobre a escolha do melhor método de geração de componentes como uma espécie de “arte negra”. Contudo, após um número bem razoável de investigações experimentais conduzidas por pesquisadores de renome ao longo da década passada [41, 63, 124, 129, 205, 306, 352, 424, 456, 497], já se chegou a um certo consenso que, em princípio, métodos empregando algum tipo de variação nos dados de treinamento e métodos alocando

diferentes dimensões do espaço de entradas a diferentes componentes são aqueles com maiores chances de propiciarem maiores níveis de diversidade para uma *ensemble* qualquer.

Com efeito, *Bagging*⁽⁹⁾ [61] e *Boosting* [160, 412] dizem respeito a técnicas estatísticas poderosas que lançam mão do conceito de redistribuição aleatória de dados (*bootstrapping* [197] — ver Apêndice A) como meio efetivo de gerar componentes diversos para serem agregados. Ambas as técnicas, segundo Breiman [62], podem reduzir os níveis de polarização da hipótese resultante, embora seja sabido que o grande poder por trás delas esteja na substantiva redução dos níveis de variância que normalmente promovem.

Breiman [63] também argumenta que os estimadores mais propícios para serem combinados via *Bagging* são aqueles de gênero *instável*, dentre os quais figuram os modelos neurais, os de árvore de decisão e os baseados em noções de vizinhança (como *k-nearest neighbors* [197]). Preditores instáveis são assim chamados pelo fato de seu processo de indução apresentar alta sensibilidade a pequenas mudanças nas condições de treinamento; por exemplo, perturbações no conjunto de dados ou condições iniciais podem acarretar a geração de hipóteses bastante discrepantes no que tange à sua capacidade de generalização. Contudo, para preditores estáveis, como o decisor bayesiano ingênuo [197], o uso de *Bagging* pode incorrer em efeito contrário ao desejado, ou seja, aumentar a variância.

A explicação para isso decorre diretamente do modo como *Bagging* opera: tal técnica estipula que diferentes conjuntos de treinamento (digamos, K deles) sejam produzidos a partir de um repositório comum (D_N) via *reamostragem aleatória com reposição* das amostras. Cada novo conjunto de dados D'_N terá o mesmo número de amostras (N) do original. Todavia, uma vez que algumas amostras deverão ser escolhidas repetidamente, enquanto outras não serão aproveitadas para integrar D'_N , o tamanho desse conjunto derivado será *efetivamente* menor que o do original, não existindo praticamente nenhuma chance de eles serem idênticos. Isso porque há um grande desbalanceamento entre a probabilidade de uma amostra qualquer não ser escolhida para integrar D'_N (em torno de 0,368) e a probabilidade de ela ser escolhida pelo menos uma vez (quase o dobro da anterior) — ver discussão no Apêndice A. Ademais, normalmente um único indutor é usado em *Bagging* para o propósito de geração dos modelos dos estimadores, sendo que o k -ésimo estimador f_k deve ser produzido com base nos dados do k -ésimo conjunto derivado (D_N^k). Para a agregação dos componentes resultantes, são usualmente empregados combinadores fixos (ver Subseção 4.1.5), como voto majoritário e média simples, o primeiro para problemas de classificação e o segundo para regressão [124].

O Alg. 1 traz um pseudocódigo genérico dessa técnica, ao passo que a Fig. 4.3(b) ilustra um exemplo hipotético indicando a sua operação sobre um conjunto de dados artificial. Uma vez que o conjunto D_N^1 dessa subfigura apresenta amostras repetidas, deixando as de rótulos '#4' e '#5' de fora, é bem provável que o classificador resultante apresente maior taxa de erro sobre o teste que aquele treinado sobre todos os dados disponíveis (Fig. 4.3(a)). De fato, todos os quatro classificadores treinados sobre $D_N^1, \dots, D_N^4$ tendem a apresentar pior desempenho na fase de teste que o classificador da Fig. 4.3(a); todavia, quando combinados, tais estimadores podem (e freqüentemente o fazem) produzir uma taxa de erro mais baixa sobre o conjunto de teste que o classificador simples, uma vez que a diversidade entre os componentes passa a compensar os

⁽⁹⁾De *bootstrap aggregation and combining*.

seus baixos desempenhos individuais.

Algoritmo 1 *Bagging* para classificação

Entrada: $\mathcal{L}$ (indutor instável), D_N (repositório de dados original de tamanho N), $C = \{1, \dots, M\}$ (conjunto de rótulos das classes) e K (número de iterações) .

Saída: $\Omega(\mathbf{z}) = \arg \max_{j \in C} \sum_{k=1}^K f_k(\mathbf{z}, j)$ (voto majoritário), onde $f_k(\mathbf{z}, j) = \begin{cases} 1 & \text{se } f_k(\mathbf{z}) = j \\ 0 & \text{caso contrário} \end{cases}$.

Computação:

$k \leftarrow 1$;

while ($k \leq K$) **do**

$D_N^k \leftarrow \emptyset$;

$i \leftarrow 1$;

while ($i \leq N$) **do**

$\mathbf{x} \leftarrow \text{AMOSTRAGEM\_ALEATORIA}(D_N)$;

$D_N^k \leftarrow D_N^k \cup \{\mathbf{x}\}$;

$i \leftarrow i + 1$;

end while

$f_k \leftarrow \mathcal{L}(D_N^k)$;

$k \leftarrow k + 1$;

end while

Técnicas de *Boosting* [129, 160, 411, 412], (referenciadas também como *Arcing*⁽¹⁰⁾ por Breiman [62]) constituem uma família de métodos de cunho estatístico também baseada em noções de reamostragem de dados para a geração de uma *seqüência* de conjuntos de treinamento e seus respectivos preditores. No caso, a distribuição de probabilidade associada a cada um dos novos conjuntos passa a ser ajustada *adaptativamente* e é dita ser uma “representação viesada” (*overrepresentation*) da distribuição original [227], na medida em que a sua população de pontos tende a ser constituída por amostras mais similares àquelas mal interpretadas (ou seja, que tiveram suas saídas preditas erroneamente) pelos estimadores anteriores da seqüência. De um outro modo: os casos de treinamento que forem preditos incorretamente pelos primeiros estimadores da série serão escolhidos mais freqüentemente que aqueles preditos corretamente para integrarem os novos conjuntos de dados a serem gerados. Assim, em *Boosting*, os estimadores individuais passam a ser treinados *hierarquicamente*, ficando o desempenho de um condicionado àqueles exibidos por seus antecedentes, de modo que os últimos estimadores na seqüência geralmente deverão arcar com regiões mais complicadas do espaço de atributos [197, 352].

A versão original de *Boosting* foi concebida por Schapire [412], tendo como base uma variação do modelo de aprendizado PAC (*probably approximately correct*), conhecida como “modelo de aprendizado fraco”, proposta anteriormente por Kearns & Valiant [247]. Nesse contexto, um *algoritmo forte de aprendizado PAC* seria aquele que, dadas as taxas $\epsilon, \delta > 0$ e a possibilidade de acesso a amostras aleatórias, produziria

⁽¹⁰⁾De *adaptive resampling and combining*.

Uma execução de um classif. simples sobre um conjunto de dados	
Conjunto de dados (original)	
D_N^1	1, 2, 3, 4, 5, 6, 7, 8

(a)

Uma execução de <i>Bagging</i> sobre os mesmos dados	
Conjunto de dados (reamostrado)	
D_N^1	2, 7, 8, 3, 7, 6, 3, 1
D_N^2	7, 8, 5, 6, 4, 2, 7, 1
D_N^3	3, 6, 2, 7, 5, 6, 2, 2
D_N^4	4, 5, 1, 4, 6, 4, 3, 8

(b)

Uma execução de <i>Boosting</i> sobre os mesmos dados	
Conjunto de dados (reamostrado)	
D_N^1	2, 7, 8, 3, 7, 6, 3, 1
D_N^2	1, 4, 5, 4, 1, 5, 6, 4
D_N^3	7, 1, 5, 8, 1, 8, 1, 4
D_N^4	1, 1, 6, 1, 1, 3, 1, 5

(c)

Fig. 4.3: Exemplo de operação de *Bagging* e *Boosting* sobre um conjunto de dados hipotético. Assume-se que $N = 8$ exemplos e que a amostra ‘#1’ seja um valor aberrante (*outlier*), difícil de ser tratado pelo indutor instável adotado. Com *Bagging* (b), cada novo conjunto de dados D_N^k compreende uma reamostragem diferente dos dados originais; daí, alguns casos estão faltantes enquanto outros ocorrem múltiplas vezes. Os conjuntos gerados por *Boosting* (c) são também reamostragens do conjunto original, contudo a amostra aberrante ‘#1’ passa a ocorrer com maior frequência nos últimos conjuntos gerados, posto que a técnica destina-se a concentrar cada vez mais suas “forças” para prevê-lo corretamente. (Fonte: Opitz & Maclin [352].)

como saída, com probabilidade $1 - \epsilon$, uma hipótese cujo erro seria, no máximo, igual a δ . Mais ainda, o tempo de execução desse algoritmo deveria ser polinomial em relação a $1/\epsilon, 1/\delta$. Já um *algoritmo fraco de aprendizado PAC* satisfaria as mesmas condições anteriores mas somente para $\epsilon \geq 1/2 - \gamma$, sendo que γ ou seria uma constante (positiva) ou seria uma variável que decrescesse com $1/p$, com p indicando um polinômio qualquer sobre os parâmetros relevantes do problema. Considerando que γ fosse, pelo menos, um pouco maior que zero, Schapire [412] provou que, dado um número significativo de “aprendizes fracos”, sua técnica de *Boosting* poderia decrementar a taxa de erro sobre o conjunto de treinamento e converter a *ensemble* de “aprendizes fracos” em um único “aprendiz forte”, cuja taxa de erro seria arbitrariamente pequena. Teoricamente, basta que γ seja levemente maior que zero (quer dizer, que a taxa de erro seja um pouco melhor que aquela gerada por um decisor aleatório, isto é, 0,5) para cada estimador de sorte a que a taxa de erro de treinamento da *ensemble* convirja para zero. Na prática, quão maior for γ , mais rápida será a queda na taxa de erro da *ensemble* [128].

Haykin [198] aponta três diferentes abordagens seguidas por técnicas de *Boosting*:

1. *Boosting por filtragem*: Essa abordagem envolve a filtragem progressiva das amostras de treinamento pelas diferentes versões de um mesmo indutor fraco. Ela assume a disponibilidade de um grande (infinito, na teoria) repositório de amostras, sendo tais amostras ou descartadas ou mantidas ao longo do treinamento das máquinas. Uma vantagem dessa abordagem é que ela impõe baixos requisitos de memória quando comparada com as demais. O algoritmo original proposto por Schapire [412] e avaliado posteriormente por Drucker *et al.* [129] insere-se nessa classe.
2. *Boosting por reamostragem*: Essa abordagem trabalha com conjuntos de dados de tamanho fixo (igual ao do repertório inicial), sendo que os seus exemplos são escolhidos por reamostragem, de acordo com uma distribuição de probabilidade preestabelecida, ao longo do treinamento das máquinas — ou seja, não há descartes definitivos de amostras, como no caso anterior. O erro de um componente f_k é calculado com respeito a *todo* o conjunto de treinamento a ele associado (D_N^k). Os algoritmos Arcing (ARC-X4 e ARC-LH), analisados por Breiman [62] e Huber & Mayer [213], se encaixam nessa classe. Similarmente a *Bagging*, métodos Arcing lançam mão do conceito de reamostragem aleatória com reposição; no entanto, diferentemente de *Bagging*, a probabilidade de escolha de uma amostra $\mathbf{x}_1$ qualquer não é necessariamente igual a da escolha de uma segunda amostra $\mathbf{x}_2$ do repertório disponível. Tais probabilidades dependerão de quão freqüentemente $\mathbf{x}_1$ e $\mathbf{x}_2$ foram mal interpretadas pelos estimadores anteriores.
3. *Boosting por reponderação*: Essa abordagem trabalha com conjuntos de treinamento derivados idênticos ao repositório original (portanto, de tamanho fixo), embora assuma que o indutor fraco receba amostras com pesos associados (ditas “amostras ponderadas”) — quão maior o peso w_i de uma amostra $\mathbf{x}_i$, peso esse diretamente proporcional ao erro de predição sobre ela apresentado pelo preditor anterior, mais importante $\mathbf{x}_i$ será para o preditor atual. O erro de cada componente passa a ser calculado, pois, com base em todos os exemplos do repertório original — isso é uma vantagem sobre as classes anteriores — e com base nos seus pesos associados. Friedman *et al.* [197] demonstraram

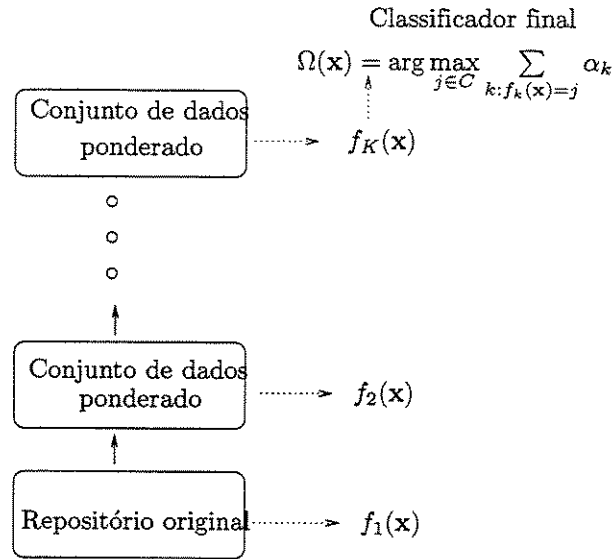


Fig. 4.4: Esquemático do método AdaBoost com amostras ponderadas. Os classificadores são treinados com versões “ponderadas” do repositório original (ou seja, com amostras discriminadas por pesos) e são posteriormente combinados via voto majoritário ponderado para realizarem a estimativa final. (Fonte: Hastie *et al.* [197].)

que essa forma de *Boosting* pode ser vista como um mecanismo alternativo de modelagem aditiva para otimizar funções de perda logística, apresentando, por conseguinte, maiores potenciais que as anteriores. O algoritmo AdaBoost, concebido por Freund & Schapire [160, 161], se encaixa nessa classe — conquanto ele apresente também a opção por reamostragem (caso anterior).

No que segue, passa-se a explicar, sucintamente, o funcionamento da versão M1 do algoritmo AdaBoost (em seu formato por reponderação)⁽¹¹⁾, haja vista ela ter sido adotada para fins de comparação com a abordagem COHENN — ver Seção 7.3. O Alg. 2 provê um pseudocódigo genérico do método AdaBoost.M1, ao passo que as Figs. 4.3(c) e 4.4 apresentam aspectos relativos ao seu procedimento operacional.

Inicialmente, todas as amostras têm pesos iguais ($w_{i1} = 1/N$), o que faz com que o primeiro classificador seja treinado sobre os dados da maneira usual. Para cada iteração sucessiva $k = 2, \dots, K$, os pesos das amostras são modificados individualmente e o indutor é reaplicado sobre as novas amostras ponderadas. No $(k+1)$ -ésimo passo, as observações que tenham sido erroneamente interpretadas pelo estimador induzido no passo anterior, ou seja, f_k , têm seus pesos incrementados, enquanto as amostras classificadas corretamente por f_k têm seus pesos decrementados. Em ambos os casos, os pesos das amostras são rescalados por um fator $\exp(\alpha_k)$, isto é, em proporção à qualidade da hipótese f_k capturada pelo parâmetro (“qualidade” da hipótese) α_k . Assim, à medida que $k \rightarrow K$, as amostras mais difíceis estarão associadas a pesos mais eleva-

⁽¹¹⁾Além de AdaBoost.M1, referido por Friedman *et al.* como “AdaBoost Discreto” [197], existem várias outras versões de AdaBoost, como AdaBoost.M2, também destinada a problemas de classificação, e AdaBoost.R e AdaBoost.R2, ambas para problemas de regressão [128, 161].

Algoritmo 2 AdaBoost.M1 com reponderação para classificação

Entrada: $\mathcal{L}$ (indutor fraco), D_N (repositório de amostras $\{(\mathbf{x}_i, y_{d_i})\}_{i=1}^N$), $C = \{1, \dots, M\}$ (conjunto de rótulos das classes) e K (número de iterações).

Saída: $\Omega(\mathbf{z}) = \arg \max_{j \in C} \sum_{k: f_k(\mathbf{z})=j} \alpha_k$ (voto majoritário ponderado).

Computação:

```

 $k \leftarrow 1;$ 
for  $i = 1$  to  $N$  do
   $w_{ik} = 1/N;$ 
end for
while ( $k \leq K$ ) do
   $f_k \leftarrow \mathcal{L}(D_N, \{w_{1k}, \dots, w_{Nk}\});$ 
   $\epsilon_k \leftarrow \frac{\sum_{i=1}^N w_{ik} I(y_{d_i} \neq f_k(\mathbf{x}_i))}{\sum_{i=1}^N w_{ik}};$ 
   $\alpha_k \leftarrow \log \left[ \frac{1-\epsilon_k}{\epsilon_k} \right];$ 
  for  $i = 1$  to  $N$  do
     $w_{i(k+1)} \leftarrow w_{ik} \times \exp[\alpha_k \cdot I(y_{d_i} \neq f_k(\mathbf{x}_i))]$ 
  end for
end while

```

dos, passando a ter maior influência, pois, no treinamento dos últimos preditores. Para o cálculo da taxa de erro ponderada de cada classificador (ϵ_k) e dos novos pesos das amostras para o próximo preditor ($w_{i(k+1)}$), utiliza-se a função-auxiliar $I(\pi)$, que assume valor ‘1’ caso a condição π seja verdadeira e ‘0’, caso contrário.

Freund & Schapire [160, 161] estabeleceram um teorema pelo qual fica demonstrado, formalmente, que, quando todas as hipóteses fracas f_k usadas pelo AdaBoost apresentam uma taxa de erro levemente menor que 0,5, o erro de treinamento da hipótese final (combinação) cai a zero de forma exponencial. Contudo, tal teorema não garante que o erro de generalização sobre os dados de teste seja necessariamente pequeno [198]. Ou seja, o desempenho, em termos de generalização, do algoritmo AdaBoost.M1 dependerá diretamente da capacidade do indutor fraco empregado e/ou do uso de um conjunto de validação à parte [128]. De todo modo, mediante experimentos conduzidos por Breiman [62], pôde-se detectar que, quando o erro de treinamento e o erro de teste foram plotados como função do número de iterações k , o erro de teste continuou a decrescer mesmo após o erro de treinamento ter caído quase a zero. Esse fenômeno notável⁽¹²⁾, segundo Haykin [198], parece entrar em contradição direta com o princípio MDL [197] (ou Lâmina de Occam), uma vez que, no lugar de se usar máquinas simples, como apregoa tal princípio, a máquina agregada resultante da aplicação de *Boosting* se torna cada vez mais complexa à medida que novos modelos fracos são combinados.

Nos últimos anos, vários autores passaram a realizar comparações extensas e detalhadas acerca das vantagens relativas de *Bagging*, *Boosting* e algumas abordagens variantes, assumindo como indutores

⁽¹²⁾Para RNAs, por exemplo, a tendência é a de que o erro de teste (validação) caia, chegue em um mínimo, e depois suba em razão do conseqüente sobreajuste aos dados.

fracos (instáveis) árvores de decisão e modelos neurais [41, 63, 124, 128, 302, 306, 352]. Embora não haja um consenso definitivo sobre que abordagem seja superior, levando-se em conta todos os critérios possíveis (como acuidade, uso de recursos computacionais e estabilidade), a maioria das comparações evidencia que *Boosting* tende a gerar melhores resultados em termos da capacidade de generalização da máquina resultante⁽¹³⁾. Todavia, duas observações gerais importantes são sintetizadas por Mao [306]: (i) em alguns casos, técnicas de *Boosting* podem apresentar um desempenho significativamente melhor que *Bagging*, enquanto, em outros casos, tal desempenho pode ser substancialmente pior⁽¹⁴⁾, o que leva à conclusão de que o ganho na taxa de reconhecimento induzido por técnicas de *Boosting* exibe uma *alta variância*; e (ii) o ganho sobre estimadores individuais propiciado por *Bagging* tende a ser mais *consistente* (quando avaliado sobre várias instâncias de problemas) que aquele decorrente da aplicação de *Boosting*. A análise realizada por Opitz & Maclin [352] veio a confirmar o aspecto de “alta variância” levantado por Mao, pois indicou que o desempenho das versões de *Boosting* apreciadas (ARC-X4 e AdaBoost.M1) era dependente das características do problema em consideração. Por outro lado, conforme Drucker [128], a principal vantagem de *Bagging* sobre *Boosting* é que *Bagging* permite que cada máquina de aprendizado seja treinada independentemente das demais, abrindo espaço para implementações paralelas que venham a aliviar o tempo computacional.

4.1.4 Seleção de componentes

Em vez de se tentar assegurar de antemão encontrar os estimadores mais díspares, em termos de generalização, tendo em vista um número predefinido de componentes a serem agregados para formar uma *ensemble*, pode-se recorrer à estratégia alternativa de se partir de um farto elenco composto por variados estimadores-candidatos e *selecionar* aqueles mais apropriados para serem combinados. O objetivo, pois, de um operador de seleção é o de reduzir, *a posteriori*, o número máximo de falhas compartilhadas a serem exibidas pelos componentes de fato. Pode-se pensar o papel de um *seletor* de componentes de uma *ensemble* como similar àquele exercido por métodos de poda em RNAs — ver Seção 2.5. Ou seja, eliminar redundâncias desnecessárias desde que se mantenha (ou mesmo incremente) o nível de generalização da arquitetura resultante. Nesse contexto, Windeatt & Ardeshtir [490] conduziram um estudo empírico comparando os prós e contras de diferentes abordagens de poda de estimadores em problemas de classificação.

Tomando como base a Eq. (4.10), Zhou *et al.* [513, 515] empreenderam um desenvolvimento matemático interessante justificando formalmente a seleção adaptativa de componentes em *ensembles*, o qual é transcrito, em parte, a seguir. Considerando o operador de combinação “média simples”, que é adotado apenas por questão de facilidade de manipulação, (4.10) pode ser reescrita como

$$E_{ens} = \frac{1}{K^2} \sum_{k=1}^K \sum_{k^*=1}^K C_{kk^*}. \quad (4.14)$$

Assumindo que o k' -ésimo estimador seja removido do grupo, então o erro de generalização da nova *ensemble*

⁽¹³⁾Nas palavras de Breiman, inventor do *Bagging*, “[AdaBoost is the] best off-the-shelf classifier in the world” [62].

⁽¹⁴⁾Em outros poucos casos, pode ser até pior do que aquele exibido por instâncias de classificadores simples [352].

será:

$$E'_{ens} = \frac{1}{(K-1)^2} \sum_{\substack{k=1 \\ k \neq k'}}^K \sum_{\substack{k^*=1 \\ k^* \neq k'}}^K C_{kk^*}. \quad (4.15)$$

Levando em conta a diferença entre (4.14) e (4.15), obtém-se:

$$E_{ens} - E'_{ens} = \frac{2 \sum_{\substack{k=1 \\ k \neq k'}}^K C_{kk'} + E_{k'} - (2K-1)E_{ens}}{(K-1)^2},$$

onde $E_{k'}$ denota o erro de generalização do candidato retirado ($E_{k'}(\mathbf{x}_i) = f_{k'}(\mathbf{x}_i) - y_{d_i}$). É fácil comprovar que $E_{ens} > E'_{ens}$ caso a restrição

$$E_{ens} < \frac{1}{2K-1} \left(2 \sum_{\substack{k=1 \\ k \neq k'}}^K C_{kk'} + E_{k'} \right) \quad (4.16)$$

seja satisfeita, significando que a nova *ensemble* (i. e., com $f_{k'}$ omitido) tem maior precisão que a original. Finalmente, considerando, simultaneamente, (4.15) e (4.16), pode-se obter uma condição a ser adotada para se avaliar se um dado membro do repertório inicial de estimadores deve ser removido ou mantido:

$$(2K-1) \sum_{\substack{k=1 \\ k \neq k'}}^K \sum_{\substack{k^*=1 \\ k^* \neq k'}}^K C_{kk^*} < 2(K-1)^2 \sum_{\substack{k=1 \\ k \neq k'}}^K C_{kk'} + (K-1)^2 E_{k'}. \quad (4.17)$$

A conclusão a que Zhou *et al.* [513, 515] chegaram é que, após a fase de criação dos possíveis candidatos a componentes, em alguns casos, parece ser bastante lucrativo (em termos da capacidade de generalização) agrupar-se apenas *alguns* deles, omitindo aqueles satisfazendo a Eq. (4.17).

Antes desse resultado teórico de Zhou *et al.*, Perrone & Cooper [369] já haviam concebido uma estratégia heurística (gulosa) e construtiva dirigida à determinação do subconjunto ótimo de componentes a serem agrupados, considerando uma população de K estimadores-candidatos e o universo de $2^K - 1$ subconjuntos não-vazios possíveis⁽¹⁵⁾. No caso, em vez de ser de remoção, o critério adotado tenta justificar a *inclusão* de um candidato a componente final. A estratégia estipula a ordenação dos candidatos de acordo com suas taxas de erro (EQM), da menor para a maior, o que gera uma lista de índices $\mathcal{O} = [o_1, \dots, o_K]$. Inicia-se o processo de seleção com a *ensemble* mínima, ou seja, aquela composta exclusivamente pelo primeiro componente da lista, o_1 . Em seguida, o processo seletivo repetirá $K - 1$ vezes os seguintes passos: (1) introduz-se o candidato cuja posição na lista $\mathcal{O}$ seja igual ao índice da iteração atual (digamos, k^*) no grupo proveniente do passo anterior (i. e., *ensemble* provisória), dando origem a um novo grupo com, digamos, $K' \leq k^*$ membros; (2) verifica-se se a seguinte condição é satisfeita:

$$(2K' + 1)EQM[\Omega_{MEB}] > 2 \sum_{k \neq k^*} E[\Delta f_{k^*} \Delta f_k] + E[(\Delta f_{k^*})^2],$$

⁽¹⁵⁾Tal estratégia foi chamada de PCSe1 no projeto da Seção 8.2.

onde $EQM[\Omega_{MEB}]$ indica o EQM do estimador MEB formado pelos $K' - 1$ membros anteriores sobre um conjunto de dados à parte (conjunto de seleção), $E[\cdot]$ é o operador esperança matemática, e Δf_{k*} representa o erro (desvio residual) produzido pelo novo candidato (o_{k*}) em separado também sobre o conjunto de seleção⁽¹⁶⁾; (3) caso a condição do passo anterior seja satisfeita, insere-se o_{k*} definitivamente na *ensemble* e calcula-se o EQM do novo estimador MEB resultante sobre o conjunto de seleção; senão, o_{k*} deve ser descartado e K' deve ser decrementado de uma unidade.

4.1.5 Combinação de componentes

Após serem selecionados, os estimadores individuais estarão prontos para terem suas saídas agregadas de alguma forma de modo a que se possa produzir a saída da *ensemble*. Como visto, esse é o papel do combinador. Vários algoritmos de combinação (ver Apêndice B) estão disponíveis para serem adotados pelo projetista, os quais podem ser distinguidos uns dos outros de acordo com suas propriedades, tais como, (não-)linearidade, adaptabilidade e capacidade de ajuste dinâmico em função das condições do problema-alvo. Assim, pode-se diferenciar entre combinadores *estáticos*, os quais *não* fazem uso de informações extraídas ao longo do treinamento dos componentes (p. ex., sua acuidade sobre um conjunto de validação à parte) como forma de revelar suas habilidades particulares, e *adaptativos*, que lançam mão de informações desse quilate. Ademais, é comum se categorizar métodos de combinação de classificadores sob duas vertentes complementares [205, 262, 494]: uma baseada em *seleção*⁽¹⁷⁾ e outra baseada em *fusão*. O que se pressupõe, no primeiro caso, é que cada componente seja realmente um especialista em alguma região local do espaço de atributos. Quando um padrão x é submetido para ser classificado, ao estimador responsável pela vizinhança que envolve x é repassada a responsabilidade por rotulá-lo. Vale notar que, embora similar a sistemas modulares, conforme a caracterização feita no início deste capítulo, essa abordagem de seleção de classificadores continua *não* tendo acesso a informações extras retiradas dos dados. Exemplos de métodos baseados em seleção são o DCS.LA [494] e o CS [261], ambos apresentados no Apêndice B. Já os métodos firmados na fusão de classificadores assumem que todos os componentes são bem “indicados” para lidarem com todo o espaço de entradas e, por conseguinte, as decisões de todo o grupo devem ser levadas em consideração. A maioria dos combinadores segue essa abordagem mais cooperativa, tal como aquele calcado em gabaritos de decisão (DT), projetado por Kuncheva e colaboradores [263, 265]. Mais recentemente, Kuncheva [264] também propôs um esquema de combinação misto (CS + DT) que emprega uma estratégia baseada em inferência estatística e intervalos de confiança para chavear dinamicamente entre um tipo de abordagem e outro.

Kittler *et al.* [253] visualizam dois possíveis cenários de combinação de classificadores. Enquanto no primeiro cenário todos os classificadores lançam mão do mesmo tipo de representação para os padrões de

⁽¹⁶⁾Empregou-se Δf_{k*} no lugar de E_{k*} apenas para facilitar a apreensão da fórmula.

⁽¹⁷⁾É válido notar que a seleção de componentes aqui é feita *dinamicamente*, para cada padrão de entrada, *não* implicando o descarte definitivo de nenhum membro da *ensemble*. Ou seja, não se deve confundir o papel de combinadores baseados em seleção com aquele exercido por seletores propriamente ditos (mencionados na seção anterior), conquanto haja uma certa sobreposição entre eles — ambos os elementos realizam algum tipo de “escolha”.

entrada, no segundo cenário, cada classificador passa a “enxergar” o espaço de entradas sob um formato de codificação particular, provavelmente decorrente de uma interpretação física distinta das demais. Nesse último caso, a integração probabilística das saídas dos classificadores torna-se mais dificultosa, já que as probabilidades *a posteriori* geradas não podem ser consideradas como estimativas para o mesmo valor funcional, pois, conforme Kittler *et al.* [253], os classificadores passam a operar em espaços de medidas físicas com propriedades estatísticas não-conformativas.

Diferentes combinadores operam com diferentes tipos de informação representando as decisões coletadas dos componentes. Para problemas de classificação, Xu *et al.* [497] agruparam esses tipos de informação em três níveis de resolução: *abstrato*, *em ranking* e *de confiança*. No nível abstrato, somente a decisão (i. e., rótulo da classe) mais provável de cada classificador será levada em conta. Já no nível de confiança (também chamado de nível de medição), cada estimador gera um valor numérico (real) para cada rótulo possível indicando o seu grau de suporte (crença ou probabilidade) à hipótese de que a amostra sob análise pertença àquela classe em particular. (Para RNAs, por exemplo, tais valores correspondem às saídas de ativação normalizadas dos seus neurônios de saída.) Para o nível em *ranking*, cada estimador atribui um posto a cada classe, sendo que ao rótulo mais provável ele atribui o maior posto. Valores em *ranking* não podem ser usados isoladamente, uma vez que só fazem sentido quando analisados em conjunto; também não trazem consigo informações mais refinadas indicando, por exemplo, quão “distantes” (em termos de crença) estão os primeiro e segundo colocados [9]. Contudo, métodos em *ranking* podem ser usados para combinar não só decisões em *ranking* (ou medidas de confiança que possam ser facilmente convertidas em *rankings*) mas também informações em nível abstrato (votos) — para tanto, será necessário se “arredondar” as informações dos demais estimadores para que estas se tornem compatíveis com o classificador que gera a decisão menos detalhada [205]. Embora as combinações em nível abstrato façam uso do tipo menos apurado de decisão, são as mais freqüentemente empregadas por permitirem agregar virtualmente qualquer tipo de classificador, quer seja estatístico ou sintático [9]. Os vários métodos de votação se enquadram nessa condição [32, 40, 273]. Por fim, os métodos baseados em confiança podem agregar *qualquer* tipo de classificador que gere medidas de suporte como saída, desde que tais medidas possam ser traduzidas para uma mesma “língua franca” — p. ex., um estimador que gere suas decisões em termos de distâncias não pode ser agregado diretamente com outro classificador que produza saídas de natureza puramente probabilística [9].

Em se tratando especificamente de problemas de regressão, Hashem [196], tomando por base o método de *ensemble* generalizada de Perrone & Cooper [369], formulou quatro modelos de combinação linear ótima (chamados de CLO-EQM) com o propósito de estimar os pesos de agregação (α_k 's) a partir dos dados de entrada, tendo em vista a minimização do EQM. As quatro variações de combinação resultantes divergem apenas em dois aspectos: (1) a inclusão de uma parcela constante (com papel similar àquele produzido por um neurônio de polarização); e/ou (2) a inclusão da restrição de que a soma dos pesos deva ser igual à unidade ($\sum_k \alpha_k = 1$). Dentre as quatro variações propostas, aquela *com* o termo constante e *sem* restrição aos pesos foi a que apresentou melhor desempenho na maioria das simulações realizadas por Hashem [196]. Tal variação foi utilizada nos experimentos de regressão ou previsão de séries temporais conduzidos tanto para *ensembles heterogêneas de SVMs* [284, 291] como para *ensembles de ANFIS* [285], ambos os projetos

discutidos no Cap. 8.

Finalmente, é válido comentar acerca da distinção comumente feita entre combinadores *ajustáveis* e *fixos*. Aqueles pertencentes à segunda classe não requerem ajuste (p. ex., via treinamento) de alguns parâmetros configuráveis, ao passo que os da primeira sim. O preço que se paga pelos ganhos de desempenho decorrentes da flexibilidade inerente aos combinadores ajustáveis é o aumento inevitável na carga computacional [227]. Dentre os combinadores ajustáveis disponíveis na literatura, a abordagem de *empilhamento* (*Stacking*) de estimadores, proposta por Wolpert [492], é a mais comumente citada e estendida [450, 500].

Em verdade, *Stacking* cobre duas etapas importantes relativas à construção de *ensembles*: a preparação dos dados para treinamento dos estimadores (similar, portanto, a *Bagging* e *Boosting*) e a sua combinação propriamente dita. Para tanto, tal *framework* de aprendizado adota ou o método de validação cruzada ou o método “deixe-um-de-fora” (ver Apêndice A) para selecionar os dados de treinamento (do “nível-0”), ao mesmo tempo que explora a estratégia de usar um segundo nível de aprendizado (“nível-1”) contendo um ou mais preditores (i. e., “generalizadores de segundo nível”) para combinar os resultados a serem gerados pelos componentes da *ensemble* (estes chamados de “generalizadores de nível de base”)⁽¹⁸⁾. Desse modo, as informações fornecidas para o treinamento dos generalizadores de base ficam sendo provenientes de múltiplas partições do repositório de dados original $\mathcal{D}_N$. Assumindo que sejam Q partições no total, pode-se definir $\mathcal{D}_N^q$ e $\mathcal{D}_N^{(-q)} = \mathcal{D}_N - \mathcal{D}_N^q$ como sendo, respectivamente, os conjuntos de teste e de treinamento relativos a q -ésima partição, $q = 1, \dots, Q$. Considerando K indutores distintos⁽¹⁹⁾, serão gerados K modelos para cada partição, sendo que $f_k^{(-q)}$ indica o modelo gerado pelo indutor k para a q -ésima partição (isto é, sobre $\mathcal{D}_N^{(-q)}$). Deve-se notar que o número de componentes da *ensemble*, K , e o número total de partições, Q , não precisam ser necessariamente iguais. Já os generalizadores de segundo nível são treinados com base nas saídas produzidas pelos generalizadores de nível-0 em cada uma das Q partições. O espaço de entradas $\mathcal{E}$ de um generalizador de segundo nível ($\tilde{f}$) corresponde a uma combinação das saídas (decisões) dos componentes do nível-0 (sem levar em conta os atributos da amostra original), ou seja, $\mathcal{E} = \mathcal{Y}_1 \times \dots \times \mathcal{Y}_K$, com $\mathcal{Y}_k$ correspondendo ao espaço de saídas do k -ésimo classificador. Já o espaço de saídas de um generalizador de segundo nível corresponde ao espaço de saídas original (saída desejada). Para cada amostra $\mathbf{x}_n$ do conjunto de teste $\mathcal{D}_N^{(q)}$ relativo à partição q , uma nova amostra $\mathbf{z}_n$ será montada contendo as predições geradas pelos generalizadores de base $f^{(-q)}$ como atributos de entrada e a saída desejada para $\mathbf{x}_n$ como seu rótulo de saída. O conjunto dessas novas amostras $\mathbf{z}_n$ formará o repositório de treinamento para o segundo nível, $\mathcal{D}_N^{CV}$. A Fig. 4.5 traz um esboço pictórico desse processo operacional.

Segundo Sharkey [427], é possível encarar outros métodos de combinação fixos (p. ex., média) como subcasos do método de empilhamento, embora nesses casos o generalizador de segundo nível não será ajustável. Ting & Witten [450] realizaram um estudo prospectivo comparando diferentes instâncias de *Stacking* com instâncias de modelos gerados por técnicas de reamostragem de dados (*Bagging* e *Arcing*), tendo como referencial problemas de classificação. Dois aspectos importantes foram investigados em maior deta-

⁽¹⁸⁾É possível se incrementar ainda mais esse modelo hierárquico, gerando generalizadores de níveis cada vez mais elevados [162].

⁽¹⁹⁾Pode-se recorrer a um mesmo indutor, desde que ele seja instável (p. ex., sensível a condições iniciais).

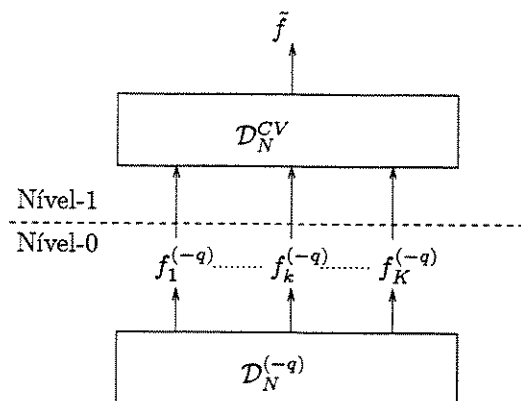


Fig. 4.5: Esquemático do processo de validação cruzada relativo a q -ésima partição do método de empilhamento. Um novo repositório $\mathcal{D}_N^{CV}$ estará disponível ao final das Q iterações de modo a que se possa gerar o meta-modelo de combinação $\tilde{f}$. (Fonte: Ting & Witten [450].)

lhe: (i) quais os tipos de estimadores seriam mais adequados para serem aplicados como generalizadores do nível-1; e (ii) quais os tipos de atributos a serem usados para treinar tais generalizadores de segundo nível (ou seja, qual o tipo de informação a ser extraído dos classificadores do patamar inferior). Com base nos resultados, os autores argumentaram que o uso de decisões probabilísticas (i. e., graus de suporte às classes) é o mais indicado, pois permite ao generalizador de segundo nível mensurar com maiores detalhes as confianças dos classificadores quando da análise de suas decisões. Dentre os três indutores avaliados, o mais indicado para gerar os meta-modelos de combinação foi uma técnica de regressão linear multiresposta, que apresenta características similares a RNAs — modelos neurais não foram avaliados, mas costumam ser também adotados [162]. Por sua vez, mais recentemente, Yang *et al.* [500] propuseram um novo esquema de combinação, chamado de *ensembles de redes neurais multiestágios* (ERNM) e fundamentado no método de empilhamento. Os autores advogaram o uso de RNAs como meta-modelos por conta da sua capacidade de aproximação universal, o que possibilita a representação das complexidades inerentes à função de combinação dos estimadores. Ademais, por serem modelos baseados em *thresholds* quantitativos, RNAs são particularmente adequadas para gerarem pesos de combinação, indicando os graus de confiança sobre os preditores do nível de base. Contudo, em vez de estar atrelada à validação cruzada como método de particionamento dos dados para geração dos preditores de nível-0 (também RNAs), a abordagem ERNM vislumbra também a possibilidade de se usar técnicas correlatas para esse fim, tais como *Bagging*, *Boosting*, injeção de ruído e alocação aleatória de amostras, garantindo, de antemão, a introdução de níveis de diversidade aos estimadores da primeira camada da *ensemble*.

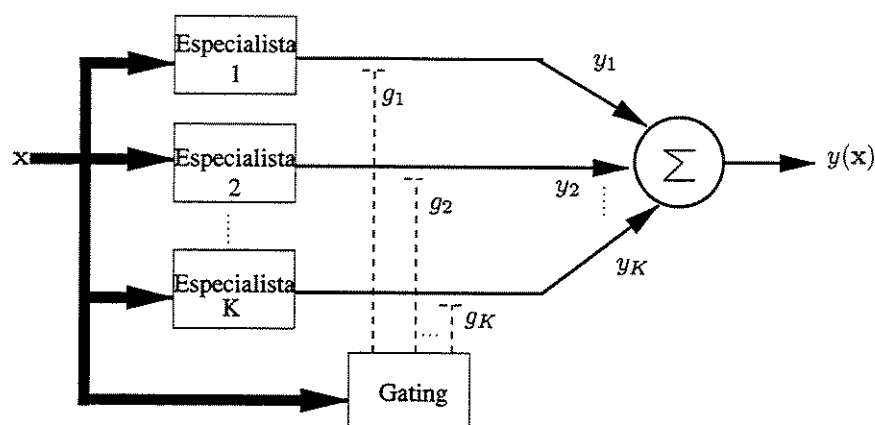


Fig. 4.6: Diagrama de blocos da arquitetura ME. Redes especialistas competem entre si para aprenderem o padrão de treinamento corrente (x), ao passo que a rede *gating* intermedia essa competição via suas portas. (Fontes: Haykin [198] e Jacobs [226].)

4.2 Misturas de especialistas

De acordo com Sharkey [425, 427], uma das questões mais importantes no que concerne à construção de sistemas modulares efetivos diz respeito a como se decompor, de maneira apropriada, uma tarefa complexa em várias subtarefas de menor granulosidade, de modo a que se obtenha ganhos substanciais tanto em termos de desempenho do sistema geral (eficácia) como em termos de rapidez na convergência do processo de aprendizado dos módulos especialistas (eficiência). Duas estratégias distintas para lidar com esse requisito foram levantadas pela autora: *explícita* e *automática*. A opção por decomposições explícitas (ou seja, manuais) usualmente requer, por parte do projetista, um profundo conhecimento da tarefa em mãos, uma vez que o modo como a tarefa é decomposta terá impacto direto sobre a “qualidade” dos módulos a serem gerados. A arquitetura conexionista Meta-Pi, proposta por Hamshire & Waibel [189], constitui um exemplo de sistema modular derivado de um processo de decomposição explícita, uma vez que os seus módulos (subredes) foram projetados para operar com diferentes fontes de dados (sensores), estas espacialmente distribuídas e conhecidas de antemão. Por outro lado, abordagens automáticas tentam extrair dinamicamente informações importantes acerca da natureza (física ou estatística) do problema como meio de se conduzir a sua partição em subtarefas. Dentre os modelos adaptativos de aprendizado supervisionado concordantes com tal filosofia de “divisão-e-conquista”, passa-se na sequência a caracterizar com mais detalhes o *framework* probabilístico conhecido como misturas de especialistas [225, 226, 239, 328, 368, 469, 472, 496], este concebido originalmente por Jacobs no curso de seu doutorado [222].

A hipótese neurofisiológica que sustenta o *framework* de MEs é a de que a existência combinada de correspondências “estrutura-função” e de competição entre os módulos neurais traz a reboque a emergência de especializações funcionais para esses módulos [224]. Essa teoria está embasada em duas noções relevantes. A primeira é a de que existem no cérebro humano correspondências “estrutura-função”: porque diferentes

regiões do cérebro apresentam diferentes propriedades estruturais (p. ex., diferentes padrões de conectividade entre seus neurônios), tais regiões devem ser mais apropriadas para arcar com diferentes tipos de funções. A segunda noção é a de que as regiões do cérebro competem entre si pela habilidade em realizar um determinado conjunto de tarefas. Tais regiões se tornam funcionalmente especializadas devido à competição; quer dizer, diferentes regiões aprendem a realizar diferentes funções. Todavia, a segunda noção é influenciada diretamente pela primeira. Ou seja, tais processos competitivos costumam ser direcionados pelas correspondências “estrutura-função”, fazendo com que cada região *tenda* a vencer as competições por aquelas funções para as quais a sua estrutura a torna particularmente mais adequada que as demais.

A mecânica operacional por trás de uma arquitetura típica de ME, esta ilustrada na Fig. 4.6, tenta emular computacionalmente o fenômeno biológico supracitado. Destarte, tal arquitetura consiste de dois tipos de módulos neurais: redes especialistas e uma rede *gating*. Os K módulos especialistas competem entre si pelo aprendizado de diferentes padrões de treinamento $(n) = \{\mathbf{x}^{(n)}, \mathbf{t}^{(n)}\}$; como resultado, diferentes especialistas aprenderão a realizar funções computacionais distintas. Já a única rede *gating* tem o papel de mediar esse processo competitivo, particionando o espaço de entradas e atribuindo os especialistas às regiões resultantes. Esse procedimento de decomposição automática, o qual é influenciado pelos próprios dados amostrais — ou seja, assim como os especialistas, a *gating* também tem acesso aos sinais de entrada —, pode apresentar “ambigüidades” (sobreposições) entre as regiões, fazendo com que um mesmo ponto do espaço de entradas pertença, simultaneamente, a mais de uma região.

Os módulos especialistas podem exibir padrões arbitrários de conectividade, sendo que, originalmente [226], eles foram projetados para gerar médias gaussianas condicionadas à entrada via modelos lineares generalizados [197], tendo seus valores combinados via *gating* logística linear, cuja função de ativação é dada pela função *soft-max* (discutida, brevemente, à página 20 e logo a seguir). Posteriormente, Jacobs & Jordan [225] passaram a levar em conta também redes *feedforward* de camada simples ou de múltiplas camadas para atuarem como módulos especialistas, algo explorado em seguida por Weigend *et al.* [472] quando da caracterização das MEs engatilhadas (ver Subseção 4.2.3). Em se tratando de um especialista linear de camada única e com variância constante, a sua saída $y_k(\mathbf{x})$ pode ser representada como $y_k(\mathbf{x}) = \theta_k^T \mathbf{x}$, $k = 1, \dots, K$, onde θ_k denota o seu vetor de parâmetros livres. Para problemas de classificação binária, a saída do k -ésimo especialista pode ser dada pela função sigmoideal binária (ver Tab. 2.1). Já a rede mediadora é condicionada a ter o seu número de unidades de saída — chamadas de *gates* (portas) — igual ao número de especialistas. Ademais, as saídas de ativação de tais portas $g_k(\mathbf{x})$, $k = 1, \dots, K$ devem ser não-negativas (i. e., $g_k \geq 0$) e, quando somadas, devem se igualar à unidade (i. e., $\sum_k g_k = 1$). Para tanto, geralmente é adotada a função de ativação não-linear⁽²⁰⁾:

$$g_k(\mathbf{x}) = \frac{\exp(u_k)}{\sum_{k^*} \exp(u_{k^*})}, \quad (4.18)$$

⁽²⁰⁾ A transformação exponencial “normalizada” produzida pela Eq. (4.18) pode ser encarada como uma generalização multi-entradas da função logística, em que fica preservada a ordenação (*ranking*) dos valores de entrada. Ao mesmo tempo, tal função pode ser vista como uma generalização diferenciável da operação *winner-takes-all* (“o vencedor leva tudo”), que escolhe o nível de ativação mais elevado como saída [198].

onde se assume que o padrão $\mathbf{x}$ seja D_x -dimensional e que $u_k = \sum_d^{D_x} x_d \nu_{kd}$ simbolize o grau de ativação da k -ésima porta, com ν_k representando o vetor de parâmetros livres associado a essa porta. Por fim, o vetor de saídas do sistema é obtido via a média ponderada das saídas dos especialistas, cujos pesos de combinação são dados pelas portas da *gating*:

$$\mathbf{y}(\mathbf{x}) = \sum_{k=1}^K g_k(\mathbf{x}) \mathbf{y}_k(\mathbf{x}). \quad (4.19)$$

Matematicamente, misturas de especialistas podem ser caracterizadas como um modelo de probabilidade especial conhecido como *modelo de mistura condicional de densidades* [239, 368, 381]. O processo de configuração dos seus parâmetros livres $\Theta = \{\theta, \nu\}$ (onde $\theta = \{\theta_k\}_{k=1}^K$ e $\nu = \{\nu_k\}_{k=1}^K$) combina aspectos de aprendizado competitivo (supervisionado) e de aprendizado associativo (não-supervisionado). Com efeito, a parte supervisionada do treinamento fica por conta de cada especialista, que deve aprender a estimar a média condicionada a cada nova observação de entrada (p. ex., o próximo valor de uma série temporal). Já a parte não-supervisionada diz respeito ao treinamento da *gating*, que deve aprender a revelar os processos (p. ex., regimes em séries temporais) embutidos nos próprios dados amostrais, de tal forma a atribuir a responsabilidade da próxima estimação ao especialista mais apropriado.

Antes de discorrer mais sobre uma possível interpretação probabilística de MEs (Subseção 4.2.1), é pertinente apresentar uma descrição mais intuitiva, baseada naquela encaminhada mais recentemente por Jacobs [224]. Os parâmetros livres das redes especialistas e *gating* devem ser ajustados simultaneamente durante o treinamento. Isso quer dizer que há um forte acoplamento entre o que a *gating* e os especialistas aprendem, algo que deve ser tratado adequadamente pela abordagem de treinamento utilizada. O processo de configuração é iterativo, sendo que a saída $\mathbf{y}_k$ de cada especialista k é comparada com o valor-alvo atual $\mathbf{t}^{(n)}$ a cada passo. Aquele especialista que produzir a saída mais próxima do alvo é chamado de *vencedor* da competição por $\mathbf{t}^{(n)}$; os outros são chamados de *perdedores*. Finda a competição, cada especialista receberá uma certa quantidade de informação de treinamento proporcional ao seu desempenho relativo sobre $\mathbf{t}^{(n)}$. O vencedor receberá uma grande parcela, a fim de que possa aprender mais ainda sobre as características da amostra atual, ao passo que os perdedores receberão pouca ou nenhuma parcela de informação de treinamento. A rede mediadora, por sua conta, terá acesso ao desempenho de cada especialista sobre $\mathbf{t}^{(n)}$. Daí, ela ajustará os seus parâmetros livres (p. ex., pesos) em conformidade com esses desempenhos, de sorte que, quando o mesmo padrão de entrada (ou similar) aparecer novamente para ser predito, a porta correspondente ao especialista vencedor esteja altamente ativada (próximo de um), enquanto as portas associadas aos perdedores tenham baixo nível de ativação (próximo de zero).

Esse processo de aprendizado exibe um efeito interessante de realimentação positiva que força com que diferentes especialistas aprendam a arcar com diferentes subtarefas. Esse efeito se fundamenta no fato de que, em geral, padrões de treinamento relativos a uma mesma subtarefa compartilham de uma “estrutura” intrínseca comum, ao tempo que padrões relativos a subtarefas distintas apresentam propriedades estruturais dessemelhantes. Ao vencer a competição por um dado padrão $\mathbf{t}^{(n)}$, o especialista k demonstra possuir, pelo menos em parte, algum “conhecimento” acerca da estrutura da subtarefa a que $\mathbf{t}^{(n)}$ pertence.

Conseqüentemente, no futuro, existirá uma grande *chance* de que ele também vença as competições pelos padrões remanescentes daquela subtarefa. Desse modo, o k -ésimo módulo de aprendizado se tornará cada vez mais especialista (daí o nome) na subtarefa em questão. Por outro lado, como decorrência dessa especialização, esse módulo *tenderá* a apresentar baixo desempenho quando do tratamento de padrões relativos a outras subtarefas — a menos que estas sejam bem similares à que ele se especializou. Por conseguinte, outras máquinas de aprendizado do elenco passarão a se especializar nessas outras subtarefas e, dessa forma, o sistema passará a lidar, de maneira efetiva e eficiente, com toda a tarefa complexa original.

Essa abordagem de decomposição automática proporcionada pelas MEs tornou-se alvo de extensas experimentações ao longo da década passada e começo desta, mostrando-se particularmente adequada para (1) a atribuição de especialistas a diferentes regimes de séries temporais caóticas, não-lineares e/ou não-estacionárias [141, 304, 381, 441, 472]; (2) a modelagem e identificação (inclusive, para fins de controle) de mapeamentos arbitrários (multivariáveis e descontínuos) de E/S [225, 286, 346, 381]; e (3) o reconhecimento escalável de padrões em problemas cujo tamanho da dimensionalidade de entrada varia de moderado a elevado [187, 329, 469].

Na próxima subseção, é encaminhada uma interpretação probabilística de MEs, com base na exposição feita por Waterhouse [469]. Na Subseção 4.2.2, passa-se a contemplar uma versão do algoritmo EM (*Expectation Maximization*) — este proposto originalmente por Dempster *et al.* [120] e geralmente aplicado a modelos de misturas *não-condicionais* [316, 451] — destinada ao treinamento desacoplado de redes especialistas e *gating* [239, 240]. O material dessa subseção fundamentou-se diretamente naquele encaminhado por Moerland [328] quando da análise unificada de diferentes abordagens dirigidas ao treinamento de MEs. Finalmente, apresentar algumas variações de MEs, evidenciando aquelas orientadas para problemas de previsão de séries temporais, constitui o objetivo da Subseção 4.2.3.

4.2.1 Interpretação probabilística de MEs

A abordagem ME pode ser alvo de uma interpretação probabilística na medida em que for considerada como um modelo especial de mistura condicionada à entrada⁽²¹⁾ [240]. Sob essa perspectiva, assume-se que os dados amostrais sejam gerados por uma *série* de processos estatísticos. Assume-se também que cada amostra $(n) = \{\mathbf{x}^{(n)}, \mathbf{t}^{(n)}\}$ seja decorrente da aplicação de um único processo k , sendo este “selecionado” por uma densidade de probabilidade multinomial $p(z)$. Assim, $z_k^{(n)}$ alude à decisão de “escolher o k -ésimo processo para gerar a n -ésima amostra”. Analisando o procedimento operacional de MEs descrito anteriormente, pode-se depreender que os módulos especialistas assumem a função de modelar os distintos processos-geradores dos dados, ao passo que à rede mediadora cabe o papel de modelar $p(z)$. Essa visão probabilística é capturada de maneira simplista pela Fig. 4.7, a qual ilustra uma possível rede de crenças associada ao modelo de misturas

⁽²¹⁾Misturas de especialistas distinguem-se de modelos convencionais de misturas gaussianas [316, 451] pelo fato de que cada coeficiente da mistura (i. e., cada porta da *gating*) passa a ser encarado como uma função particular do sinal de entrada.

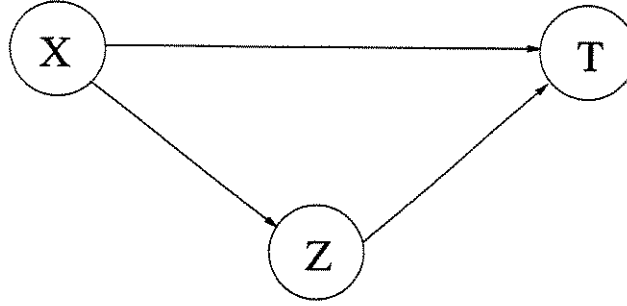


Fig. 4.7: Interpretação, via rede de crenças, do mecanismo de mistura associado a MEs. A estimação do alvo T depende tanto da entrada X como da variável aleatória multinomial Z . (Fonte: Waterhouse [469].)

de especialistas. Por esse esboço, fica evidente que, segundo MEs, o alvo T é visto como dependente tanto da entrada X como da variável aleatória multinomial Z .

Formalmente, essa interpretação de modelo de mistura condicional dada a MEs pode ser sintetizada a partir da seguinte expressão, a qual provê a probabilidade do valor-alvo $\mathbf{t}^{(n)}$ condicionada à entrada $\mathbf{x}^{(n)}$ e ao conjunto de parâmetros livres de toda a arquitetura $\Theta = \{\theta, \nu\}$:

$$p(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}, \Theta) = \sum_{k=1}^K p(m_k|\mathbf{x}^{(n)}, \nu) p(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}, \theta_k, m_k), \quad (4.20)$$

sendo que m_k faz alusão ao k -ésimo componente da mistura. Por uma questão de simplificação notacional, pode-se reescrever a expressão acima sob uma nova roupagem, em que ficam omitidas as dependências para com os parâmetros livres [329]:

$$p(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}) = \sum_{k=1}^K g_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}), \quad (4.21)$$

onde $\phi_k(\mathbf{t}^{(n)}|\mathbf{x}^{(n)})$ passa a representar a densidade condicional do vetor-alvo $\mathbf{t}^{(n)}$ para o k -ésimo especialista, enquanto $g_k(\mathbf{x}^{(n)})$ deve ser interpretado como a probabilidade *a priori* de se atribuir $\mathbf{x}^{(n)}$ ao processo modelado por k . Uma vez que as portas da *gating* sempre somam um (Eq. (4.18)) e que $\phi_k(\cdot)$'s são densidades de probabilidade, então é certo afirmar que $\int_{-\infty}^{+\infty} p(\mathbf{t}|\mathbf{x}) = 1$.

O formato da densidade condicional $\phi_k(\cdot)$ varia de acordo com o tipo de tarefa em vista. Para o caso de regressão, geralmente $\phi_k(\cdot)$ assume a forma de uma densidade condicional gaussiana multidimensional, dada por [53]:

$$\phi_k^{gs}(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}) = \frac{1}{(2\pi)^{(D_t/2)}} \exp\left(-\frac{\|\mathbf{t}^{(n)} - \mathbf{y}_k(\mathbf{x}^{(n)}, \theta_k)\|^2}{2}\right), \quad (4.22)$$

onde D_t indica a dimensionalidade de $\mathbf{t}^{(n)}$. Para o caso de classificação, $\phi_k(\cdot)$ pode assumir a forma de uma densidade multinomial, formulada como [328]:

$$\phi_k^{mn}(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}) = \prod_{j=1}^M \left(y_{kj}^{(n)}(\mathbf{x}^{(n)}, \theta_k)\right)^{t_j^{(n)}},$$

onde $y_{kj}^{(n)}(\mathbf{x}^{(n)}, \theta_k)$ denota a j -ésima saída do especialista k para o n -ésimo padrão, em consonância com o formato de codificação “1-of- M ” — ver Subseção 2.4.1.

Com a definição da Eq. (4.20), torna-se propício adotar uma formulação baseada no critério de máxima verossimilhança⁽²²⁾ para fins de estimação dos parâmetros livres do modelo ME, o que conduz a

$$L(\Theta, \mathbf{t}, \mathbf{x}) = \prod_n p(\mathbf{t}^{(n)}, \mathbf{x}^{(n)} | \Theta) = \prod_n p(\mathbf{t}^{(n)} | \mathbf{x}^{(n)}, \Theta) p(\mathbf{x}^{(n)}). \quad (4.23)$$

Quando do treinamento de uma instância de ME, em vez de tentar maximizar diretamente (4.23), é mais comum que se tente minimizar o negativo do logaritmo dessa função plausibilidade sobre todo o conjunto de amostras, o que remete à seguinte função-custo (energia)⁽²³⁾:

$$\begin{aligned} E &= - \sum_{n=1}^N \ln p(\mathbf{t}^{(n)} | \mathbf{x}^{(n)}) \\ &= - \sum_{n=1}^N \ln \sum_{k=1}^K g_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{t}^{(n)} | \mathbf{x}^{(n)}). \end{aligned} \quad (4.24)$$

Essa interpretação probabilística de MEs possibilita a aplicação direta do algoritmo EM [120], com vista ao desacoplamento dos ajustes feitos quando do treinamento simultâneo da rede *gating* e dos módulos especialistas. A próxima subseção trata desse assunto.

4.2.2 Treinamento de misturas de especialistas via EM

O trabalho original de Jacobs *et al.* [222, 226] empregou uma implementação do algoritmo de descida do gradiente para o treinamento da arquitetura ME. Posteriormente, Jordan & Jacobs [239] e Jordan & Xu [240] desenvolveram e analisaram uma versão customizada do algoritmo EM voltada para modelos de misturas condicionadas à entrada. Segundo Moerland [328], quando comparada à implementação baseada no gradiente do erro, essa nova versão do algoritmo EM possibilita um melhor desacoplamento entre as diferentes densidades-componentes (especialistas) ao longo do treinamento do sistema, implicando ganhos em termos da taxa de convergência.

A hipótese básica por trás do algoritmo EM é que a maximização da função de verossimilhança L (Eq. (4.23)) pode ser bastante simplificada caso se assuma que cada observação (n) esteja associada exatamente a um único processo-gerador modelado por um dado especialista. Ou seja, assume-se que os especialistas sejam mutuamente exclusivos: a probabilidade de dois especialistas distintos gerarem a mesma

⁽²²⁾ Estimadores de máxima verossimilhança partem do pressuposto de que os valores dos parâmetros livres (no caso, Θ) sejam fixos, porém desconhecidos, e consideram que a melhor estimativa desses parâmetros seja aquela que maximiza a probabilidade de se obter as amostras observadas (no caso, $p(\mathbf{t}^{(n)}, \mathbf{x}^{(n)})$) [130].

⁽²³⁾ O termo $p(\mathbf{x}^{(n)})$ em (4.23) foi removido quando da definição da função-erro visto que não depende dos parâmetros do modelo [328].

observação (n) , dada a entrada correspondente $\mathbf{x}^{(n)}$, é nula [472]. Tal idéia passa a ser facilmente implementada ao se lançar mão de um conjunto de *variáveis faltantes* (*latentes*) $z_k^{(n)} (k = 1, \dots, K; n = 1, \dots, N)$, cuja interpretação é dada a seguir:

$$z_k^{(n)} = \begin{cases} 1 & \text{se a observação } (n) \text{ foi gerada pelo especialista } k \\ 0 & \text{caso contrário} \end{cases}$$

Na presença de ambos os tipos de variáveis, observáveis $(\mathbf{t}^{(n)}, \mathbf{x}^{(n)})$ e não-observáveis $(\mathbf{z}^{(n)} = \{z_k^{(n)}\}_{k=1}^K)$, e assumindo a independência estatística dos erros de medição de cada amostra (n) , pode-se definir a *função densidade de probabilidade condicional completa* como sendo:

$$p(\mathbf{t}^{(n)}, \mathbf{z}^{(n)} | \mathbf{x}^{(n)}) = \sum_{k=1}^K z_k^{(n)} \left(g_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{t}^{(n)} | \mathbf{x}^{(n)}) \right) = \prod_{k=1}^K \left(g_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{t}^{(n)} | \mathbf{x}^{(n)}) \right)^{z_k^{(n)}}.$$

Ao se substituir essa expressão em (4.24), obtém-se a nova *função-custo completa*, dada por:

$$E_c = - \sum_{n=1}^N \sum_{k=1}^K z_k^{(n)} \ln \left(g_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{t}^{(n)} | \mathbf{x}^{(n)}) \right). \quad (4.25)$$

Pode-se verificar que a introdução das variáveis latentes permitiu o deslocamento do logaritmo para dentro do segundo somatório, o que vem a possibilitar a decomposição do problema original em vários subproblemas de minimização, cada um alocado a um especialista. Contudo, deve-se ter em mente que, normalmente, não se sabe em princípio qual é a verdadeira distribuição associada ao vetor $\mathbf{z} = \{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N)}\}$ de variáveis faltantes. Disso, o próximo “ingrediente” importante por trás do algoritmo EM diz respeito a um procedimento iterativo que se reveza entre dois passos bem definidos e complementares, chamados de *passo-E* e *passo-M*. No primeiro, calcula-se a esperança de E_c (Eq. (4.25)) com relação às variáveis latentes e com base nos valores atuais dos parâmetros da arquitetura (Θ) . Já o segundo passo tem o papel de minimizar E_c , ou seja, maximizar a verossimilhança completa. Esses dois passos são descritos mais formalmente a seguir⁽²⁴⁾.

4.2.2.1 Passo-E

A esperança de E_c pode ser escrita como⁽²⁵⁾:

$$\mathcal{E}(E_c) = - \sum_{n=1}^N \sum_{k=1}^K \mathcal{E}(z_k^{(n)}) \ln \left(g_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{t}^{(n)} | \mathbf{x}^{(n)}) \right), \quad (4.26)$$

onde os valores esperados das variáveis faltantes são obtidos como (via uso da regra de Bayes):

$$h_k^{(n)} \equiv \mathcal{E}(z_k^{(n)}) = P(z_k^{(n)} = 1 | \mathbf{t}^{(n)}, \mathbf{x}^{(n)}) = \frac{p(\mathbf{t}^{(n)} | z_k^{(n)} = 1, \mathbf{x}^{(n)}) P(z_k^{(n)} = 1 | \mathbf{x}^{(n)})}{p(\mathbf{t}^{(n)} | \mathbf{x}^{(n)})}. \quad (4.27)$$

⁽²⁴⁾ Uma análise teórica bem detalhada acerca dos aspectos de convergência inerentes a essa nova versão do algoritmo EM pode ser encontrada no trabalho de Jordan & Xu [240].

⁽²⁵⁾ Adotou-se o símbolo $\mathcal{E}$ para denotar o operador esperança matemática com o intuito de evitar uma possível confusão com a função-custo E_c .

Usando a equação de interpretação probabilística de MEs ((4.20) e (4.21)) e definindo

$$\pi_k(\mathbf{x}^{(n)}, \mathbf{t}^{(n)}) = \frac{g_k \phi_k}{\sum_{k^*} g_{k^*} \phi_{k^*}}, \quad (4.28)$$

(4.27) passa a dar origem a:

$$h_k^{(n)} = \frac{g_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{t}^{(n)} | \mathbf{x}^{(n)})}{\sum_{k^*} g_{k^*}(\mathbf{x}^{(n)}) \phi_{k^*}(\mathbf{t}^{(n)} | \mathbf{x}^{(n)})} = \pi_k(\mathbf{x}^{(n)}, \mathbf{t}^{(n)}). \quad (4.29)$$

Finalmente, substituir (4.29) em (4.26) provê:

$$\mathcal{E}(E_c) = - \sum_{n=1}^N \sum_{k=1}^K \pi_k(\mathbf{x}^{(n)}, \mathbf{t}^{(n)}) \ln(g_k(\mathbf{x}^{(n)})) - \sum_{n=1}^N \sum_{k=1}^K \pi_k(\mathbf{x}^{(n)}, \mathbf{t}^{(n)}) \ln(\phi_k(\mathbf{t}^{(n)} | \mathbf{x}^{(n)})). \quad (4.30)$$

O primeiro termo de (4.30) pode ser interpretado como a entropia intrínseca ao processo de se atribuir as amostras aos especialistas. Esse custo é mínimo quando as regiões alocadas aos especialistas não apresentam sobreposição e cresce em proporção ao número de padrões compartilhados pelos especialistas. Já o segundo termo assume a forma geral de um problema de máxima verossimilhança ponderada, cujos pesos são dados pelos π_k 's; isso implica que os especialistas importantes para lidar com uma dada amostra sejam aqueles com grandes valores para a probabilidade *a posteriori* definida em (4.28). Ambos os termos de (4.30) podem ser minimizados separadamente no decorrer de um passo-M.

4.2.2.2 Passo-M

Como dito, o papel deste passo é o de minimizar a função-custo completa dada em (4.25) com respeito ao conjunto de parâmetros Θ . Conforme Moerland [329], “essa função-erro $[E_c]$ pode ser interpretada como a soma de uma parte não-supervisionada, que estimula a formação de boas “regras de decisão” sobre que densidade usar para tratar cada nova observação (papel da *gating*), com uma parte supervisionada, que visa à predição correta do valor-alvo (papel dos especialistas)”. O termo a ser minimizado para a *gating* é:

$$E_g = - \sum_{n=1}^N \sum_{k=1}^K \pi_k(\mathbf{x}^{(n)}, \mathbf{t}^{(n)}) \ln(g_k(\mathbf{x}^{(n)})), \quad (4.31)$$

ao passo que, para cada especialista k , a parcela a ser otimizada é:

$$E_e = - \sum_{n=1}^N \sum_{k=1}^K \pi_k(\mathbf{x}^{(n)}, \mathbf{t}^{(n)}) \ln(\phi_k(\mathbf{t}^{(n)} | \mathbf{x}^{(n)})). \quad (4.32)$$

Este passo pode assumir diferentes implementações dependendo do tipo de modelo adotado para a rede mediadora e para os especialistas. Ao fazer uso, tanto para a *gating* como para os especialistas, de redes *feedforward* com múltiplas camadas (esse é o mote por trás das MEs engatilhadas [472]), deve o projetista recorrer a métodos de otimização baseado em informação do gradiente do erro (p. ex., *Backpropagation* ou métodos de segunda ordem). Ademais, uma variante denominada GEM (*Generalized EM*), proposta

também por Jordan & Xu [240], baseia-se na estratégia alternativa de se garantir apenas o decréscimo (não necessariamente a minimização) das funções-erro (4.31) e (4.32) a cada iteração, incorrendo em algum ganho de desempenho e evitando problemas de convergência a mínimos locais.

4.2.3 Pontos adicionais

O efeito de se adotar uma camada simples de Perceptrons com ativações *soft-max* como rede mediadora é o de dividir, via hiperplanos “suaves”, o espaço de entradas em regiões sobrepostas, com fronteiras de decisão convexas e conectadas. Ramamurti & Ghosh [381] advogam que o uso de redes *gating* dessa natureza pode vir a causar prejuízo de desempenho quando da modelagem de funções não-triviais. Segundo os autores, para as entradas que *não* estejam muito próximas a nenhum dos hiperplanos projetados pela *gating*, tipicamente vários g_k 's podem assumir valores bem maiores que zero. Isso decorre do simples fato de que qualquer um desses pontos está localizado no lado positivo de, em média, 50% dos hiperplanos projetados. Esse alto acoplamento entre os especialistas pode vir a prejudicar sobremaneira o processo de estimação dos seus parâmetros, mormentemente quando o número de especialistas, K , for elevado.

Almejando ganhos de desempenho, Xu *et al.* [496] propuseram um modelo alternativo para a rede mediadora, dando origem à variação conhecida como MEs localizadas (de *Localized Mixtures of Experts* — LMEs). Em vez de hiperplanos, tal modelo se destina a dividir o espaço de entradas via hiperelipsóides “suaves”, de modo a gerar regiões *localizadas* de especialização. Nesse sentido, a saída de ativação de cada porta da *gating* passa a assumir um novo formato:

$$g_k(\mathbf{x}) = \frac{\alpha_k p_k(\mathbf{x}|\nu_k)}{\sum_{k^*} \alpha_{k^*} p_k(\mathbf{x}|\nu_{k^*})}, \quad (4.33)$$

onde $\sum_k \alpha_k = 1, \alpha_k \geq 0$ e $p_k(\mathbf{x}|\nu_k)$ denota uma função densidade de probabilidade com formato idêntico daquela em (4.22):

$$p_k(\mathbf{x}|\nu_k) = \frac{1}{(2\pi\Sigma_k^2)^{D_x/2}} \exp\left(-\frac{\|\mathbf{x} - \mu_k\|^2}{2\Sigma_k^2}\right),$$

onde Σ_k^2 representa uma variância local para a k -ésima porta. Disso, as saídas da *gating* continuam sendo não-negativas e, quando somadas, igualam-se à unidade⁽²⁶⁾. O que (4.33) retrata é que a região de influência do k -ésimo especialista deve girar em torno do centro μ_k , sendo esse valor configurado durante o passo-M.

Esse novo modelo de *gating* leva a uma nova interpretação para LMEs, obtida substituindo-se (4.33) em (4.21) (e desconsiderando-se a influência dos parâmetros livres):

$$p(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}) = \sum_{k=1}^K \frac{\alpha_k p_k(\mathbf{x}^{(n)})}{\sum_{k^*} \alpha_{k^*} p_k(\mathbf{x}^{(n)})} \phi_k(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}).$$

⁽²⁶⁾É interessante notar que os numeradores de (4.33) podem ser interpretados, por sua conta, como componentes de um segundo modelo de mistura mais simples [316, 451], algo que pode ser usado, segundo Moerland [328], como artifício para se determinar bons valores iniciais para os parâmetros da função densidade $p_k(\cdot)$.

Todavia, almejando obter uma solução analítica exata, de passo único, para a estimação dos parâmetros livres da *gating* (passo-M), Xu *et al.* [496] passaram a trabalhar com a densidade condicional conjunta,

$$p(\mathbf{t}^{(n)}, \mathbf{x}^{(n)}) = p(\mathbf{t}^{(n)}|\mathbf{x}^{(n)})p(\mathbf{x}^{(n)}) = \sum_{k=1}^K \alpha_k p_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}),$$

a qual, por máxima verossimilhança, leva à seguinte função-energia sobre todo o conjunto de dados:

$$E = - \sum_{n=1}^N \ln \sum_{k=1}^K \alpha_k p_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}). \quad (4.34)$$

Recorrendo-se ao conceito de variáveis latentes, (4.34) passa a dar origem à seguinte função-custo completa:

$$E_c = - \sum_{n=1}^N \sum_{k=1}^K z_k^{(n)} \ln \left(\alpha_k p_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}) \right).$$

Ramamurti & Ghosh [381] relataram que conseguiram resultados muito expressivos com o modelo LME, tanto em termos de acuidade como de eficiência computacional, em várias experimentações, conseguindo superar, muitas vezes, o desempenho de arranjos hierárquicos de MEs (ver a seguir). A razão levantada pelos autores foi justamente a existência dessa solução analítica exata para o passo-M proposta por Xu *et al.*⁽²⁷⁾.

Arranjos hierárquicos de misturas de especialistas [239] foram propostos visando também ao incremento das potencialidades trazidas com o modelo original. A arquitetura típica de uma ME hierárquica assume o formato de árvore, onde várias redes mediadoras ocupam os nós não-terminais e os seus correspondentes especialistas residem aos nós-folhas. Tal modelo hierárquico difere do original na medida em que o espaço de entradas passa a ser dividido em subespaços *aninhados*, sendo a informação combinada e redistribuída entre os especialistas mediante o controle seqüencial das várias redes *gating* existentes. Ou seja, pode-se encarar o modelo ME hierárquico como uma versão *multiresolucional* do modelo original, em que o princípio de “divisão-e-conquista” passa a ser aplicado recursivamente. Assim, uma rede *gating* de mais alto nível se ocupa de combinar e redistribuir efetivamente a informação que lhe chega aos módulos sob seu controle (i. e., pertencentes à subárvore de sua competência). Como as subárvores estão aninhadas umas às outras, isso garante que haja coesão entre os ajustes dinâmicos feitos sobre os vários parâmetros livres do arranjo total. Conforme Haykin [198], tal fato, por si só, já dá razão à expectativa de que MEs hierárquicas consigam superar, em termos de acuidade e capacidade de generalização, arranjos simples de MEs, apresentando, no pior caso, desempenho equiparado. Todavia, o preço que se paga por esse incremento de desempenho é o aumento, às vezes considerável, do custo computacional [381, 469].

Como mencionado *ein passant* anteriormente, uma outra abordagem alternativa ao modelo ME original está no emprego do que Weigend e colaboradores chamaram de *especialistas engatilhados* [472]. Nessa variante, a rede *gating* é modelada como uma rede MLP com funções de ativação *soft-max* para os neurônios de saída. A vantagem em se adotar um modelo desse tipo é o de possibilitar que a rede mediadora

⁽²⁷⁾Jordan & Jacobs também propuseram uma solução de passo único para a rede *gating soft-max* [239, Alg. 2]; contudo, em vez de ser exata, tal solução é apenas de caráter *aproximado*.

possa realizar, pelo menos em tese, qualquer tipo de decomposição do espaço de entradas, haja vista a capacidade de aproximação universal de redes MLP. (Isso não é garantido, por exemplo, para o caso de redes mediadoras lineares com uma única camada, o que vem a justificar também o uso de arranjos hierárquicos.) Ao se lançar mão de redes *gating* não-lineares, passa a ser necessário empregar algum método de otimização não-linear (se possível, global) para o ajuste dos seus pesos. No caso de se apelar para a metodologia *Backpropagation*, pode ser mostrado que a correção do erro E_g retropropagada através da camada *soft-max* da *gating* pode ser escrita como [141]:

$$\frac{\partial E_g}{\partial u_k} = \sum_n \left[g_k(\mathbf{x}^{(n)}) - \pi_k(\mathbf{x}^{(n)}, \mathbf{t}^{(n)}) \right],$$

sendo $\pi_k(\cdot)$ a probabilidade *a posteriori* definida em (4.28). O que se depreende dessa formulação é que a tarefa da nova rede *gating* passa a ser a de tentar capturar o formato dessas probabilidades *a posteriori*.

Weigend *et al.* [472] passaram a modelar também os especialistas como redes MLP (ou seja, os subprocessos componentes da mistura podem ser também modelados não-linearmente), introduzindo uma variância adaptativa σ_k^2 (que não depende do sinal de entrada) em cada um deles [346]. A razão de se empregar essas variâncias locais é a de se tentar obter uma melhor calibragem na modelagem dos dados mediante o uso de uma estatística (momento) de maior ordem. Ademais, tal recurso provê uma medida do grau de confiança que cada especialista tem sobre as suas predições. O efeito dessas variâncias locais sobre o modelo ME geral é o de que, agora, pode-se tratar, de maneira mais efetiva, os diferentes níveis de ruído inerentes aos dados amostrais; algo particularmente muito útil quando se tem que arcar com a previsão de séries temporais estacionárias por partes (também conhecidas como multiestacionárias), as quais apresentam mudanças freqüentes entre diferentes regimes — séries caóticas e econômicas são um bom exemplo (ver Seção 8.3). De acordo com Weigend *et al.* [472], a introdução desses novos parâmetros vem a (i) reduzir problemas de *overfitting*; e a (ii) facilitar a subdivisão da tarefa entre os especialistas. Particularmente, no caso de previsão de séries temporais, (i) e (ii) se traduzem em: ter-se diferentes regimes aproximados com diferentes níveis de acuidade (i); e agrupar-se separadamente áreas com previsibilidades distintas (ii).

Considerando o uso de variâncias locais (σ_k^2), (4.22) pode ser reescrito como [328]:

$$\phi_k^{gs}(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}) = \frac{1}{(2\pi\sigma_k^2)^{(D_t/2)}} \exp\left(-\frac{\|\mathbf{t}^{(n)} - \mathbf{y}_k(\mathbf{x}^{(n)}, \theta_k)\|^2}{2\sigma_k^2}\right).$$

Além disso, existe uma solução analítica que pode ser usada quando do cálculo da atualização a ser dada a cada variância local no passo-M [141]:

$$\frac{\partial E_e}{\partial \sigma_k^2} = 0 \rightarrow \sigma_k^2 = \frac{\sum_n (\pi_k(\mathbf{x}^{(n)}, \mathbf{t}^{(n)}) \|\mathbf{t}^{(n)} - \mathbf{y}_k(\mathbf{x}^{(n)})\|^2)}{\sum_n \pi_k(\mathbf{x}^{(n)}, \mathbf{t}^{(n)})}.$$

Já o ajuste a ser dado a um certo peso w_{kp} do k -ésimo especialista não-linear passa a ser proporcional ao seguinte gradiente:

$$\frac{\partial E_e}{\partial w_{kp}} = - \sum_n \frac{g_k(\mathbf{x}^{(n)}) \frac{\partial \phi_k(\mathbf{t}^{(n)}|\mathbf{x}^{(n)})}{\partial w_{kp}}}{\sum_{k^*} g_{k^*}(\mathbf{x}^{(n)}) \phi_{k^*}(\mathbf{t}^{(n)}|\mathbf{x}^{(n)})} = \sum_n \frac{\pi_k(\mathbf{x}^{(n)}, \mathbf{t}^{(n)})}{\sigma_k^2} \left[(\mathbf{t}^{(n)} - \mathbf{y}_k(\mathbf{x}^{(n)})) \frac{\partial}{\partial w_{kp}} \mathbf{y}_k(\mathbf{x}^{(n)}) \right].$$

Essa regra de aprendizado ajusta os parâmetros livres do especialista de tal sorte a que a sua saída $y_k(\mathbf{x}^{(n)})$ se aproxime cada vez mais do valor-alvo $t^{(n)}$. Deve-se notar que essa correção nos pesos do k -ésimo especialista é modulada em proporção à sua importância relativa para a modelagem de $t^{(n)}$, algo refletido pelo fator π_k . Por outro lado, o fator $1/\sigma_k^2$ modula o aprendizado de acordo com o nível de ruído intrínseco ao processo estatístico (regime temporal) atribuído a esse especialista [472].

Numa outra direção, é válido mencionar rapidamente acerca de três estudos paralelos realizados dentro do tema específico de seleção de modelos [246] para misturas de especialistas. Tanto Fritsch *et al.* [163] como Waterhouse [469] focaram sobre MEs hierárquicas: enquanto os primeiros propuseram um método de crescimento (*growing*) dinâmico em que o especialista com a menor verossimilhança associada passa a ser substituído por múltiplos especialistas, o segundo autor concebeu e avaliou diferentes técnicas dirigidas tanto para o crescimento como para a poda dinâmica de instâncias inteiras de MEs. No estudo de Waterhouse, o arranjo hierárquico pode crescer uma camada por iteração mediante a substituição de um especialista por uma instância de ME composta por dois especialistas e uma *gating*. O especialista a ser substituído será aquele que venha a maximizar o aumento do logaritmo da plausibilidade por conta da substituição. Por outro lado, Ramamurti & Ghosh [381] realizaram um estudo extensivo envolvendo aspectos de crescimento e poda adaptativos de MEs localizadas. No caso, dois algoritmos foram avaliados em problemas de aproximação/previsão de mapeamentos variantes no tempo: o primeiro algoritmo estipulando somente o crescimento do número de especialistas, enquanto o segundo alternando fases de crescimento e poda dos especialistas. Tais estudos serviram como ponto de referência para a proposta da abordagem HGT-GAME, a qual encaminha uma alternativa evolucionária para essa questão de seleção de modelos em MEs (ver Seção 8.3).

Finalmente, outros pontos pertinentes de serem brevemente comentados aqui são:

1. Segundo Waterhouse [469], o modelo de misturas de especialistas pode ser considerado como uma espécie de generalização dos métodos de *ensembles* de estimadores (Seção 4.1); enquanto a *gating* realiza um processo de escolha ponderada e dinâmica em função do sinal de entrada (em similaridade, portanto, com as técnicas de reamostragem dinâmica dos dados), os especialistas realizam a modelagem dos dados propriamente dita (papel dos componentes da *ensemble*). Dentro desse contexto, Jacobs [223] passou a estudar e a quantificar um certo número de propriedades da arquitetura ME, tendo como palco o dilema Polarização-Variância. Como resultado, o autor constatou que (1) a variância em MEs pode ser expressa como a soma de dois termos, um relacionado às variâncias dos próprios especialistas e o outro relacionado às covariâncias entre os seus padrões de erro; e (2) que arquiteturas de MEs propendem a produzir especialistas polarizados cujas estimativas são negativamente correlacionadas.
2. Gilde [177] realizou um estudo prospectivo sobre uma versão recorrente da arquitetura ME engatilhada, em que RNAs recorrentes são empregadas tanto para a definição da *gating* como para a definição das redes especialistas. Essa versão recorrente foi avaliada perante MEs engatilhadas com redes

feedforward tendo como requisito a previsão de séries temporais geradas por um sistema dinâmico simulado. O autor constatou que tanto uma versão quanto a outra de MEs engatilhadas apresentaram desempenhos similares, sendo que, em média, a arquitetura recorrente pôde encontrar representações mais sucintas (i. e., com menos parâmetros livres no total) para as séries temporais estudadas.

3. Fancourt & Príncipe [141] apresentaram uma solução para a questão de como se embutir, em arquiteturas MEs engatilhadas, informação *a priori* acerca das taxas de comutação entre os diferentes regimes de uma série temporal qualquer. A solução baseia-se na adição de um termo de memória às densidades de probabilidade dos especialistas, modelando a raiz quadrada do erro quadrático médio (ver Tab. 2.2) local de cada especialista como uma distribuição qui-quadrado. Novas equações de retropropagação de erro relativas ao passo-M (tanto para a *gating* como para os especialistas) foram derivadas pelos autores, os quais comprovaram, via simulação, que a versão com memória apresentou melhorias quanto ao processo de segmentação da série frente à arquitetura normal de MEs engatilhadas.
4. Ademais, Srivastava *et al.* [441] introduziram formalmente uma extensão multiresolucional de MEs engatilhadas, denominada *Especialistas Engatilhados Sensíveis a Escala* (de “*Scale-Sensitive Gated Experts*” — SSGE). Enquanto a arquitetura ME engatilhada normal é capaz de prover apenas um único particionamento do espaço de entradas, o modelo SSGE oferece ao projetista a flexibilidade de descobrir e operar com *famílias* de possíveis segmentações. Para tanto, a granulosidade das partições geradas sobre a superfície de regressão em análise passa a ser controlado por um parâmetro externo, sendo que cada valor desse parâmetro dá origem a um membro da família de segmentações. Avaliando o desempenho dessa nova extensão ante a um problema de segmentação de uma série temporal não-trivial, os autores comprovaram a eficácia da nova abordagem mediante a descoberta de alguns regimes de comportamento do sinal não detectados pela arquitetura ME engatilhada convencional. Com isso, a interpretação do processo não-linear por trás da série analisada ficou mais facilitada.

4.3 Sinopse do capítulo

Duas classes de abordagens complementares de Comitês de Máquinas foram analisadas neste capítulo: *ensembles* de estimadores e sistemas modulares. Enquanto a primeira classe apregoa a fusão efetiva de um certo número de preditores redundantes (i. e., que realizam a mesma tarefa sob perspectivas distintas), a segunda classe de abordagens fundamenta-se diretamente no preceito de “divisão-e-conquista” para fins de decomposição de tarefas não-triviais em subproblemas de menor complexidade. Dentre as várias arquiteturas de sistemas modulares existentes na literatura, ênfase especial foi dada ao modelo de misturas de especialistas. Decerto, uma motivação biológica, uma interpretação probabilística, aspectos de treinamento via algoritmo EM, bem como algumas considerações sobre abordagens variantes de MEs foram anunciados com mais detalhes. Para *ensembles*, além de resultados teóricos justificando o estudo do tema, foram discutidos alguns aspectos relacionados à noção de diversidade e aspectos concernentes às operações de criação, seleção e combinação dos estimadores-componentes.

Capítulo 5

Coordenação em sistemas com múltiplos agentes

“A união faz a força.”

Adágio popular

“A cooperação é a convicção plena de que ninguém pode chegar à meta se não chegarem todos.”

Virginia Burden

A noção por trás do conceito de *agente de software* [172, 214] vem se tornando cada vez mais popular à medida que um sem-número de aplicações, projetadas sob a ótica desse novo paradigma computacional, vem ratificando as suas enormes potencialidades. O termo “agente”, na sua conotação mais difundida, se refere a todo ente com capacidade e permissão de atuar e responder em nome de outrem.

Segundo essa acepção mais “canônica”, qualquer entidade (real ou artificial) para quem se *delega* alguma tarefa ou atividade pode ser considerada um agente. Contemplados aqui estão os agentes humanos, como os de viagem, os de negócios e até os funerários, e os *agentes virtuais*, que residem em mundos artificiais (como a Internet). Dessa descrição, advêm duas propriedades inerentes ao conceito de “ser agente” (ou *agência*), quais sejam, a da *autonomia*, característica de quem é autodeterminado e atua independentemente (i. e., sem supervisão ou intervenção de outra parte) e de acordo com seus próprios *objetivos* (possuindo, portanto, uma *agenda* própria), e a da *especialização*, refletida na capacidade de atuar em domínios bem específicos.

Considerando, particularmente, os agentes computacionais, quer sejam os implementados em *software* ou *hardware*, não existe ainda um consenso definitivo quanto ao conjunto de propriedades básicas que estes devam exibir [158]. Por exemplo, consoante a concepção “clássica” dada por Russel e Norvig [402], um agente pode ser definido como qualquer artefato com habilidades de perceber o ambiente onde reside, mediante o uso de sensores, e de atuar sobre esse ambiente, via seus atuadores. A ênfase maior aqui está no fato de

que um agente tem à sua disposição um repertório de ações e percepções por intermédio das quais ele pode atingir seus objetivos. Além disso, o agente *deve* estar *situado* em um ambiente (mundo) bem definido, tendo suas atividades possivelmente afetadas pelos eventos que nele acontecem.

Nesse sentido, é pertinente caracterizar um dado ambiente de acordo com as propriedades estruturais que ele apresenta frente aos agentes que o integram. Tal caracterização se torna possível tendo em mente cinco dimensões distintas [402]: (i) *acessível* ou *inacessível*, onde, no primeiro caso, o agente pode obter informação completa, exata e atualizada acerca do estado do ambiente; (ii) *determinístico* ou *não-determinístico*, onde, para o primeiro caso, não há incerteza acerca do resultado realizado por uma dada atividade, já que a cada ação só se tem um único efeito garantido; (iii) *episódico* ou *não-episódico*, onde, no primeiro tipo de ambiente, o desempenho exibido pelo agente, em um dado cenário temporal (episódio), independe das atividades realizadas por ele no passado (em outros episódios); (iv) *estático* ou *dinâmico*, onde, no segundo caso, o estado do ambiente, entre duas ações sequenciais realizadas por um mesmo agente, pode mudar, como fruto das ações realizadas por outros agentes; e (v) *discreto* ou *contínuo*, onde, para o primeiro caso, existe um número limitado e finito de ações e percepções à disposição do agente. Tomando como exemplo o domínio de futebol de robôs (real ou virtual), este pode ser qualificado como sendo inacessível, não-determinístico, episódico, dinâmico e discreto.

A despeito de ser bastante ampla, abrangendo desde insetos até robôs, passando por células e termostatos, a conceituação supracitada não delimita todos os possíveis atributos intrínsecos ao conceito de agência. Além da autonomia, comportamento especializado e vinculação a um certo ambiente, outras propriedades têm sido constantemente enaltecidas na literatura [158, 172, 214], sendo algumas delas descritas na Tab. 5.1⁽¹⁾. Das propriedades listadas, as duas últimas se tornam bastante relevantes quando se pretende estipular o que seria um *agente inteligente* [495]. Com relação à capacidade de adaptação contínua (aprendizado), os agentes inteligentes podem ser classificados como *deliberativos* (ou cognitivos), *reativos* (ou reflexivos) ou *híbridos* [355]. Os integrantes da primeira classe possuem um módulo de raciocínio simbólico na forma de uma base de conhecimento, na qual fica representado o “modelo do mundo” onde atuam. Tal modelo é geralmente preconcebido, mas seu estado é alterado continuamente pelo agente em resposta às novas informações sobre o ambiente capturadas por seus sensores. E é com base na interpretação desse modelo que o agente toma decisões (de forma racional) e *planeja* quais ações serão necessárias para atingir o seu objetivo final [370, 407].

Já os agentes categorizados como reativos [64, 146] não necessitam de modelagem do mundo para determinar suas ações. O seu comportamento é guiado por processos do tipo “estímulo-resposta”, onde há um mapeamento direto entre situações (condições) e respostas (consequências) associadas. Tal mapeamento pode ser implementado por meio de funções matemáticas, via regras “IF-THEN” (booleanas ou *fuzzy*), ou mesmo por uma rede neural *feedforward*. A *granulosidade* desses agentes tende a ser menor, haja vista seu domínio de atuação ser mais restrito.

⁽¹⁾Complementam essa lista outros atributos secundários como antropomorfismo, credibilidade, personalidade e capacidade de exibir emoções.

Tab. 5.1: Algumas propriedades de agentes de *software*.

Propriedade	Descrição
<i>Benevolência</i>	agentes <i>altruístas</i> se dispõem a ajudar outros parceiros, caso isso não gere conflitos com seus próprios objetivos, ao passo que agentes <i>egoístas</i> não apresentam tal qualidade.
<i>Pró-atividade</i>	um agente deve ser orientado para metas e deve demonstrar iniciativa e capacidade de <i>antecipação</i> aos fatos.
<i>Continuidade temporal</i>	um agente deve possuir um <i>ciclo de vida</i> bem definido e ininterrupto, mantendo-o continuamente ativo.
<i>Mobilidade</i>	um agente deve ser capaz de se deslocar de um ambiente a outro.
<i>Racionalidade limitada</i>	um agente deve atuar de forma racional [407], tomando decisões de acordo com seus objetivos e <i>crenças</i> , mesmo que a otimalidade dessas decisões seja limitada por restrições temporais, espaciais e/ou de recursos.
<i>Adaptabilidade</i>	o agente deverá adaptar-se freqüentemente às condições reinantes no ambiente onde reside, exibindo capacidade de alterar o seu próprio comportamento com base na sua experiência.
<i>Sociabilidade</i>	os agentes são capazes de interagir com outros agentes (humanos ou não), exibindo comportamento social, geralmente mediante o intercâmbio de mensagens concordantes com uma linguagem de comunicação de mais alto nível.

Por outro lado, um agente híbrido apresenta características comuns a ambos os tipos acima definidos, sendo passível de ser empregado, por exemplo, quando se pretende solucionar a incapacidade de ação adequada por parte de um agente puramente deliberativo, no momento em que este deve tomar uma decisão rápida e espontânea no enfrentamento de situações imprevistas. De um outro modo, pode-se proporcionar a um agente puramente reativo a capacidade de raciocínio e planejamento, para quando este se deparar com situações nas quais o estado do ambiente diverge bastante dos seus objetivos iniciais.

Em se tratando da propriedade da sociabilidade, esta leva em consideração o fato de que, comumente, um agente inteligente não atua isoladamente em seu ambiente, devendo, muitas vezes, *interagir* com outros parceiros para a execução de tarefas complexas, para a tomada de decisão conjunta ou para fins de compartilhamento de recursos. Como um ser que interage, um agente pode ser afetado em suas atividades por outros agentes e, talvez, por humanos. Mediante o uso de *protocolos de comunicação* já estabelecidos, os agentes podem intercambiar conhecimento e informação de alto nível, apreendidos ao longo da sua execução, de tal forma a operarem mais produtivamente na realização de seus objetivos. De outro modo, por intermédio de *protocolos de interação* (negociação e cooperação), podem resolver conflitos, oriundos dos efeitos da interferência social [169, 398], ou compartilhar, alocar e delegar tarefas entre si [110].

A possibilidade de se explorar, efetivamente, o trabalho coletivo entre diferentes entidades autônomas traz consigo uma série de benefícios em termos computacionais, servindo de motivação e justificativa para os trabalhos de pesquisa na área da Inteligência Artificial Distribuída (IAD) [132, 350]. Dentre tais justificativas, é comum serem citadas [214, 355]: (1) computações distribuídas são mais fáceis de serem projetadas sempre que os requisitos do problema-alvo forem de natureza inerentemente complexa e distribuída; (2) há

vezes em que uma metodologia de caráter centralizado se torna impossível de ser implementada, pelo fato de tanto as fontes de informação como os pontos de controle/monitoração já se encontrarem naturalmente (geograficamente) distribuídos; (3) cada vez mais, aumenta-se o número de processadores interconectados em redes de computadores, havendo a necessidade crescente de *interoperá-los* para fins de resolução descentralizada de problemas complexos (p. ex., quebra de chaves criptográficas, análise de imagens de satélites, seqüenciamento de cadeias gênicas, etc.); e (4) em atividades que envolvam risco (como controle de tráfego aéreo e aplicações médicas) ou de forte apelo econômico (como *brokering* de serviços financeiros), há a necessidade de se trabalhar com plataformas de *software* paralelas altamente confiáveis, exigindo medidas de segurança e tolerância a falhas cada vez mais robustas e adaptativas.

De forma sucinta, a razão principal para se interconectar agentes computacionais é o de habilitá-los a *cooperarem* na resolução de problemas (haja vista que cada agente tem informação incompleta do mundo e é limitado em suas habilidades), a compartilharem *know-how* (oriundo dos especialistas que os projetaram), a trabalharem em paralelo em problemas comuns, a representarem múltiplos pontos-de-vista sob determinado assunto e a de serem reusáveis em novas situações antes não previstas. Contudo, há um preço alto a ser pago por tanta *flexibilidade*: como garantir que os agentes trabalhem e interajam *coerentemente* em grupo? Isso só é possível mediante o emprego de uma *política de coordenação* apropriada ao cenário de aplicação em questão, definindo como os agentes devem adaptar as suas próprias atividades (agendas) em face das interações que precisam realizar ao longo do seu ciclo de vida. O objetivo de longo prazo da IAD é, portanto, o de entender melhor como se dão os processos de interação (coordenação, principalmente) entre entidades inteligentes, de modo a se desenvolver métodos e mecanismos que possam implementá-los em ambientes computacionais. Esse objetivo desperta uma série de questões teóricas e práticas, todas elas centradas em torno de um mote básico, traduzido na frase “*quando e como interagir com quem*” [475].

O restante deste capítulo se ocupa da análise do problema de coordenação em sistemas computacionais contendo múltiplos módulos autônomos, apresentando definições-chave para o tema bem como algumas abordagens voltadas à sua implementação. O intuito é prover um ferramental teórico para o trabalho apresentado na Seção 8.4.

5.1 Inteligência artificial distribuída — Aspectos importantes

Historicamente, e também por motivações científicas distintas, o domínio de atuação da IAD passou a ser dividido em duas linhas de investigação diferentes (mas complementares), denominadas *Resolução Distribuída de Problemas* (RDP) e *Sistemas Multiagentes* (SMAs) [133].

Resumidamente, pode-se afirmar que, na metodologia RDP [118, 435], os agentes ou nodos de computação — que se encontram, geralmente, alocados fisicamente a máquinas distintas — cooperam uns com os outros, dividindo e compartilhando conhecimento sobre o problema-alvo, o qual já deve ser previamente conhecido e precisamente delineado, e sobre o processo de obtenção de uma possível solução. Nessa

linha de pesquisa, mais centrada nos requisitos do problema do que nas características a serem exibidas pelos módulos [475], os agentes costumam *não* existir *a priori*, sendo projetados especificamente para um determinado fim. Desse modo, o reuso desses agentes, bem como do seu modelo organizacional (também preconcebido e, geralmente, não-adaptativo), não é normalmente possível, haja vista terem sido customizados pelo projetista (especialista no domínio do problema) [169]. Além do mais, os agentes devem ser benevolentes entre si, ou seja, devem ajudar uns aos outros sempre que necessário, e devem saber “dialogar”, isto é, exibir representações dos fatos e linguagens de comunicação compatíveis. Inteligência, nesse contexto, pode ser concebida como a capacidade do grupo de resolver eficientemente uma certa classe de problemas [355].

Já as abordagens SMA estão mais voltadas para as atividades e interações realizadas pelos agentes autônomos em um mundo multiagente [215]. Nessa linha, os módulos, cuja existência pode preceder ao surgimento de um problema específico, podem trabalhar conjuntamente em direção a um único objetivo global ou rumo a objetivos individuais separados mas que, de alguma forma, se sobrepõem; em ambas as situações, entretanto, a autonomia de um agente não fica condicionada a de outros, como em sistemas RDP. Mais ainda, em sistemas multiagentes, é comum lançar-se mão do conceito de *comunidade*, cujo enfoque principal se dá na existência de uma sociedade composta por vários agentes auto-reguladores. Do comportamento social integrado exibido pelo coletivo é que se faz *emergir* a inteligência do sistema.

Do ponto de vista de uma sociedade multiagente, a concepção dos módulos, dos seus protocolos de interação e dos modelos organizacionais que os integram e coordenam se dá de maneira independente de um problema particular a ser solucionado, tornando possível, pois, reutilizar tais componentes em aplicações similares. Em verdade, os próprios agentes é que irão instanciar dinamicamente os seus protocolos de interação ou sua arquitetura organizacional, quando tiverem que lidar com novos problemas. Em certas situações, conflitos de interesse poderão surgir, exigindo ciclos e protocolos de negociação para intermediar disputas por determinados recursos (quando os agentes são não-benevolentes entre si) ou para se chegar a um consenso coletivo (quando os agentes cooperam).

Em sistemas RDP, a interação entre os nodos geralmente ocorre via intercâmbio de primitivas de baixo nível predefinidas ou por compartilhamento de dados comuns. Tendo em mente requisitos de eficiência e confiabilidade, tais procedimentos de comunicação são definidos *a priori* e estão intimamente ligados ao modelo algorítmico subjacente, algo que, muitas vezes, acaba limitando a autonomia dos agentes. A arquitetura de *quadro-negro* (*blackboard*), baseada na idéia de *brainstorming* de especialistas, onde hipóteses e soluções parciais são progressivamente escritas ou modificadas em uma base de dados global, é comumente adotada na construção desses sistemas [215].

Por outro lado, em abordagens que seguem a filosofia SMA, a concepção do repertório de interações é realizada independentemente de um cenário específico, buscando-se desenvolver protocolos de interação [215] mais genéricos, que possam ser reutilizados freqüentemente. Tais protocolos de interação são construídos sobre protocolos de comunicação de mais alto nível, quando comparados àqueles de RDP, sendo geralmente especificados sob o formato de uma *linguagem de comunicação de agentes*. KQML (de *Knowledge Query and Manipulation Language*), por exemplo, é uma linguagem padronizada que separa a semântica do protocolo

de comunicação (que é independente do domínio) da semântica da mensagem embutida (que pode depender do domínio), exibindo algumas características interessantes [149]: (i) é uma linguagem estruturada em três níveis: conteúdo, mensagem e comunicação; (ii) mensagens KQML podem ser aninhadas, ou seja, o conteúdo de uma mensagem KQML pode ser outra mensagem KQML; (iii) seu repertório de mensagens-padrão é baseado em *performatives*, isto é, declarações com poder de ação verbal (tais como, requerer, insistir, perguntar, convencer, reportar, demandar, etc.), segundo a Teoria do Ato de Fala (*Speech-Act*) [215]; (iv) agentes que conversam em KQML atuam como “clientes” e “servidores” uns dos outros, sendo que suas trocas de comunicação podem se dar de modo síncrono ou assíncrono; e (v) KQML pode coexistir com diferentes padrões de suporte à construção de *ontologias*, conjuntos de termos e definições trazendo significado a particularidades do domínio de aplicação em questão.

Em um mundo contendo pelo menos dois agentes *A* e *B*, a simples coexistência destes pode ocasionar em empecilhos/benefícios ao atendimento de seus objetivos, mesmo que ambos os módulos não estejam conscientes dos anseios e planos do outro. A esse fenômeno, dá-se o nome de *interferência social* [169], podendo se manifestar de forma positiva (p. ex., quando uma ação de *A* aproxime *B* de seus objetivos), negativa (p. ex., quando uma ação de *A* afaste *B* de seus objetivos) ou neutra. Ao se levar em conta o efeito simultâneo de interferência mútua entre os agentes, tem-se o fenômeno da *interação social*. Garcia e Sichman [169] apresentam diferentes padrões de interação social entre agentes, seguindo uma nomenclatura baseada nas relações simbióticas entre espécies naturais, tais como aquelas relatadas na Seção 6.7.

Ainda sob a ótica da interação em um ambiente multiagente, agentes podem exibir redes de interdependências, na medida em que passam a delegar (parte de) suas tarefas a outros membros da sociedade. Para tanto, muitas vezes, um agente *A* deverá exercer algum tipo de *influência* sobre outro agente *B* a fim de que este possa adotar o novo objetivo de realizar uma ação *z* para que *A* atinja seu objetivo. Uma vez que *B* adota a nova ação *z* em favor de *A*, ele firma um *compromisso* (*commitment*) junto a *A* em relação a *z*. Tal obrigação continua válida durante todo o período de execução da ação, a menos que se alterem as condições nas quais o acordo se baseou. Para delimitar sob que circunstâncias os *commitments* devam ser reavaliados é que existem as *convenções* (*conventions*) [215]. De acordo com essas regras de caráter emergencial, o agente *B* pode decidir manter, retificar ou mesmo abandonar seu compromisso junto a *A*.

No que segue, passa-se a tratar especificamente do problema de coordenação de agentes, levando-se em consideração as noções discutidas nesta seção.

5.2 Coordenação em sistemas multiagentes

Primeiramente, é relevante ressaltar que abordagens RDP e SMA normalmente costumam divergir entre si com relação ao modelo de coordenação a ser empregado para a orquestração dos módulos cooperantes. De acordo com a filosofia por trás da primeira linha, é comum existir uma noção forte de *controle global*, por meio do qual fica garantido o comportamento coerente do grupo de agentes como um todo, sendo esse

controle tipicamente centralizado em um ou poucos nós. Uma vez que não há canais sempre disponíveis para comunicação intensa entre os nós, projetistas de sistemas RDP passam a estudar como as atividades locais de resolução de problemas podem ser coordenadas a fim de atenderem ao objetivo global. Coordenação, nesse caso, está intimamente ligada a protocolos de cooperação, levando a questões de decomposição e alocação de tarefas (“dividir para conquistar”) e de integração progressiva das soluções parciais geradas [133].

Como em sistemas multiagentes não há a necessidade de se conceber os módulos em função de um problema previamente estipulado, ou mesmo a restrição de que esses módulos devam cooperar entre si, o problema de coordenação passa a assumir uma perspectiva diferente da do caso anterior: o comportamento coerente do grupo deve ser fruto das ações e planejamentos realizados pelos próprios agentes, não existindo, portanto, um controle global determinando/planejando os passos a serem seguidos por toda a comunidade a todo momento. Cabe aos agentes decidirem entre si como será feita a divisão e a alocação de tarefas, de acordo com as potencialidades (especializações) de cada um. Nesse sentido, é comum dizer que a coerência do grupo deve emergir a partir das suas próprias interações [154, 236].

Na seqüência, passa-se a caracterizar mais minuciosamente o problema de coordenação em IAD. Descreve-se, primeiramente, uma lista de definições capturando aspectos relativos ao tema. Posteriormente, são introduzidos os conceitos de políticas, protocolos e mecanismos de coordenação [355]. Por fim, são discutidas, brevemente, algumas abordagens proeminentes já existentes na literatura, que tratam do problema de coordenação multiagente sob variados pontos de vista.

5.2.1 Caracterização do problema de coordenação

Definir com precisão o que seja o processo de coordenação não é uma tarefa das mais simples. Devido a interesses epistemológicos distintos, não se torna surpresa a existência de posições divergentes com relação a quando e como o termo “coordenação” deve ser empregado. Seguem abaixo algumas definições comumente citadas na literatura e sumarizadas por Ossowski [355]:

- “Coordenação é um tipo de cola dinâmica que une as tarefas em partes maiores e mais significativas” (Holt 1988);
- “Coordenação indica como se adaptar ao ambiente” (von Martial 1992);
- “A integração e o ajuste harmonioso de esforços individuais em direção à realização de um objetivo maior” (B. Singh 1992);
- “O ato de gerenciar interdependências entre atividades” (Malone e Crowston 1994).

Barbuceanu e Fox [38] provêem um refinamento da definição dada por Malone e Crowston, tratando especificamente de sistemas com múltiplos agentes. Segundo os autores, coordenação é o processo de se gerenciar dependências entre atividades de agentes autônomos, em condições de conhecimento incompleto

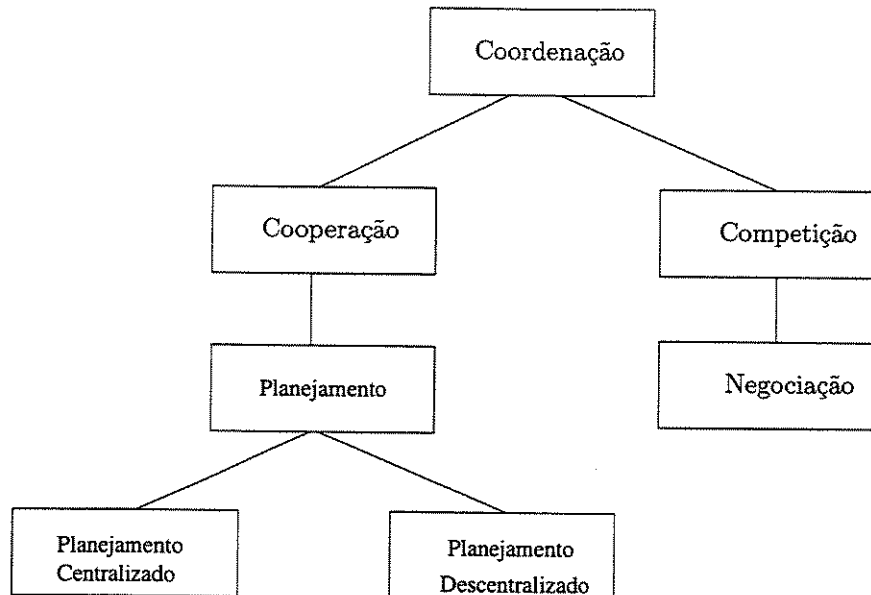


Fig. 5.1: Uma taxonomia contendo alguns dos diferentes modos pelos quais os agentes podem coordenar seus comportamentos e atividades. (Fonte: Huhns e Stephens [215].)

acerca das mudanças dinâmicas do ambiente, e de acordo com as ações, reações e objetivos desses agentes, de tal modo a que seja atingido um nível apropriado de *coerência* no comportamento do sistema como um todo. Nessa conceituação, fica destacado o termo “coerência”, que significa quão bem o sistema de agentes vem se comportando enquanto percebido como unidade⁽²⁾. Já o *grau de coordenação* entre os módulos pode fornecer uma pista sobre como os agentes vêm evitando atividades supérfluas, obsoletas ou redundantes, de como vêm evitando conflitos entre suas dependências⁽³⁾, e de como estão lidando com situações adversas/imprevisíveis.

Consoante Huhns e Stephens, “cooperação é coordenação entre agentes não-antagônicos, enquanto negociação é coordenação entre agentes egoístas ou competidores” [215]. Desse modo, tanto cooperação como competição podem ser vistas como formas de interação (positiva e negativa) entre entidades, só que, no primeiro caso, os agentes falham ou têm êxito conjuntamente, haja vista a existência de um objetivo grupal mais relevante que o(s) individual(is) — o conceito de *time* [206, 448] representa bem essa situação. A Fig. 5.1 traz uma possível árvore taxonômica, sugerida por Huhns e Stephens, relacionando a noção de coordenação a outros conceitos de IAD.

⁽²⁾No caso de sistemas conexionistas (Cap. 2), essa coerência é mais fácil de ser percebida, pois ela é devida ao forte acoplamento existente entre os módulos (neurônios), cuja autonomia, quando comparada à de agentes de *software*, também é mais limitada. A abordagem MASCoNN (Seção 8.5) propicia uma nova perspectiva nesse sentido, uma vez que o processo de treinamento construtivo da rede passa a ser coordenado por neurônios mais granulados, exibindo perfil de agentes.

⁽³⁾Por exemplo, agentes esperando por eventos que nunca ocorrerão (*livelock*) ou agentes esperando por eventos interdependentes em sua causalidade (*deadlock*).

5.2.2 Políticas, protocolos e mecanismos de coordenação

Mediante uma *política de coordenação* apropriada é que ficarão especificados os objetivos do grupo, como esses objetivos serão mapeados em tarefas, a ordem de execução das ações (*scheduling*), qual agente irá realizar qual (sub) tarefa (decomposição e alocação), como os agentes irão trocar informações sobre os seus resultados parciais, como terão acesso a recursos escassos, etc. O processo de coordenação se torna ainda mais vital (e difícil) em sociedades multiagentes, onde os módulos autônomos, e geralmente heterogêneos (i. e., com estruturas internas distintas), têm uma visão limitada do que acontece no ambiente e das consequências que suas ações trarão para o estado global do sistema (*bounded rationality* [407]). Nessas circunstâncias, as ações realizadas por um agente restringirão e serão restringidas pelas ações de outros agentes. A fim de alcançarem seus objetivos, tais agentes deverão gerenciar *dinamicamente* tais restrições em concordância com um *protocolo de coordenação*, que irá definir, por exemplo, como os agentes irão eventualmente alterar a prioridade de suas ações em função das ações realizadas ou em realização por outros agentes (exigindo, muitas vezes, renegociação constante entre as partes). A instanciação de um protocolo de coordenação, levando-se em consideração a política de coordenação e as nuances da aplicação-alvo, se dá mediante o emprego de um ou mais “dispositivos” especiais conhecidos como *mecanismos de coordenação*.

É interessante frisar de antemão que nem sempre é possível (ou mesmo exigido) se estabelecer uma fronteira clara entre esses conceitos. Como exemplo, em alguns dos trabalhos investigando o uso de técnicas evolutivas (ou similares) para o propósito de coordenação de agentes autônomos [8, 179, 193, 200, 300, 308] (ver Seção 6.9), inclusa aqui a abordagem apresentada na Seção 8.4, tanto aspectos relativos ao que seria o protocolo como ao que seria(m) o(s) mecanismo(s) de coordenação costumam ser projetados simultaneamente, sob o mesmo processo de busca automática, tendo em mente atender-se a requisitos da aplicação-alvo e/ou da política de coordenação vigente — estes capturados quando da definição da função de *fitness*. Ademais, como se verá a seguir, a maioria das propostas é destinada a tratar pontos isolados do problema de coordenação multiagente, desconsiderando as interrelações entre as partes (políticas, protocolos e mecanismos); exceção feita ao *framework* conceitual integrado (TAEMs) proposto recentemente por Decker e Lesser [117].

Particularmente, em relação à formalização do processo de coordenação temporal (*scheduling*) entre agentes, uma abordagem baseada em lógica foi proposta recentemente por Singh [430]. Segundo esse *framework*, cada agente é representado por um pequeno arcabouço (*skeleton*) composto somente pelos eventos (ou transições) por ele realizados e que são de interesse para a atividade de coordenação. Assim, os requisitos de coordenação passam a ser especificados na forma de *relações n-árias* — diretivas de lógica temporal — entre tais eventos, tais como “*e disables f* : $\bar{e} \vee \bar{f} \vee f \cdot e$ ” (se *e* ocorrer, então *f* deve ter ocorrido antes) ou “*e and f jointly require g* : $\bar{e} \vee \bar{f} \vee g$ ” (se *e* e *f* ocorrerem, então *g* deve também ocorrer, não interessando a ordem em que isso aconteça), onde *e, f, g* denotam possíveis eventos vinculados a ações de agentes sob coordenação.

Ainda com relação ao processo de especificação formal de interdependências temporais entre tarefas, Raposo [382] apresentou um conjunto de 13 primitivas, baseadas nas relações intervalares definidas por

Allen [12], que foram designadas para a *sincronização* de atividades em sistemas colaborativos⁽⁴⁾. Mais ainda, dentro desse mesmo contexto, o autor caracterizou três tipos de dependências básicas voltadas à gerência de recursos, para serem usadas em complementação às primitivas temporais. Tais dependências básicas capturam (i) o nível de *compartilhamento* ou divisão de recursos; (ii) a *simultaneidade* no uso de recursos; e (iii) o grau de *volatilidade* de recursos. É possível criar novas dependências derivadas mediante a combinação de tais dependências básicas, de forma a se especificar relações mais complexas [382].

Existem também metodologias baseadas em Teoria de Jogos que servem para fins de modelagem e simulação de interações sociais entre diversos agentes racionais [407]. Dentro dessa perspectiva, uma solução para um problema de coordenação constitui um *ponto de equilíbrio* da superfície formada pelas *estratégias* de todos os agentes, considerados *jogadores*. Jogos de soma não-constante (também conhecidos como jogos parcialmente conflitantes) representam situações onde as posições dos jogadores não se mostram totalmente antagônicas entre si, sendo tais cenários particularmente interessantes para o estudo da coordenação — principalmente, em se tratando de times, onde os agentes compartilham uma função-utilidade comum mas não têm acesso a toda informação relevante acerca do mundo [206]. Cenários como esses, onde há potencial para cooperação, podem ser analisados sob dois prismas. Uma opção é assumir que (alguns) jogadores desconfiem de outros e passem, daí, a tirar o máximo a favor de si da atual conjuntura. Tal situação pode ser modelada via *jogos não-cooperativos de soma não-constante*, onde uma solução pode ser obtida mediante o conceito de *equilíbrio de Nash* [407]. A outra opção é assumir que os agentes estão plenamente dispostos a encontrarem um acordo para sua coordenação (i. e., *jogos cooperativos de soma não-constante*), passando a solução a ser capturada, geralmente, via o conceito de *eficiência de Pareto* [407].

Refletindo especificamente sobre o processo de partição das tarefas, este pode vir a ser efetuado previamente pelo projetista ou pode ser realizado, dinamicamente, pelos próprios agentes, via, por exemplo, estratégias de planejamento hierárquico [370]. A decomposição das tarefas pode ser feita *espacialmente*, baseando-se na disposição física das fontes de informação e/ou pontos de decisão, ou *funcionalmente*, de acordo com as habilidades dos agentes disponíveis. Uma vez decompostas, as tarefas podem ser alocadas em consonância com alguns critérios de desempenho, tais como [215]: (a) evitar a sobrecarga de recursos críticos; (b) casar os requisitos das (sub)tarefas com as capacidades dos agentes; (c) atribuir responsabilidades sobrepostas entre os agentes, de modo a atingir maior coerência; (d) designar tarefas altamente interdependentes a agentes (espacialmente ou semanticamente) vizinhos; e (e) se necessário, reatribuir, dinamicamente, tarefas urgentes.

O mecanismo de coordenação em atividade em um dado sistema de agentes determina tanto a sua dinâmica interna, isto é, as interações entre os agentes, como as propriedades exibidas externamente pela

⁽⁴⁾ Colaboração, que significa “trabalhando conjuntamente”, refere-se a formas de interação de alto nível em que os interesses dos participantes interagentes *podem* (mas não necessariamente *devem*) se sobrepor — uma espécie de “ajuda mútua”. Exemplo de atividade colaborativa é a de cunho educacional, onde estudantes, cada um com seu próprio objetivo, podem interagir entre si com o intuito de melhorarem seu desempenho acadêmico (p. ex., via troca de informações ou notas de aula), embora tal situação *não* seja requisito para o êxito de cada um. Mesmo sendo os dois termos, “colaboração” e “coordenação” considerados, muitas vezes, equivalentes, cooperação, no contexto taxonômico da IAD, é a palavra mais usualmente adotada.

sociedade. O fato é que tais mecanismos variam normalmente com o domínio da aplicação e dependem tanto dos agentes (e sua inclinação para cooperarem), como do ambiente (e seu grau de estabilidade e previsibilidade), como das tarefas (e seu grau de decomposição). A seguir são apresentadas algumas das mais importantes classes de mecanismos de coordenação usadas em sistemas IAD⁽⁵⁾.

5.2.2.1 Planejamento

Mediante esse mecanismo, é determinado um conjunto de planos (*scripts*) especificando todas as possíveis (inter)ações futuras do grupo. Planos podem ser revisados (i. e., replanejamento) durante sua execução atendendo a certas contingências. Distinguem-se duas estratégias [355]: *centralizada*, onde uma única entidade fica responsável pela criação ou reparo dos planos; e *descentralizada*, onde o conjunto de planos é desenvolvido por vários agentes, nenhum dos quais tem visão completa das atividades sendo realizadas em toda a comunidade [370].

5.2.2.2 Reatividade

Como mencionado, os agentes reativos são capazes de perceber estímulos à sua volta e reagir, prontamente, a esses estímulos. A propriedade da reatividade pode ser encarada também como um possível mecanismo de coordenação, segundo o qual as informações concernentes aos comportamentos dos agentes e que são necessárias para o processo de coordenação encontram-se disponíveis no próprio ambiente, não havendo necessidade de se utilizar comunicação direta/explicita entre as partes [146]. Destarte, a coordenação reativa consiste tanto na reação do agente a modificações que ocorrem em seu ambiente local como na adaptação de suas ações frente àquelas de outros agentes [164]. Desse modo, dada uma rede de interrelações entre agentes e um evento e qualquer, uma simples reação a e poderá desencadear o surgimento de outros eventos e, por conseguinte, de outras reações, não havendo, pois, necessidade de trocas de mensagens de mais alto nível.

5.2.2.3 Negociação

Mediante esse mecanismo, dois ou mais agentes, geralmente com objetivos díspares, comunicam, primeiramente, suas *posições*, as quais podem conflitar, para depois tentarem um acordo comum, implicando, muitas vezes, *concessão* das partes. Dentre as principais características apresentadas por um mecanismo de negociação em particular, estão a linguagem usada pelos agentes participantes e o processo de decisão que cada agente adota para determinar suas posições, concessões e critérios para estabelecimento dos contratos [215, 398].

⁽⁵⁾O processo de sincronização de tarefas, mencionado anteriormente, também podem ser visto como um tipo de mecanismo de coordenação [146].

5.2.2.4 Mecanismos de mercado

Mecanismos de mercado [478] (i. e., *economias computacionais*) constituem uma abordagem efetiva para a coordenação das atividades de vários agentes, exigindo, para isso, *pouca* comunicação direta entre as partes. Nesse contexto, tudo que é de interesse para os agentes passa a ser descritível por intermédio da abstração de *preços*, cujos valores servem de medida de referência para comparar as suas preferências em relação aos recursos disponíveis. Existem dois tipos de agentes, os *consumidores*, que trocam bens entre si, e os *fornecedores*, os quais transformam alguns bens em outros. Agentes lançam ofertas (*bids*) — de forma a maximizar seus lucros (função-utilidade) — a vários preços, mas todas as trocas ocorrem somente a preços de mercado, estabelecidos como pontos de equilíbrio associados a alocações ótimas [215].

Outros mecanismos de mercado conhecidos, como votações, leilões e barganhas, cujo objetivo maior está na alocação de recursos a mais eficiente possível, de sorte a maximizar as funções-utilidade — *payoff* — dos agentes (ou, pelo menos, de sua maioria), são possíveis também de serem empregados em circunstâncias onde há conflito de interesses, sendo que uma análise formal e detalhada desses mecanismos foi propiciada recentemente por Sandholm [407].

5.2.2.5 Organizações

Para se diminuir a complexidade dos processos de coordenação, padrões de interação recorrentes costumam ser capturados em estruturas preestabelecidas, carreando o surgimento de organizações. Segundo Fox [156], uma organização pode ser definida como “uma composição de estrutura e regime de controle”. Alguns dos padrões organizacionais analisados pelo autor são ilustrados na Fig. 5.2. Com relação aos regimes de controle, Fox sugere que estes podem variar desde relações de trabalho, onde processos (agentes) são diretamente supervisionados por outros via comandos já acordados, até “sistemas de preços” (uma espécie de “terceirização”), onde serviços passam a ser prestados ou contratados por entidades organizacionais independentes — muito se parecendo, pois, com a idéia da Rede de Contratos discutida na sequência.

De um modo geral, uma organização consiste de divisões e subdivisões, um conjunto de agentes alocados a essas divisões, um conjunto de *papéis* (*roles*) que tais agentes assumem e um conjunto de metas (geralmente, de longo prazo). Nesse âmbito, papéis são protótipos de funções a serem desempenhadas pelos agentes, descrevendo suas habilidades, direitos, obrigações e *autoridades* para com os demais. Sobre os papéis, é comum a existência de relações de *hierarquia*, onde um papel fica subordinado a outro, e de *especialização*, onde um papel pode especializar outro e herdar seus direitos, obrigações, autoridades, etc. [169].

Como enfatizado por Decker [117], há um consenso geral de que nenhuma estrutura organizacional é apropriada para lidar com todas as circunstâncias possíveis. Assim, à medida que as condições do problema mudam, deve-se reavaliar as potencialidades da estrutura atual para, se for o caso, reprojeta-la. Tal recurso, porém, não deve ser adotado a todo momento, pois tende a ter grande impacto (custo) sobre o desempenho operacional do sistema. Existem abordagens que tentam (re)projetar automaticamente as estruturas

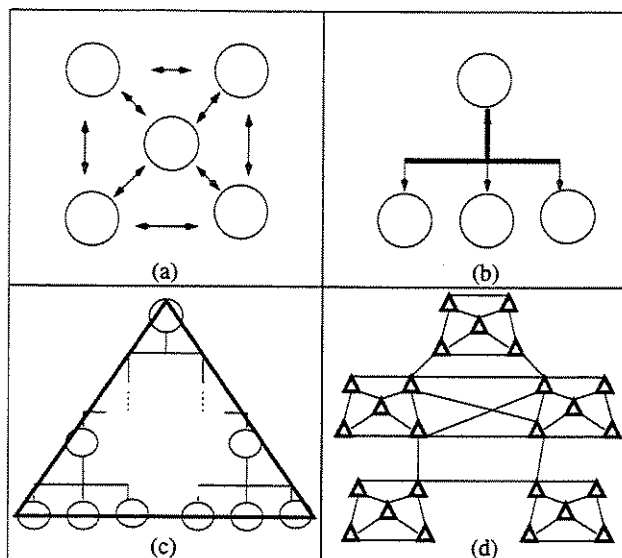


Fig. 5.2: Alguns padrões organizacionais [156]. (a) parceria (time); (b) hierarquia simples; (c) hierarquia multidepartamental; e (d) heterarquia (rede de hierarquias).

relacionais do sistema multiagente tendo em vista os requisitos do domínio da aplicação, dentre as quais se destacam aquelas voltadas a explorar aspectos de auto-organização entre os agentes [179] e aquelas de cunho evolutivo.

Além do conceito de time [206], que já trás predefinida a *necessidade* de sinergia entre os seus membros — uma vez que, como bem atestam Singh *et al.* [430], estes “*devem* alcançar um ou mais objetivos comuns de alguma natureza” —, existe também o conceito de *coalizão* (aliança ou coligação), tratando-se de estruturas organizacionais cooperativas de caráter mais dinâmico (e, portanto, mais efêmero) [429]. Diferentemente de times, em coalizões, os agentes não necessitam ter a benevolência como propriedade intrínseca, passando a se unir por razões mais *circunstanciais* (como assim o fizeram, recentemente, americanos e ingleses, quando decidiram “lutar contra o mal”). Em verdade, um agente pode pertencer a mais de uma aliança ao mesmo tempo, desde que tais grupos apresentem objetivos diferentes e não compitam entre si. O processo de formação de uma coalizão compreende três atividades principais [407]: (i) geração da estrutura organizacional da coalizão; (ii) otimização dos papéis dos agentes; e (iii) atribuição das recompensas aos membros à medida que atingem o objetivo da coligação, sendo esse objetivo geralmente de curto/médio prazo se comparado ao de um time.

5.2.3 Análise de algumas linhas de atuação

A partir dos primeiros trabalhos, realizados a partir da década de 1980 e lidando especificamente com sistemas RDP, várias propostas, seguindo diferentes orientações de pesquisa, vêm sendo constantemente

apresentadas na literatura, com o intuito de propiciar novas perspectivas para o problema da coordenação em IAD [38, 110, 117, 215, 355, 398, 429, 474, 475, 478]. Segue abaixo uma discussão condensada de algumas dessas abordagens.

O primeiro protocolo de negociação, criado dentro dos auspícios da RDP, ficou conhecido como *Rede Contratual* (*Contract Net* — CNET) [110, 435]. Esse protocolo é fundamentado nas noções de mercado econômico citadas anteriormente, sendo sua implementação considerada um mecanismo eficiente de alocação de tarefas em consonância com critérios de “disponibilidade” e “adequação”. O seu modo operacional pode ser descrito, de forma abreviada, como a seguir. Um agente “gerente”, responsável por uma dada tarefa, realiza previamente a sua decomposição e passa a enviar mensagens de anúncio, contendo descrições abstratas das subtarefas, para aqueles agentes da sua rede de contatos os quais ele acredita que tenham condições de resolvê-las (i. e., possíveis “contratantes”). Uma vez feito o *broadcasting* da mensagem relativa a uma subtarefa, somente aqueles agentes que realmente se sintam em condições de resolvê-la é que irão responder ao gerente, enviando-o a sua oferta (*bid*) pela prestação de serviço. De posse das propostas, ou o gerente convoca, segundo seus critérios de otimalidade, um dos contratantes, ou decide enviar uma contra-proposta a este, ou, ainda, decide por realizar um novo ciclo de anúncios e respostas (tipicamente, após efetuar uma nova divisão da subtarefa em subtarefas menores).

Como se percebe, os papéis dos agentes nesse *framework* não são especificados *a priori*, ou seja, qualquer agente pode atuar como gerente ou contratante em alocações diferentes. Essa característica concede uma grande flexibilidade ao sistema, pois um agente contratante pode decidir, num segundo momento, realizar uma outra decomposição da sua nova subtarefa e atribuir as partes resultantes a outros agentes, mediante um novo ciclo de negociações. Isso pode dar origem a formas descentralizadas de organização (coordenação) bem interessantes, além de garantir degradação suave de desempenho em caso de falhas dos agentes. Contudo, o uso demasiado de mensagens de anúncio, (contra-)proposta e atribuição pode acarretar sobrecarga ao sistema de comunicação, induzindo, muitas vezes, a problemas de eficiência (tempo de comunicação não inferior a tempo de computação). Pesquisas mais recentes estenderam e incrementaram esse protocolo para lidar com tais problemas e com situações onde os agentes não são benevolentes entre si (como em cenários de comércio eletrônico) [407].

Por sua vez, as primitivas sociais básicas de compromissos e convenções, introduzidas na Seção 5.1, também vêm sendo freqüentemente empregadas como meio eficiente de se garantir coordenação em sociedades contendo múltiplos módulos autônomos [231, 232], haja vista possibilitarem que os objetivos pessoais dos agentes se tornem vinculados uns aos outros. O uso de *commitments* permite que os agentes gerenciem melhor as suas agendas de trabalho, na medida em que fica aumentado o grau de *previsibilidade* acerca das ações futuras que irão ocorrer no ambiente. Uma vez que um agente só irá assumir novos compromissos se tiver condições para tal — são desconsiderados aqui agentes não-benevolentes ou egoístas —, evita-se, com isso, tempo e recursos desperdiçados com alocações de tarefas indevidas.

Mais ainda, quando agentes passam a trabalhar em equipe, eles devem se comprometer uns com os outros em relação ao objetivo comum, caracterizando assim um nível maior de pacto social. Como várias

ações comuns tendem a depender da participação de todo o time, uma simples quebra de compromisso por parte de um único membro pode pôr em perigo todo o esforço da comunidade. A fim de evitar situações como essa, é que existem as convenções de grupo (*joint conventions*), as quais estipulam os requisitos mínimos (e essenciais) para se criar e manter os compromissos do grupo (*joint commitments*). De forma resumida, nas palavras de Huhns e Stephens [215], “compromissos proporcionam a estrutura necessária para interações predizíveis, ao passo que convenções sociais provêem o nível necessário de suporte mútuo”.

Seguindo uma outra perspectiva, Durfee e Montgomery [131] abordam o problema de coordenação como uma instância de um processo de *busca distribuída* em um espaço projetado pelos possíveis comportamentos (individuais ou em grupo) dos agentes. Uma vez que esse espaço de comportamentos é arquitetado de maneira hierárquica (agentes formando times, times compondo grupos de times, etc.) e é distribuído entre os próprios agentes, a coerência do sistema fica assegurada na medida em que os agentes *necessitam* se comunicar freqüentemente para trocar informações sobre o estado do sistema, este modelado como o conjunto de todos os comportamentos. Dentro desse *framework* conceitual, cinco componentes-chave são identificados pelos autores: representação da hierarquia, método de busca distribuída, algoritmos de busca local, conhecimento de controle e métricas de avaliação do comportamento coletivo.

Como alternativa a essas abordagens mais centradas na configuração “*off-line*” de protocolos/mecanismos de coordenação, existem abordagens mais recentes que tentam explorar a conexão entre aprendizado e coordenação, com o intuito de prover maior flexibilidade aos agentes quando operando em ambientes dinâmicos, incertos e não-determinísticos [476, 477]. A idéia é possibilitar que os agentes possam aprender mais uns sobre os outros e sobre o ambiente, a partir de suas inúmeras interações, podendo, daí, decidirem melhor como se organizarem e, por conseguinte, coordenarem. Fundamental aqui é lidar com o *Problema de atribuição de crédito* (*Credit assignment problem*) que diz respeito a como se recompensar, de maneira justa, um dado agente por suas contribuições para todo o grupo. Isso se torna vital, por exemplo, em times e coalizões, onde, no primeiro caso, os agentes precisam receber realimentação sobre quão bem vêm atuando em prol do grupo (para, eventualmente, reorientarem sua linha de atuação em tempo) e, no segundo caso, os agentes só se mantêm agregados enquanto seus níveis de utilidade assim o determinarem. No contexto multiagente, a frase a capturar tal conceito seria: “que ação realizada por qual agente contribuiu em que dimensão para a mudança de desempenho do grupo?” [476].

Um dos primeiros trabalhos nessa direção, proposto por Weiss [474], investigou agentes reativos se comunicando de forma explícita para decidirem sobre que ações (individuais ou em grupo) deveriam tomar a fim de realizarem uma dada tarefa no ambiente (no caso, movimentação de blocos num tempo limitado). Mediante dois algoritmos de aprendizado por reforço criados por Weiss (ACE e AGE), os agentes aprenderam coletivamente a estimar a relevância de suas ações e, baseando-se nessas estimativas, a coordenarem suas ações para gerarem seqüências apropriadas de movimentação dos blocos. Outro trabalho sobre aprendizado por reforço em sistemas multiagentes foi conduzido por Sen *et al.* [419]. Nesse caso, os agentes não compartilham informação (nem têm noção da existência ou das habilidades uns dos outros), ao mesmo tempo que se emprega o algoritmo *Q-learning* [326] para que os agentes possam desenvolver, individualmente, políticas

de maximização do seu desempenho com base nas realimentações provenientes do ambiente.

Duas abordagens diferentes lidam com a estratégia de se prover aos agentes a capacidade de eles próprios aprenderem a atribuir seus papéis de acordo com a situação que vivenciam. Nagendra Prasad e colaboradores [378] apresentam uma metodologia de gênero probabilístico que combina raciocínio baseado em casos e aprendizado por reforço para habilitar membros de um grupo de agentes a selecionarem, adaptativamente, os seus papéis organizacionais, de modo a que suas funções se complementem. Nesse caso, um agente pode assumir um dentre vários papéis possíveis em uma dada situação. Já Stone e Veloso [444] investigam a eficácia de se ter times aprendendo a coordenar suas ações quando em confronto direto com times oponentes. O domínio estudado foi futebol de robôs em simulação (o mesmo empregado nos experimentos reportados na Seção 8.4). De acordo com essa abordagem, uma nova metodologia de aprendizado em camadas é adotada, onde o aprendizado de habilidades de baixo nível é seguido pelo aprendizado de tomadas de decisão de mais alto nível. Uma rede neural foi embutida nos agentes para que estes aprendessem a chutar a pelota numa certa direção, ao passo que uma árvore de decisão foi empregada para que eles aprendessem, posteriormente, a selecionar parceiros para realizar passes. Com essa abordagem em camadas, foi possível constatar o surgimento de algumas “situações de jogo” bem interessantes, típicas de partidas de futebol praticadas por humanos. Por exemplo, alguns jogadores aprenderam a se projetar em pontos futuros no campo, à espera de passes vindos de seus companheiros, sem que para isso houvesse suporte de algum canal de comunicação entre eles.

Com efeito, além do trabalho de Stone e Veloso [444], é válido mencionar também a existência de outras abordagens voltadas à especificação de estratégias de coordenação em ambientes onde não é possível haver comunicação direta entre os agentes. Fenster *et al.* [145] apresentaram uma proposta nesse sentido, sendo esta baseada no emprego de heurísticas especiais chamadas de *pontos focais*. Segundo os autores, “pontos focais referem-se a soluções proeminentes oriundas de uma interação, soluções para as quais os agentes foram atraídos”, fazendo alusão clara a situações análogas a equilíbrios em jogos. Um exemplo de aplicação dessa abordagem foi a de se estudar como agentes, que não se comunicam, podem se coordenar na escolha de objetos físicos similares, dentre vários disponíveis, usando somente seus pontos de vista locais (heurísticas próprias) acerca do ambiente. Seguindo uma visão mais orientada a agentes reativos de Vida Artificial (ver Seção 1.1), Franklin [157] examinou uma série de situações nas quais comportamentos coordenados puderam emergir sem a necessidade de planejamento ou comunicação. O que o autor chamou, na ocasião, de “coordenação via estigmergia” (*stigmergic coordination*) vem sendo hoje bem explorado por trabalhos na linha da Inteligência Coletiva [56], consoante a qual os agentes passam a representar possíveis soluções (que interagem entre si) para fins de resolução heurística descentralizada de problemas de otimização, busca ou aprendizado [126].

Finalmente, vale salientar a análise comparativa realizada recentemente por Frozza e Alvares [164] contemplando algumas das abordagens de coordenação mais conhecidas em SMAs. Para tanto, os autores propõem alguns critérios de avaliação de caráter mais prático (voltados para os requisitos das aplicações), tais como preditividade, adaptabilidade, características das aplicações a que se destinam, além de algumas

vantagens e desvantagens. Estudos comparativos como esse são interessantes como forma de auxiliar projetistas de sistemas IAD na escolha de políticas, protocolos e mecanismos de coordenação mais apropriados aos requisitos da aplicação em consideração.

5.3 Síntese do capítulo

Neste capítulo, foi propiciada uma revisão geral sobre sistemas computacionais compostos por múltiplos módulos autônomos, focando principalmente em aspectos relacionados à sua coordenação. Além de uma fundamentação teórica, incluindo definições e jargões da área, foi realizada uma análise pragmática de várias abordagens existentes na literatura, que tentam lidar com o problema da coordenação em sistemas multiagentes sob diferentes pontos de vista.

Situando a abordagem coevolutiva apresentada na Seção 8.4 no contexto deste capítulo, pode-se dizer que ela se insere na classe de modelos de coordenação dinâmicos ou ascendentes (*bottom-up*) [169], uma vez que propicia com que a organização espacial de um time de futebol de robôs (que fica responsável, pois, por sua coordenação) emergja em função das interações simbióticas positivas (vide Seção 6.7) entre espécies artificiais, sendo estas criadas, progressivamente, como forma de representar os papéis funcionais a serem exercidos pelos agentes em campo (p. ex., atacante, defensor ou ala).

Capítulo 6

Computação evolutiva e sua aplicação em sistemas modulares

“É mais freqüente que a confiança seja gerada pela ignorância do que pelo conhecimento: são os que conhecem pouco, e não os que conhecem muito, os que afirmam tão positivamente que este ou aquele problema nunca será solucionado pela ciência.”

Charles Darwin

O interesse em métodos computacionais inspirados em fenômenos relacionados à evolução de espécies biológicas tem sido crescente nas últimas décadas, culminando com o desenvolvimento metodológico e formal dos algoritmos evolutivos, alvo de estudo da disciplina conhecida como Computação Evolutiva. O propósito deste longo capítulo é o de introduzir os principais fundamentos dessa área, focando tanto em aspectos teóricos como de aplicação (atentando, sobretudo, à subclasse conhecida como Algoritmos Genéticos), tendo em mente o escopo de investigação desta tese delineado no Cap. 1.

Para tanto, o conteúdo deste espaço está dividido em duas partes. Na primeira, serão apresentados: (i) um arquétipo genérico de caracterização dos AEs, dando realce a suas propriedades fundamentais, seus componentes e alguns pontos críticos vinculados ao desempenho satisfatório desses algoritmos; (ii) uma discussão sucinta referente à teoria dos Algoritmos Genéticos, bem como uma análise de algumas implementações de seus possíveis componentes; (iii) uma descrição breve de alguns dos tópicos de estudo mais recentes na área; e (iv) uma explanação mais detalhada de duas correntes mais recentes de pesquisa em CE, Coevolução Artificial e Algoritmos Evolutivos Híbridos (AEHs), que serviram de base para os projetos apresentados nos Caps. 7 e 8. Aproveitando o ensejo, passa-se, na sequência, a discorrer um pouco sobre as propriedades de sistemas emergentes (*bottom-up*) e a situar alguns trabalhos envolvendo AEs nesse contexto. Na segunda parte, é traçado um panorama condensado acerca da aplicação de técnicas evolutivas ao conjunto de sistemas inteligentes modulares discutido nos capítulos anteriores, comentando sobre algumas abordagens proeminentes da literatura.

6.1 O que caracteriza um algoritmo evolutivo?

Como suporte teórico e fonte de inspiração para os AEs está o *Neodarwinismo* (ou *Teoria sintética da evolução*), corrente de pensamento fundamentada nos princípios da *seleção natural* e “sobrevivência do mais apto” [105] e nas leis da genética populacional. Consoante tal escola de pensamento, na natureza, indivíduos de uma certa *população* competem constantemente entre si pela obtenção de recursos escassos, como comida, abrigo, território, acasalamento, etc. Nesse cenário, os indivíduos que se mostrarem mais susceptíveis a sobreviver, em face das condições impostas pelo ambiente, e a atrair mais parceiros sexuais terão maiores *chances* de se reproduzir e de gerar, por conseguinte, maior prole. Isso significa que os *genes* compondo o material *genotípico* desses indivíduos, ditos “mais adaptados”, *tendem* a se fundir e a se disseminar mais rapidamente pela população, na medida em que *gerações* de seus descendentes passam a perpetuar as características físicas (que compõem o seu *fenótipo*⁽¹⁾) codificadas por esses genes. Ao longo desse processo, diz-se que as espécies *evoluem* gradualmente de modo a se tornarem cada vez mais ajustadas às pressões dinâmicas exercidas pelo ambiente onde vivem, incorrendo na elevação perene do *fitness* médio das populações em sucessão.

Do parágrafo anterior, pode-se depreender a existência de três *fatores evolutivos* atuando, amiúde e continuamente, sobre as gerações de indivíduos de uma dada espécie: *recombinação gênica*, *mutação* e *seleção* [15, 241]. As mutações representam perturbações auto-reproduzíveis na estrutura (seqüência de bases) dos genes, que podem alterar a “informação” neles codificada no momento em que esta é replicada. Destarte, os genes alterados podem condicionar novas características hereditárias, constituindo-se, portanto, na fonte primária de variabilidade. Já o fenômeno da recombinação gênica (*crossing-over* em cromossomos homólogos), relativo à reprodução sexuada, permite que genes de indivíduos distintos sejam agrupados em arranjos seqüenciais variados. Com isso, mutações ocorridas em diferentes indivíduos e em escalas de tempo distintas podem ser continuamente (re)combinadas, mantendo os níveis de diversidade fenotípica na população. Por fim, o fator “seleção” nada mais é do que um processo discriminatório, baseado em *competição*, entre os fenótipos dos indivíduos da população, implicando a reprodução diferencial entre as diversas combinações gênicas produzidas em cada geração.

O engenho evolucionário descrito acima pode ser analisado sob a ótica de um processo de otimização, busca ou aprendizado, servindo, pois, de metáfora para o desenvolvimento de métodos mais robustos e flexíveis orientados para a resolução de problemas computacionais e de engenharia [30, 159, 178, 320]. Com efeito, todos os algoritmos evolutivos compartilham uma base conceitual comum voltada à simulação de processos evolutivos *artificiais*, envolvendo, de um modo ou de outro, os seguintes componentes básicos [30, 31, 51]: (a) manutenção de *populações* de estruturas heterogêneas na forma de *cromossomos* (cadeias identificando possíveis soluções para o problema-alvo), que *competem* entre si e que evoluem, de geração em geração, em conformidade com regras de *seleção*; (b) aplicação de *operadores de variação* de cunho

⁽¹⁾De maneira sucinta, pode-se dizer que o fenótipo engloba todos os caracteres físicos (*traits*) apresentados por um organismo, sejam estes morfológicos, fisiológicos ou etológicos, traduzindo-se, pois, na manifestação do genótipo no ambiente.

estocástico (i. e., métodos heurísticos baseados nos fatores evolutivos supracitados) sobre os cromossomos, de maneira a se gerar novas estruturas mais “aptas” e se percorrer, eficiente e paralelamente, o espaço de busca de soluções; e (c) emprego de uma medida de adaptação — *função de fitness* — ao ambiente (problema), com o intuito de se distinguir entre soluções apropriadas ou não, privilegiando as primeiras quando da seleção dos indivíduos que comporão as próximas gerações.

Enquanto outros métodos e meta-heurísticas⁽²⁾ baseados em busca local, tais como métodos diretos, *descida do gradiente*, *Busca Tabu* e *Recozimento Simulado*, percorrem o espaço de pontos utilizando somente uma única solução-candidata por iteração, os algoritmos evolutivos o fazem com várias simultaneamente, exibindo uma estratégia de busca essencialmente *paralela*, destinada à alocação de maior número de indivíduos (e, portanto, recursos computacionais) às regiões mais promissoras. Métodos de otimização mais tradicionais têm propensão a apresentar, ainda, outros agravantes, conforme salientado por Deb [116]: a sua convergência à solução ótima depende, usualmente, de quão boa é a solução inicial escolhida; a maioria dos algoritmos, sobretudo aqueles calcados em informação do gradiente, tende a “capitular” ante a presença de ótimos locais (subótimos globais); nem sempre um algoritmo bem eficiente na resolução de um dado tipo de problema consegue manter esse patamar de eficiência na resolução de outros problemas (mesmo para aqueles similares); e tais algoritmos não costumam ser de fácil paralelização, acarretando ineficiência computacional. No que tange a esses inconvenientes, os AEs, cuja estratégia de busca cai na categoria de algoritmos de “geração-e-teste”, se apresentam como alternativas bem atraentes, fazendo uso de regras de transição heurística, de natureza não-determinística (estocástica), além de uma estrutura de implementação deveras flexível, capaz de ser customizada em congruidade com as nuances do problema-alvo.

Ainda que existam diversos métodos computacionais passíveis de serem agrupados sob o gênero “evolucionário”, pode-se alinhar um *framework* genérico comum a todas as variações. Para tanto, seja I um espaço arbitrário de indivíduos $a \in I$, e $F : I \rightarrow \mathbb{R}$ uma função de aptidão caracterizando uma superfície de avaliação (conhecida como *fitness landscape*). Usando μ e λ para denotar os números de pais e filhos, respectivamente, pode-se representar uma população em uma certa geração t como sendo $P(t) = (a_1(t), \dots, a_\mu(t)) \in I^\mu$. Seleção, recombinação e mutação são operadores atuando sobre toda a população⁽³⁾ e descritos como $s : I^\mu \rightarrow I^k$, $r : I^k \rightarrow I^\lambda$, e $m : I^\lambda \rightarrow I^\lambda$, respectivamente. Esses operadores dependem, tipicamente, de conjuntos de parâmetros adicionais Θ_s , Θ_r e Θ_m , os quais, por sua vez, são customizados em função das peculiaridades dos operadores e do tipo de representação das soluções utilizado.

Das definições acima, pode-se compreender um esquema de execução básico de um AE, sendo este alicerçado na seqüência de passos reproduzida no Alg. 3. No primeiro passo, a população de estruturas é encetada, geralmente de forma aleatória, em consonância com o formato adotado de descrição dos indivíduos. Em seguida, os indivíduos são avaliados conforme uma ou mais métricas de aptidão relativa (compondo a

⁽²⁾Sem perda de generalidade, pode-se denominar *heurísticas* todos os métodos aproximativos criados especificamente para resolver determinado tipo de problema em tempo polinomial, e *meta-heurísticas* aquelas heurísticas de uso mais geral, também referidas como “heurísticas de heurísticas”, que fornecem soluções de boa qualidade mas que não garantem encontrar, sempre, os pontos de ótimo global [467].

⁽³⁾Vale frisar que a mutação ocorre em nível de variável de decisão (gene), ao passo que seleção e recombinação atuam sobre conjuntos dessas variáveis (cadeias cromossômicas dos indivíduos).

função de *fitness*) determinadas pelo projetista. Entra-se, daí, num ciclo de iterações (gerações), onde os indivíduos passam a ser continuamente combinados, refinados, avaliados e selecionados até que uma condição de parada seja satisfeita, sendo esta possivelmente dependente de outros parâmetros (Θ_e) — por via de regra, tal condição se traduz em um número de iterações t_f predeterminado. A função “RECOMBINAR” indicará, via seleção probabilística, quais indivíduos procedentes da geração anterior serão capazes de se reproduzir mediante o cruzamento (*crossover*) de suas cadeias cromossômicas, sendo que o tamanho da prole gerada por um indivíduo é propensa a ser proporcional ao seu grau de aptidão. Na concepção dos novos indivíduos, estes podem ter parte de seus códigos alterados via mutação (ou mesmo *inversão* [207]) de seus genes, sendo esse(s) operador(es) encapsulado(s) na função “ALTERAR”. Disso, avalia-se o grau de adaptação da nova prole e passa-se a selecionar, dentre pais e filhos, aqueles que irão compor a próxima geração (papel da função “SUBSTITUIR”). No final da execução, o AE deve revelar a população referente à última geração, com indicação da melhor solução encontrada ao longo do processo.

Algoritmo 3 Algoritmo Evolutivo

Entrada: $\mu, \lambda, \Theta_e, \Theta_r, \Theta_m, \Theta_s$.

Saída: a^* , melhor indivíduo encontrado, ou $P(t_f)$, população relativa à última geração.

Computação:

$t \leftarrow 0$;

$P(t) \leftarrow \text{INICIAR\_POPULACAO}(\mu)$;

$F(t) \leftarrow \text{AVALIAR}(P(t), \mu)$;

while ($e(P(t), \Theta_e) == \text{false}$) **do**

$t \leftarrow t + 1$;

$P'(t) \leftarrow \text{RECOMBINAR}(P(t-1), F(t-1), \Theta_r)$;

$P''(t) \leftarrow \text{ALTERAR}(P'(t), \Theta_m)$;

$F(t) \leftarrow \text{AVALIAR}(P''(t), \lambda)$;

$P(t) \leftarrow \text{SUBSTITUIR}(P''(t), F(t), \mu, \lambda, \Theta_s)$;

end while

Apesar de aparentemente simples, os algoritmos evolutivos têm se mostrado bastante robustos, com habilidades para resolver, eficiente e eficazmente, problemas de elevada complexidade. Além disso, por não serem limitados por suposições antecipadas acerca do espaço de busca, tais como de continuidade, diferenciabilidade ou convexidade, sua aplicabilidade vem sendo demonstrada e ratificada progressivamente em diferentes domínios de atuação [30, 51, 75, 152, 153]. Em geral, todos os AEs são passíveis de serem agrupados em uma dentre cinco grandes subáreas da CE, ilustradas na Fig. 6.1 e descritas a seguir [30, 31, 37, 51, 137].

Algoritmos Genéticos (AGs): Compreende, de longe, o campo da CE com maior número de aplicações e extensões. O formato padrão de tais algoritmos foi proposto originalmente por John Holland como parte de suas tentativas em explicar os processos adaptativos ocorrendo em sistemas naturais e de construir sistemas artificiais calcados em tais processos [207]. A maior parte das análises experimentais e

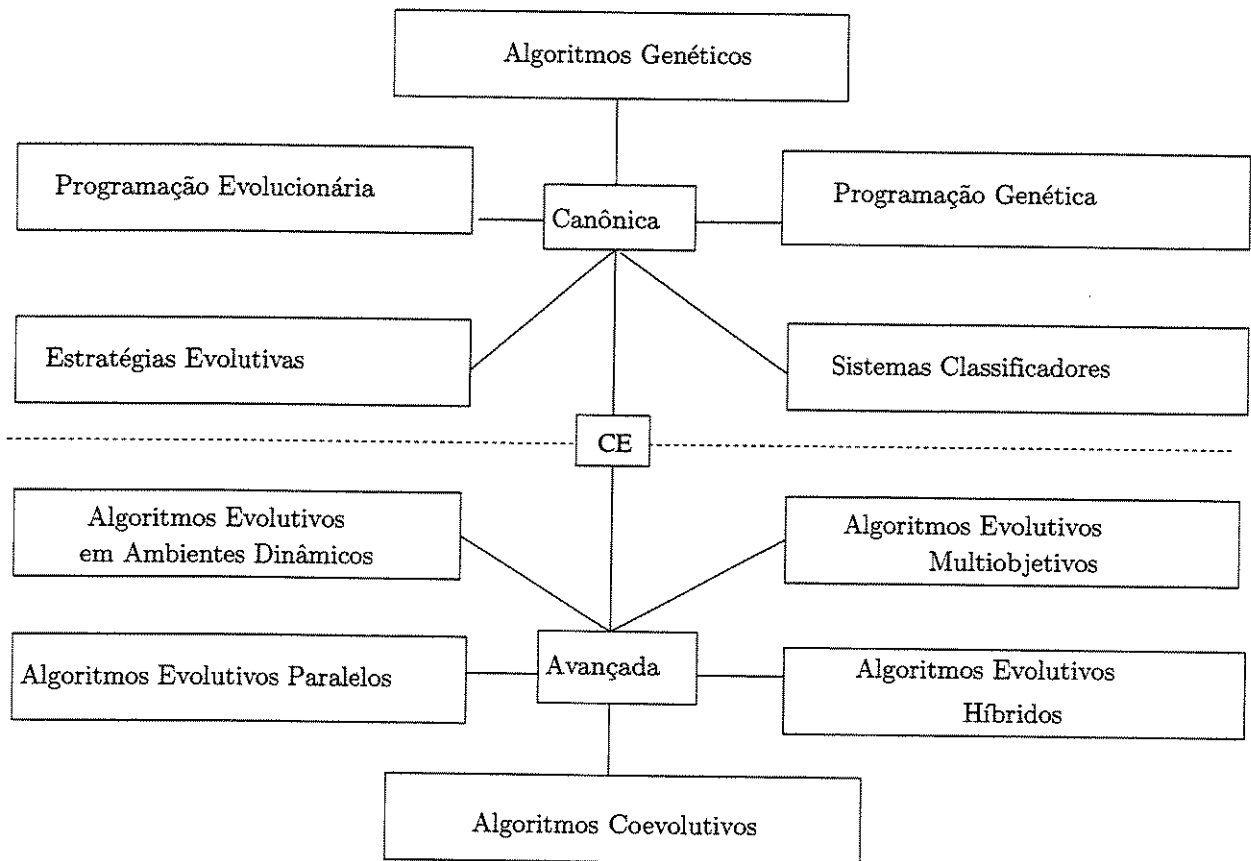


Fig. 6.1: Linhas de atuação (tradicionais e mais recentes) em Computação Evolutiva.

teóricas feitas sobre os AEs destina-se a explicar a lógica de funcionamento dos AGs (ver discussão condensada de parte desses trabalhos na Seção 6.3). Conceitualmente, tais algoritmos adotam dois espaços separados bem definidos, a saber: o *espaço de busca* e o *espaço de solução*. O primeiro compreende o espaço formado pelas soluções *codificadas* (conhecidas como *genótipos*) para o problema, ao tempo que o segundo espaço trata das soluções reais (i. e., *fenótipos*), sendo estas que irão ser avaliadas pela função de aptidão. Na terminologia da área [178], um parâmetro codificado é chamado de *gene*, cujos possíveis valores são denominados *alelos*. O algoritmo genético canônico, conhecido como AG *simples*, pode ser visto como uma “mistura” de três componentes: (a) uma representação em cadeias de *bits*; (b) operadores de recombinação (*crossover*), para intercâmbio de pedaços dessas cadeias, e de mutação simples, para variações aleatórias e independentes dos *bits*; e (c) seleção proporcional ao *fitness* (método da roleta) junto com substituição “geracional” ou *steady-state*. Tipicamente, os AGs empregados em problemas reais apresentam-se mais sofisticados que o canônico, pela incorporação de operadores de reprodução e seleção mais elaborados e customizados às características do problema-alvo. Alguns exemplos de tais componentes são apurados na Seção 6.4.

Estratégias Evolutivas (EEs): são costumeiramente aplicadas a problemas de otimização de parâmetros com valores reais, sendo estes, geralmente, discretizados previamente. Tradicionalmente, costuma-se admitir somente dois operadores: mutação gaussiana e *estratégias de substituição*, estas conhecidas como $(\mu + \lambda)$ e (μ, λ) , onde μ identifica o número de pais e λ , o número de filhos. Historicamente, também não há operação semelhante a de *crossover* em AGs, mas isso vem sendo modificado mais recentemente com a aplicação de operadores de recombinação linear ou de troca de genes. A característica mais atraente por trás de EEs tem sido a da *auto-adaptação* das variâncias das distribuições gaussianas (uma para cada variável de decisão a ser otimizada) associadas ao operador de mutação. O princípio básico, com isso, é o de se adicionar tais parâmetros aos genótipos dos indivíduos para que possam também ser evoluídos dinamicamente — retirando do projetista a carga de ter que especificar, *a priori*, tais valores.

Programação Evolucionária (PE): Inicialmente, essa classe de métodos estava mais dirigida à tarefa de evolução de autômatos de estado finito para fins de aprendizado de máquina. Decerto, tanto aspectos de descrição das soluções como os próprios operadores de variação estavam atados a esse ramo de aplicação. Mais ainda: originalmente, cada pai na população daria origem, via mutação gaussiana (similar à usada em EEs), a um único filho, sendo que, via um operador de substituição elitista, somente metade da população deveria ser mantida ao término de cada geração. A PE “contemporânea”, todavia, já evoluiu para usar outros tipos de representação (com parâmetros reais, tipicamente), além de variações estocásticas para o operador de substituição. Além disso, mais recentemente, Fogel [152] sugeriu uma variação, chamada de *meta-PE*, onde, assim como em EEs, as variâncias relativas às variáveis de decisão passariam também a ser adaptadas ao longo da execução do algoritmo. Contudo, contrapondo-se ao que ocorreu com EEs, algoritmos inspirados em Programação Evolucionária continuam a *não* empregar operadores de recombinação.

Programação Genética (PG): Constitui uma variação dos AGs dedicada à evolução automática e eficiente de rotinas algorítmicas, ou, em outros termos, à auto-programação de máquinas. Em PG, em vez de se servir de variáveis de decisão, lança-se mão de códigos de programa (escritos em uma dada linguagem

de programação conhecida, como C, C++, Java) como meio de representar possíveis soluções (normalmente bem complexas) para o problema sob análise. Desse modo, o formato de descrição das soluções é tópico crucial quando se passa a estudar PG, haja vista que um método de codificação só é considerado apropriado se garantir que as propriedades de *fechamento* (*closure*) e *suficiência* sejam atendidas de antemão, de modo a possibilitar consistência e convergência à solução final [37]. Para tanto, normalmente, as rotinas-solução são traçadas sob a forma de árvores de *parsing* relativas a expressões lógicas descrevendo modelos ou procedimentos. Assim, em PG, fica permitida a geração de cromossomos de tamanhos diferentes. Tais cromossomos já representam a solução, não havendo, pois, mapeamento “genótipo $\rightarrow$ fenótipo” como em AGs. Além disso, os operadores de variação passam a ser também customizados para manipular eficientemente essas representações não-lineares e devem ser projetados de maneira a [37] (i) estimular *parcimônia* (otimização do tamanho do código), ou seja, evitar a tendência de *bloating* (crescimento contínuo das árvores ao longo do processo evolutivo) por intermédio da eliminação de redundância (*introns* ou *junk genes*) e de blocos supérfluos (sem função aparente); (ii) evitar inconsistências (erros sintáticos); e (iii) possibilitar a fácil interpretação do modelo desenvolvido. Operadores de *permutação*, *edição* e *encapsulação* (definição de módulos), somados aos de recombinação e mutação, são artefatos comumente empregados para esse fim.

Sistemas Classificadores (SCs): Compreendem uma classe bem definida de agentes (ver Cap. 5) com capacidade de aprendizado por meio de regras de produção, empregando AGs como mecanismo primário de descoberta de tais regras (denominadas *classificadores*). Embora não exista uma versão canônica de SCs, todas as implementações compartilham de um conjunto comum de elementos, a saber: detectores, atuadores, base de regras, lista de mensagens e módulos de geração de regras, resolução de conflitos e de atribuição de crédito. Um ciclo operacional de um SC envolve tipicamente os seguintes passos: (1) detecção de mensagens postadas no ambiente; (2) casamento das mensagens com os antecedentes de todos os classificadores; (3) seleção dos classificadores com maior grau de casamento; (4) eliminação de possíveis conflitos entre as ações resultantes de tais regras selecionadas; (5) geração de mensagens para a lista de mensagens; (6) interpretação das mensagens pelos atuadores, convertendo-as em ações devidas; e (7) atuação do sistema de atribuição de crédito (tipicamente, o algoritmo *bucket-brigade* [207]), caso um sinal de recompensa (ou punição) seja recebido do ambiente, de sorte a atualizar as medidas de desempenho (*fitness*) dos classificadores, a serem usadas para geração (via AG) de novas regras. Variações mais recentes desses sistemas foram introduzidas na literatura e vêm sendo constantemente avaliadas por Wilson, Lanzi, Kovacs, Butz e outros colaboradores [277], tendo como base os modelos XCS (*Special Classifier Systems*) [489] e ZCS (*Zeroth-level Classifier Systems*) [488]. Tais variantes são endereçadas à produção de soluções mais compactas, compostas por classificadores mais precisos (*accurate*) e de máxima generalização.

Dentre tais “dialetos” da CE, todos concordantes com os mesmos fundamentos descritos acima, maior ênfase será dada, no decorrer da exposição feita nas próximas seções, aos AGs, haja vista os métodos evolutivos desenvolvidos para os projetos desta tese se enquadrarem nessa categoria.

6.2 Pontos críticos

Há dois aspectos importantes que devem ser seriamente considerados sempre que se pretende lançar mão das benesses de um AE particular. Um desses aspectos diz respeito à *preservação da diversidade genética* da população no curso do processo evolutivo. Isso porque os AEs pendem para alocar cada vez mais indivíduos aos subespaços de busca mais promissores, mediante a difusão, na população, dos genes dos indivíduos pertencentes a tais subespaços. O fato é que nem sempre as regiões iniciais aparentemente mais promissoras constituem áreas contendo verdadeiros ótimos globais, algo que leva o algoritmo, muitas vezes, a perder rapidamente informações relevantes acerca da qualidade de outros subespaços. Esse fenômeno de perda de diversidade é de suma importância tendo em vista o bom desempenho dos AEs, e ficou conhecido na literatura como *convergência prematura*. Uma alternativa, nem sempre muito eficiente, de lidar com problemas desse jaez é a de se empregar altas taxas de mutação (Θ_m), garantindo a introdução freqüente de novo material genético na população. Ainda nesse âmbito, existem também algumas frentes de atuação propondo mudanças conceituais ou estruturais aos AEs; constam dessa lista as estratégias de *niching* e especiação, os algoritmos evolutivos paralelos e os algoritmos coevolutivos, estes perscrutados mais adiante.

Por outro lado, a estratégia de busca por trás dos AEs requer um equilíbrio apropriado entre dois objetivos aparentemente conflitantes: o aproveitamento (*exploitation*) das melhores soluções obtidas até o momento e a exploração (*exploration*) adequada do espaço de busca. Esse dilema “*exploitation-exploration*” está muito longe de ocorrer quando se considera técnicas de busca mais convencionais. Por exemplo, algoritmos fundamentados em informação do gradiente aproveitam, geralmente, apenas a melhor solução na busca de possíveis aprimoramentos, sem realizar fases de exploração (*diversificação*) do espaço — são considerados, portanto, métodos de busca local. Por sua vez, métodos de busca aleatória constituem exemplos típicos de algoritmos que exploram bastante todo o espaço de busca mas que ignoram o aproveitamento (também conhecido como *intensificação*) de indivíduos em regiões mais promissoras. Já os algoritmos evolutivos se encontram no meio do caminho, resolvendo esse dilema sem levar em conta aspectos concernentes à superfície de busca. Isso é possibilitado via o emprego de um processo de busca multidirecional alternando etapas de exploração e de aprimoramento das soluções. Muitas vezes, a fase de intensificação é delegada a procedimentos de otimização local mais customizados às nuances do problema em questão, resultando no desenvolvimento dos AEs híbridos (ver Seção 6.6).

Em geral, há duas forças diretivas por trás do engenho de busca de um AE qualquer, quais sejam, variação e seleção. A segunda implica pressão em busca de qualidade, reduzindo a diversidade da população. A primeira captura todo esforço em direção à injeção de material genético novo, promovendo, por sua vez, o acréscimo de diversidade. Portanto, para se ter um AE efetivo, um balanço oportuno entre essas duas forças necessita ser bem estabelecido. Todavia, até o momento, não há muita teoria suportando a escolha correta dos pesos a serem atribuídos a tais fatores em função da natureza do domínio de estudo, algo quase sempre decidido após intensa experimentação [137].

6.3 Aspectos teóricos em AGs

A maior parte dos trabalhos em AGs tem sido de natureza experimental, com o propósito de estabelecer regras empíricas para seu correto e eficaz funcionamento. Não existe uma “teoria geral” que justifique as suas propriedades, conquanto algumas hipóteses interessantes têm sido anunciadas. A *Teoria dos Esquemas*, proposta por Holland [207], constitui a primeira tentativa rigorosa de explicar como os AGs simples funcionam. Um *esquema* alude a um certo padrão bem definido de valores de genes, o qual pode ser descrito (em código binário) por uma cadeia de caracteres sobre o alfabeto $\{0,1,\#\}$, sendo que o símbolo-coringa “#” significa “qualquer dos valores” (*don't care*)⁽⁴⁾. Um cromossomo *a* particular é dito conter um dado esquema *s* caso haja casamento entre *s* e a cadeia de *a*. Pela terminologia de Holland, a *ordem* de um esquema *s*, $o(s)$, designa o número de símbolos diferentes de “#” que *s* possui, enquanto o *tamanho de definição* de *s*, $td(s)$, refere-se à maior distância (em número de bits) entre os seus dois símbolos mais externos que não sejam de *don't care*. O fato é que Holland mostrou que o modo mais otimizado de explorar o espaço de busca seria o de alocar “oportunidades de reprodução” (*reproductive trials*) aos indivíduos em proporção ao seu grau de aptidão, fazendo, com isso, que os melhores esquemas se disseminarem pelo ambiente de maneira exponencial [45]. No mais, como normalmente um único indivíduo já contém diversos esquemas diferentes associados à sua cadeia, o número de esquemas sendo efetivamente processados em cada geração de um AG gira em torno de n^3 , com $n = \mu + \lambda$ indicando o tamanho da população. Essa propriedade, conhecida como *paralelismo implícito*, foi considerada por Holland como a explicação mais plausível para o grande poder e flexibilidade por trás dos AGs.

Goldberg [178] estendeu mais ainda esse formalismo ao introduzir o conceito de *blocos construtivos*, isto é, esquemas de baixa ordem, tamanho de definição pequeno e *fitness* acima da média, compostos por genes que “trabalham bem juntos”. Segundo o autor, para se aumentar as chances de êxito, um formato de codificação deveria encorajar, já de antemão, a possibilidade de formação freqüente, ao longo do processo evolutivo, de blocos construtivos; para tanto, tal formato deveria asseverar, na medida do possível, a justaposição de genes (variáveis de decisão) que de alguma forma se relacionem. Por outro lado, seria bom que houvesse o mínimo de *interação* (interdependência) entre genes pertencentes a blocos construtivos distintos. Tal interdependência, conhecida como *epistasia*⁽⁵⁾, significa que o papel (efeito fenotípico) de um dado gene fica condicionado ao papel exercido por outro(s) no mesmo cromossomo⁽⁶⁾. Em verdade, tais condições são dificilmente satisfeitas *a priori*, devendo ser tratadas mais como *recomendações* do que como regras [45].

Por fim, cabe aqui salientar que Holland [207] também demonstrou que um AG combina passos de

⁽⁴⁾Em geral, considerando alfabetos de cardinalidade k e cadeias de tamanho l , há $(k+1)^l$ possíveis esquemas [178].

⁽⁵⁾Os geneticistas empregam o termo “epistasia” num sentido um pouco diferente: para indicar que a presença de um ou mais genes de um dado locus *suprime* o efeito de expressão de um outro gene em outro locus [15, 241].

⁽⁶⁾O problema de interações epistáticas passa a ser bem saliente, por exemplo, ao se aplicar AGs simples para a otimização de arquiteturas de RNAs, uma vez que o papel exercido por um neurônio da rede só é interpretável se considerarmos os papéis exercidos por todos os neurônios a um só tempo (alto acoplamento entre as partes). Além disso, como os parâmetros (pesos) de uma RNA não têm interpretação prévia associada, fica muito complicado identificar antecipadamente quais neurônios devam ser adjacentes quando da definição do formato de representação da solução.

diversificação e de intensificação de modo otimizado, servindo-se da analogia do “bandido com k braços” (*k-armed bandit analogy*) [178]. Embora teoricamente correto, tal resultado, por se firmar em suposições simplificadas (p. ex., população de tamanho infinito e genes não interagindo significativamente), não garante convergência ao ótimo global, haja vista a possibilidade, na prática, de ocorrerem problemas ou erros de gênero estocástico (tais como o da *deriva genética*). De outro modo, via análise apoiada em funções especiais (funções de Walsh) e em cadeias de Markov, tem sido possível identificar classes de problemas que se mostrem de difícil trato por parte dos AGs. Normalmente, em tais problemas, ditos *deceptivos* [178], o *fitness* médio de esquemas que não pertençam ao ótimo global passa a ser maior que o *fitness* médio daqueles pertencentes, conduzindo, pois, o AG a explorar, erroneamente, regiões inapropriadas do espaço de busca [30].

6.4 Componentes de algoritmos evolutivos

Devido ao alto grau de flexibilidade exibido pelo arquétipo conceitual mostrado anteriormente, é possível se projetar variantes dos métodos evolutivos mais ajustadas aos requisitos das aplicações-alvo. Isso pode ser feito mediante a escolha oportuna dos vários componentes que integrarão a instância final de AE a ser usada. Nesta seção, serão analisados, concisamente, os papéis desses componentes, e comentadas algumas de suas possíveis implementações no contexto de AGs.

Representação: Pelo que foi exposto anteriormente, pode-se concluir que a escolha da codificação das possíveis soluções seja uma das etapas mais críticas na configuração de um algoritmo evolutivo. Com efeito, uma definição inadequada das estruturas cromossômicas pode causar problemas de convergência prematura ou mesmo inviabilizar a busca eficiente pelo ponto de ótimo global, visto que é via a etapa de representação que se passa a definir, indiretamente, o formato peculiar ao espaço de busca. Por exemplo, caso um cromossomo venha a codificar l parâmetros de uma função (assumindo representação por valores reais), então o espaço de busca passa a ser definido no $\mathbb{R}^l$.

Em AGs, a estrutura de um cromossomo via de regra captura uma solução completa, codificada no espaço genotípico via vetores de tamanho finito (estes, por sua vez, definidos sobre alfabetos de cardinalidade também finita). A representação binária, por ser de fácil manipulação e análise, é normalmente a mais utilizada, conquanto não seja, necessariamente, a mais conveniente a todas as ocasiões — p. ex., se um problema tem parâmetros contínuos e o projetista deseja elevados níveis de precisão, provavelmente, usando codificação binária, acabará com longos cromossomos para denotar as soluções, implicando a necessidade de maior quantidade de memória, e, às vezes, de maior número de gerações para convergência ao ótimo.

De fato, como os parâmetros de vários dos problemas a serem tratados tendem a assumir valores reais (ou inteiros), nada mais óbvio do que se utilizar de representações não-binárias para esses casos. Da mesma forma que para alfabetos binários, já existe também algum suporte teórico comprovando a eficácia de codificações inteiras ou de ponto flutuante⁽⁷⁾. Uma vantagem de se usar representações não-binárias em AGs

⁽⁷⁾Goldberg, por exemplo, também introduziu o conceito de *alfabetos virtuais* para mostrar que representações de

é a de viabilizar o desenvolvimento de operadores genéticos mais otimizados e de mais fácil interpretação. Por fim, mais recentemente, vem sendo comum o emprego de codificações *mescladas* (*mixed representations*) de cromossomos, combinando partes binárias com não-binárias. Nos experimentos reportados nesta tese, os três tipos de representação, binária, não-binária e mesclada, foram utilizados.

Função de aptidão: A função de aptidão (*fitness*) deve capturar o grau de “utilidade” por trás das estruturas geradas, tendo como referencial os requisitos impostos pelo problema. Constitui o principal *ponto de controle* por parte do projetista sobre a qualidade da solução final a ser obtida pelo AE. É mediante, pois, o projeto dessa medida de adaptação relativa que o usuário pode embutir, de antemão, grande parte do seu conhecimento acerca do domínio de atuação. Contudo, nem sempre é factível se projetar tal função de maneira precisa e objetiva (p. ex., via equações ou regras), devendo-se, nesse caso, recorrer a outras formas de avaliação mais subjetivas. Para tanto, existem variantes dos AEs, chamadas de *AEs interativos* [51], que estipulam que as soluções produzidas sejam avaliadas por um ou mais especialistas humanos. Ainda nesse assunto, existem também trabalhos empregando funções de *fitness fuzzy* [332] como meio de se capturar incerteza ou vagueza na apreciação das soluções.

No geral, as funções de aptidão podem ter um ou mais objetivos, podem ser uni- ou multimodais, discretas ou contínuas, suaves ou ruidosas, estáticas ou variantes com o tempo. Qualquer que seja o perfil da função empregada, o fato é que o processo de avaliação dos indivíduos compreende, quase sempre, a etapa mais computacionalmente custosa em AEs — isso fica ainda mais patente em aplicações de projeto automático ou que envolvam a execução de demoradas simulações, como aquelas abordadas nesta tese. Existem algumas propostas que tentam lidar com tal dificuldade, via de regra pelo emprego de técnicas de *fitness aproximado*, consoante as quais a função de *fitness* original passa a ser substituída, gradualmente (i. e., ao longo da execução do AE), por outra menos refinada ou precisa, porém com menos carga computacional [45]. RNAs podem também ser usadas para esse propósito.

Iniciação da população: A população inicial é tipicamente criada via amostragem aleatória (com distribuição uniforme ou gaussiana) do espaço de busca. Contudo, em alguns casos, é possível também se “inocular” algumas boas soluções, previamente conhecidas, na primeira população — provavelmente, oriundas de outro(s) método(s) de busca ou de execuções anteriores do mesmo AE (ver também Seção 6.6).

Operadores de seleção e substituição: A Teoria sintética estabelece que os indivíduos mais adaptados são aqueles com maiores chances de se reproduzirem e sobreviverem. Esses dois fenômenos, reprodução e sobrevivência, são capturados na estrutura dos AEs mediante o emprego de operadores de *seleção* (de pais para reprodução) e de *substituição* (de pais por filhos mais aptos). Tais operadores diferem entre si em um aspecto essencial: durante a fase de seleção, um mesmo indivíduo *pode* ser escolhido várias vezes, ao passo que, na etapa de substituição, cada indivíduo (entre pais e filhos) ou é mantido (por, pelo menos, mais uma geração) ou é descartado para sempre.

Existem várias implementações de métodos de seleção, sendo que a maioria delas *tem inclinação* a alta cardinalidade acabam convergindo para aquelas de baixa cardinalidade ao longo das gerações de um AG; a partir disso, fica garantida, via Teoria de Esquemas, a possibilidade de convergência para ótimos globais [45, 116].

favorecer os mais adaptados, atribuindo a estes maiores probabilidades de serem escolhidos. Esses operadores são conhecidos como de *seleção proporcional ao fitness*. O *método da roleta*, de longe o mais usado, se enquadra nessa categoria, segundo o qual cada indivíduo da população passa a ser representado por uma fatia de uma roleta com tamanho proporcional ao seu grau de aptidão. A cada giro da roleta (o número de giros é definido pelo tamanho da prole a ser gerada), o indivíduo cuja faixa associada contiver a posição final apontada pela agulha será o selecionado. Já o *método de seleção por torneio* de ordem k estipula que k indivíduos da população, escolhidos aleatoriamente, tenham seus valores de *fitness* comparados entre si, sendo o campeão (indivíduo selecionado) aquele com maior grau de aptidão no grupo. Outro método muito elegante e teoricamente mais recomendado [45, 116] é o da *amostragem universal estocástica* (SUS — *Stochastic Universal Sampling*), proposto por Baker [33], que compreende uma variação mais robusta e sofisticada do método da roleta visando a evitar problemas de convergência prematura. Aqui, em vez de uma única agulha, k agulhas igualmente espaçadas são utilizadas, sendo k o número de indivíduos a serem selecionados. Portanto, em vez de k vezes, a roleta é girada uma única vez, exibindo menos variância que as repetidas chamadas do método da roleta. Além dos operadores de seleção proporcional ao *fitness*, outros comumente adotados são: *seleção biclassista*, mediante a qual uma parcela percentual dos melhores indivíduos e outra dos piores são selecionadas para a próxima geração, e *seleção por diversidade*, por meio da qual os indivíduos mais diversos, genotípica ou fenotipicamente, da população (comumente, em relação ao melhor atual) são mantidos.

Outrossim, os métodos supramencionados podem ser aplicados para fins de substituição de pais por filhos. Ainda nesse contexto, duas estratégias opostas são em certas ocasiões empregadas: ou se substitui todos os pais pelos filhos (i. e., *substituição geracional*) ou apenas um único pai é trocado a cada geração (i. e., *substituição steady-state*). É mais comum se optar pelo meio-termo via o uso de um *intervalo geracional* (*generation gap*) definindo o percentual de indivíduos a serem substituídos a cada iteração.

Finalmente, é pertinente destacar alguns métodos de remapeamento de *fitness* destinados igualmente a evitar problemas de convergência prematura. Tais métodos podem ser aplicados em conjunção com os operadores descritos anteriormente. São exemplos de métodos de remapeamento explícito de *fitness* [45]: “escalamento de *fitness*” (*fitness scaling*), “janelamento de *fitness*” (*fitness windowing*) e “ordenação de *fitness*” (*fitness ranking*).

Operadores de variação: Pela discussão conduzida na Seção 6.2, além da força diretiva de seleção, deve-se garantir também a injeção constante de diversidade genética em AEs. Isso fica afeiçoado mediante o uso apropriado de operadores especiais, chamados de variação. Exemplos típicos são os de *crossover* (cruzamento ou recombinação) e mutação. Operadores de cruzamento ficam responsáveis pela combinação de características dos pais durante a reprodução, permitindo que as próximas gerações herdem tais características. A justificativa teórica para se empregar tais operadores advém diretamente do conceito de blocos construtivos mencionado anteriormente: como os bons valores de *fitness* dos pais são devidos à qualidade dos blocos de genes que compõem as suas cadeias genotípicas, então a recombinação de pedaços dessas cadeias *tende* a possibilitar a justaposição de características interessantes, levando ao acréscimo dos valores de

fitness da prole resultante. O efeito de tais operadores varia de explorativo, quando a população se encontra altamente diversificada, a intensificativo, quando ela passa a se isolar em regiões mais promissoras [137]. Existem vários operadores de recombinação na literatura [30], a maioria dirigida aos AGs ou algoritmos de PG. Para AGs, os mais utilizados são: *crossover de um ponto* (escolhe-se aleatoriamente uma posição de corte, que servirá de pivô para a troca das partes), *crossover multipontos* (escolhendo-se k pontos de corte aleatórios) e *uniforme* (por intermédio de uma máscara de zeros e uns, define-se quais genes serão intercambiados (“1”) ou mantidos (“0”)). Existem variações desses operadores para se trabalhar com múltiplos pais ao mesmo tempo [51] ou para se manipular representações não-binárias — p. ex., *crossover aritmético*, *crossover geométrico*, *crossover extensivo*, *crossover simplex* e *crossover fuzzy* [31, 45, 116, 320].

No caso do operador de mutação, considerado por muitos como de papel secundário (de *background*) para AGs, este fica responsável por introduzir transformações esporádicas (estocásticas) em cada um dos cromossomos (operador unário). O seu papel é o de assegurar que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca seja nula. Para tanto, *idealmente*, ao se fazer uso da mutação, um balanço bem adequado entre diversificação e intensificação deve ser mantido: mutação elevada é interessante por razões teóricas (para garantir a ergodicidade do processo estocástico intrínseco) e, às vezes, práticas (única forma de se reintroduzir, rapidamente, diversidade genética ao final do processo evolutivo); contudo, altas doses de perturbações podem transformar o AE num simples processo de passeio aleatório (*random walk*). Portanto, não há um padrão geral de uso desse operador, sendo que a estratégia mais aceita é a de se empregar taxas de mutação (Θ_m) com baixas magnitudes ($0,001 \leq \Theta_m \leq 0,5$). Para AGs, o operador de mutação simples é o mais usualmente empregado. Sua lógica é: para cada gene, produz-se um número aleatório, verifica-se se este é menor ou igual a Θ_m , e, em caso positivo, troca-se o valor corrente por um outro alelo. Extensões para números reais incluem *mutação gaussiana*, *uniforme*, *não-uniforme*, *polinomial*, *creep* e *creep geométrico* [45, 116, 320]. Existem também outros operadores de variação, como os de *inversão* e *permutação* de cadeias, mais direcionados para aplicações na área de otimização combinatorial [30, 45, 320].

Parâmetros de controle: Há ainda poucos estudos teóricos acerca da influência da escolha do critério de terminação sobre o desempenho final de um AE. Normalmente, o critério de parada escolhido é o de se estabelecer, em princípio, uma quantidade fixa de gerações ou de número de avaliações. Outra alternativa é a de terminar o AE caso não haja melhorias no *fitness* médio (ou do melhor indivíduo) da população por uma certa faixa de tempo.

Por outro lado, é visível que AEs apresentam uma grande quantidade de parâmetros de controle a serem configurados oportunamente ($\mu, \lambda, \Theta_e, \Theta_r, \Theta_m, \Theta_s$ — ver à página 117). O maior problema a esse respeito é que, mesmo considerando um só desses parâmetros, fica custoso prever, antecipadamente, qual será o real impacto exercido pela escolha de seu valor de configuração sobre o desempenho final do AE como um todo. Fica ainda mais difícil, senão impossível, de compreender a influência combinada de todos os parâmetros ao mesmo tempo. Sendo assim, muitas vezes, passa a se recorrer a várias execuções diferentes do mesmo algoritmo com o intuito de se descobrir uma calibração que seja, pelo menos, adequada ao problema em vista. Tal estratégia de “tentativa-e-erro” pode ser, às vezes, ineficiente, incorrendo em grande dispêndio

de tempo. Outra possibilidade, sugerida por Eiben e Schoenauer [137], é a de se empregar métodos estatísticos (como ANOVA) para esse fim.

Outro segmento de investigação envolve a criação de mecanismos que permitam que o próprio modelo de AE calibre seus parâmetros ao longo de sua execução. A esse respeito, Eiben *et al.* apresentam três possíveis classes de abordagens [136]: (i) *determinísticas*, em que o valor de algum parâmetro estratégico é alterado via uma regra determinística (p. ex., mudança de valor após certo número de gerações); (ii) *adaptativas*, segundo as quais alguma forma de realimentação advinda do processo de busca da melhor configuração é usada para o propósito de se determinar a próxima direção ou mudança de magnitude a ser seguida; e (iii) *auto-adaptativas*, lançando mão do artifício mencionado anteriormente de se empregar evolução sobre evolução (i. e., *meta-evolução*), adaptando os parâmetros de controle juntamente com as variáveis de decisão. Como exemplo da última classe encontra-se a abordagem hierárquica *meta-GA*, proposta por Grefenstette [183], onde, num nível inferior, uma instância de AG é usada para otimizar as variáveis de decisão do problema-alvo, ao tempo que, num patamar superior, uma outra instância de AG é empregada para otimizar os parâmetros de controle do AG associado à camada de baixo.

6.5 Tópicos avançados

Uma vez que, de acordo com os teoremas NFL⁽⁸⁾ (*No-Free-Lunch*) [493], não existe um único algoritmo capaz de resolver, com máxima eficácia, *todos* os problemas (de otimização) existentes, a questão de se AEs são de qualidade inferior/superior a outros métodos alternativos fica sem cabimento. Mais atraente é poder questionar acerca dos “prós e contras” desses algoritmos. Nesse sentido, a Tab. 6.1 traz um paralelo entre alguns dos pontos positivos e negativos exibidos pelos algoritmos evolutivos que seguem a estrutura “convencional” apresentada na Seção 6.1. A despeito do fato dos AEs se mostrarem bem apropriados para abordar uma ampla gama de problemas complexos, inclusive aqui os de engenharia de sistemas inteligentes, algumas melhorias em sua estrutura básica podem e vêm sendo propostas de sorte a contornar seus pontos desfavoráveis. Nesse sentido, pelo menos quatro classes de abordagens podem ser detectadas, as quais são discutidas *ein passant* a seguir (ver Fig. 6.1).

A primeira classe foca na *manutenção de diversidade* na população e inclui os métodos de *niching*, os *algoritmos evolutivos paralelos* e os *algoritmos coevolutivos* (estes descritos mais à frente). Os métodos de *niching* e especiação [103, 303, 409] são aqueles que postulam o compartilhamento de recursos entre membros

⁽⁸⁾De maneira informal, o que tais teoremas demonstram é que o número de domínios para os quais um dado algoritmo *A* prevalece, em desempenho, sobre um outro algoritmo *B* é, em média, equivalente ao número de domínios para os quais o inverso é verdadeiro (ou seja, *B* supera *A*). Por conseguinte, pode-se concluir, por exemplo, que, ao se levar em consideração o espaço formado por todas as superfícies de busca passíveis de serem pesquisadas, *não* há como se asseverar, de antemão, que um certo AE irá, necessariamente, desempenhar um papel melhor (em termos de eficácia) que qualquer outro algoritmo, por mais ingênuo (*naïve*) que este seja (p. ex., aqueles baseados em busca aleatória). Resumidamente: de acordo com tais teoremas, não há “almoco gratuito”! Ou seja, sem customizar um algoritmo para o domínio-alvo, não há nenhuma garantia de que ele será bem-sucedido em sua aplicação. Isso abre um precedente forte para o desenvolvimento de estratégias evolutivas híbridas (Seção 6.6).

Tab. 6.1: Alguns pontos fortes e fracos dos AEs tradicionais.

Pontos fortes	Pontos fracos
Grande flexibilidade em sua estrutura	Convergência prematura nem sempre evitável
Nenhuma imposição de continuidade, convexidade e diferenciabilidade	Dificuldade de representação e manipulação de estruturas complexas
Relativamente insensíveis a ruídos (robustez)	Escalabilidade, de certo modo, limitada e tempo computacional, muitas vezes, elevado
Paralelização intrínseca (vários blocos construtivos avaliados simultaneamente)	Heterogeneidade comprometida (mesma interpretação genotípica para todos os indivíduos: uma única espécie)
Aplicáveis a uma grande gama de problemas	Destinados, principalmente, a problemas mono-objetivo e com função de <i>fitness</i> estática e não-ruidosa

semelhantes do ecossistema, trazendo uma vertente mais ecológica para os AEs. Consoante a conceituação proposta por Mayr em meados do século passado, pode-se definir *espécies* como “grupos de populações naturais que se cruzam real ou potencialmente e que são reprodutivamente isolados de outros grupos desse tipo” [312]. Ou seja, se os membros de uma espécie podem cruzar entre si, mas não com indivíduos de outras espécies, pode-se concluir que o *patrimônio genético* de uma espécie nunca se misturará aos de outra. Já *especiação* é o processo por meio do qual uma única espécie se diferencia em duas (ou mais) outras espécies que passam a ocupar diferentes nichos. Finalmente, o conceito de *nicho ecológico* designa as funções que uma espécie desempenha no ecossistema, incluindo aqui o seu “modo de vida” e suas interações com outras espécies — muito similar, portanto, ao conceito de nicho apresentado no âmbito de SMAs, vide Cap. 5. O mote distintivo apreendido pelos métodos de *niching* é que os recursos de um dado nicho ecológico são escassos e limitados, sendo o número de organismos nele contidos determinado pelo seu grau de fertilidade em relação ao ecossistema como um todo, bem como pela potencialidade de cada organismo em explorar tal fertilidade.

Um dos primeiros estudos direcionados à indução de nichos em AGs resultou no *mecanismo de pré-seleção*. Nesse esquema, um pai é substituído por seu descendente se o valor de aptidão do segundo for maior que o do primeiro. Dessa maneira, a diversidade é mantida na população porque a substituição ocorre somente entre indivíduos similares. Uma variação desse mecanismo, denominada *crowding* e concebida por De Jong em sua tese de doutorado [237], propõe a permuta de indivíduos numa forma mais otimizada. Aqui, novos cromossomos somente substituirão outros que sejam similares e que tenham menor valor de aptidão. Existe uma variação desse mecanismo, chamada de *crowding determinística* [303], que permite que pais e filhos pertencentes a nichos distintos mas bem similares possam competir entre si. Ainda nesse assunto, é válido discorrer um pouco sobre os *mecanismos de compartilhamento de recursos* (*fitness sharing*), propostos originalmente por Goldberg e Richardson [178]. O preceito original de *fitness sharing* é o de *reduzir* o valor de aptidão de indivíduos que sejam altamente similares entre si dentro da população. Um esquema prático

para alcançar tal intento é via o uso de uma *função de compartilhamento* que determina a *vizinhança* e o *grau de compartilhamento* para cada indivíduo da população. Logo, se um organismo é idêntico a outro, o grau de compartilhamento de *fitness* entre eles será igual a unidade. Depois de acumular o número total de compartilhamentos relativos a um dado cromossomo, tal número servirá de métrica de ajuste de seu valor de aptidão. No mais, o método de *clearing* [409] constitui uma variação mais robusta e eficiente (computacionalmente) sobre *sharing*, onde os recursos à disposição de um certo nicho, em vez de serem divididos por todos os seus membros, serão atribuídos somente aos k organismos desse nicho com maiores valores de aptidão.

A segunda classe de abordagens visa a melhorar a capacidade de busca (eficiência) dos AEs bem como o seu desempenho computacional. Enquadram-se aqui os métodos de hibridização descritos na próxima seção, os AEs com dominância (alelos dominantes e recessivos) e poliploidia (múltiplos cromossomos caracterizando um indivíduo) e também as abordagens coevolutivas e de paralelização. Dentre os modelos de AEs paralelos, três subclasses podem ser depreendidas [71]: (a) *Algoritmos Paralelos Globais*, onde, mediante o uso do paradigma “mestre-escravo”, uma unidade-mestre atribui certas funções do algoritmo (tipicamente, a tarefa de avaliação dos indivíduos) a outras unidades-escravo; (b) *Algoritmos Paralelos Multipopulacionais*, particularizando-se aqui os de natureza distribuída (i. e., AEs distribuídos ou de granulosidade grossa) e os de perfil maciçamente paralelo (i. e., AEs celulares ou de granulosidade fina); e (c) os *Algoritmos Paralelos Híbridos*, que combinam características das duas subclasses anteriores. Em se tratando, particularmente, dos AEs distribuídos, estes seguem o modelo ecológico de *ilhas isoladas*, proposto por Sewall Wright [30]. Destarte, uma grande população *panmítica* é substituída por várias subpopulações pequenas semi-isoladas (*demes*), cada uma ligada a um AE independente e explorando, possivelmente, regiões promissoras distintas do espaço de busca total. Os *demes* podem, ocasionalmente, *interagir* com seus vizinhos via intercâmbio de alguns de seus indivíduos, simulando um processo migratório sazonal (conhecido como *migração* ou *swapping*) por meio do qual novo material genotípico pode ser injetado em cada população, evitando, daí, problemas de convergência precipitada. Nesse tema, vale mencionar o trabalho conduzido recentemente por Garcia [170] direcionado ao desenvolvimento e análise de desempenho de algoritmos meméticos (ver Seção 6.6) paralelos customizados para tratar problemas não-triviais de otimização combinatória — nominalmente, caixeiro viajante assimétrico e seqüenciamento de tarefas em máquina simples ou máquinas paralelas.

A terceira classe engloba abordagens dedicadas à manipulação eficiente de representações mais complexas. Inseridos aqui, no escopo dos AGs, encontram-se os *messy-GAs* [30, 51], os quais lançam mão de um estratégia de codificação exótica — calcada em pares ordenados de índices e variáveis de decisão e voltada para a manipulação *explícita* de blocos construtivos — para cromossomos de tamanhos variáveis, estes produzidos via um processo evolutivo em dois estágios (*loops*). Outro exemplo diz respeito aos *AGs estruturados* [106, 108], que estipulam um esquema de codificação hierárquico e com redundância, em que se permite que partes dos cromossomos de um nível inferior sejam habilitadas ou desabilitadas (“ON/OFF”), ao longo do processo evolutivo, por *genes de controle* pertencentes ao patamar imediatamente superior (cujos valores podem ser também adaptados).

A última classe de estudos destina-se a tratar de problemas envolvendo múltiplos critérios de avaliação ou situações em que a função de *fitness* pode variar dinamicamente ao longo da execução do AE. Para tratamento de problemas multiobjetivos, a aplicação de AEs parece ser bem conveniente haja vista estes, por decisão de projeto, já manipularem uma população de soluções, permitindo que se possa obter diversos membros do conjunto ótimo de Pareto em uma única execução do algoritmo [95]. A grande maioria das extensões dos AEs voltadas para problemas de otimização vetorial emprega métodos de seleção específicos a fim de disseminar os indivíduos da população ao longo da fronteira de Pareto (i. e., melhor conjunto de compromissos⁽⁹⁾ entre os objetivos) inerente ao problema. Estratégias de *niching* e de *ranking* são comumente aplicadas nesse sentido, dando origem a uma série de novos procedimentos, dentre os quais pode-se citar: MOGA (*Multi-Objective Genetic Algorithm*), NSGA-I/II (*Nondominated Sorting Genetic Algorithm*), NPGA-I/II (*Niched-Pareto Genetic Algorithm*) e PAES (*Pareto Archived Evolution Strategy*) [95, 116].

Por outro lado, para lidar com problemas de otimização envolvendo funções-objetivo variando com o tempo (ou seja, não-estacionárias), o propósito de se empregar AEs passa a não ser mais o de encontrar uma única solução ótima (estática) mas sim o de ser capaz de *rastrear* (*track*) eficientemente e continuamente o ponto-ótimo ao longo de sua execução. Portanto, trabalhos nessa direção passam a investigar como os AEs podem ser empregados, com êxito, para [60] (i) encontrarem soluções flexíveis, que não sejam simplesmente adequadas, mas que possam ser facilmente readaptadas (ou customizadas) quando se fizer necessário (p. ex., devido à mudança dinâmica das condições do problema); (ii) *reusarem* eficientemente o conhecimento adquirido no passado (em outras execuções ou em outros problemas similares); e (iii) encontrarem um compromisso (*tradeoff*) apropriado entre o custo relativo à qualidade da solução e o custo inerente à necessidade de adaptação freqüente. Nesse cenário, destacam-se os métodos que introduzem recursos de *memória* em AEs com o intuito de preservação de soluções (parciais) de boa qualidade. Tais recursos de memória podem ser de gênero *implícito* (p. ex., via esquemas de poliploidia, ou mesmo via emprego dos AGs estruturados [106]) ou *explícito*, empregando um *buffer* adicional com regras explícitas de armazenagem e recuperação de informação.

Na seqüência, maior destaque será dado a trabalhos nas linhas de algoritmos coevolutivos e AEs híbridos.

6.6 Algoritmos evolutivos híbridos

Não obstante os AEs apresentarem um arquétipo conceitual bem genérico, passível de ser facilmente estendido, sua eficácia pode ficar eventualmente comprometida exatamente por esse grande poder de flexibilidade. É notória a existência de situações onde o conhecimento acerca do domínio do problema se torna de tamanha valia (p. ex., para ajudar na geração eficiente de soluções de alta qualidade) que seria, no mínimo,

⁽⁹⁾Segundo a terminologia da área, tais “compromissos” são chamados de soluções *não-dominadas* e formam o *conjunto ótimo de Pareto* para o problema. O gráfico formado pelos vetores pertencentes ao conjunto ótimo é chamado de *fronteira de Pareto*.

ilógico não embutir previamente tal conhecimento quando do projeto do algoritmo de busca. Por outro lado, ao tornar um AE mais “especializado” em um certo domínio, pode-se incorrer, como efeito colateral, em perda de generalidade, ficando a sua aplicabilidade circunscrita a apenas algumas instâncias de problemas específicos. Em verdade, esse dilema entre eficácia e generalidade constitui um grande desafio para qualquer projetista de métodos de busca e/ou otimização (e, por conseguinte, de resolução de problemas), em congruência com os resultados teóricos sumarizados pelos teoremas NFL acima mencionados.

Em geral, existem diversas formas de se “hibridizar” um AE. Michalewicz [30] sugere, pelo menos, duas classes complementares de AEs híbridos, quais sejam⁽¹⁰⁾, (1) aquela composta por abordagens que estendem o paradigma evolucionário pela *incorporação* de características adicionais (busca local, heurísticas gulosas, representação e operadores de variação específicos para o problema) ao *framework* padrão; e (2) aquela em que uma ou mais instâncias de AEs atuam como componentes separados de um sistema maior. Exemplos da segunda classe incluem as abordagens *genofuzzy* e *neuroevolutivas*, introduzidas no Cap. 1. Alguns dos projetos relatados nos Caps. 7 e 8 desta tese envolvem o uso congregado de ambas as classes de AEHs, na medida em que adotam AGs estendidos para a definição automática da estrutura e configuração de sistemas modulares (classe (2)), ao mesmo tempo que os próprios algoritmos passam a ser “refinados” mediante o emprego de métodos de otimização orientados para aprendizado local (classe (1)). A discussão que se segue nesta seção acena para a primeira classe de AEHs, sendo direcionada à hibridização de AGs.

A Fig. 6.2, retirada de Bäck *et al.* [30, Cap. D3], traz três possíveis pontos (*hotspots*) de inserção de conhecimento na estrutura de execução de um AG simples. Como dito anteriormente, com relação à população inicial, esta pode vir a ser configurada automaticamente via execução iterativa de uma ou mais heurísticas (via de regra, gulosas) vinculadas ao problema-alvo; espera-se, com isso, a geração de soluções iniciais que identifiquem diferentes regiões promissoras do espaço de busca, promovendo, em princípio, qualidade na diversidade. Outro uso de heurísticas está no restabelecimento da factibilidade das soluções geradas pelos operadores de variação⁽¹¹⁾, considerando, sobretudo, problemas de otimização com restrições — vide os vários métodos analisados por Michalewicz [320]. Por fim, cabem aqui também as estratégias que integram AGs com outros métodos/heurísticas de otimização (como descida do gradiente, subida da colina, programação dinâmica, *Branch-and-Bound*, programação linear/não-linear, programação inteira, etc.) ou com outros tipos de meta-heurísticas (como Busca Tabu ou Recozimento Simulado [109, 467]) para a confecção de operadores genéticos mais customizados. Exemplos de tais estratégias encerram os Algoritmos Meméticos e as abordagens que seguem as teorias evolucionárias de Lamarck [274] e Baldwin [35].

Os Algoritmos Meméticos (AMs) [339, 340] caracterizam-se por adotar, como princípio de projeto, uma população de entidades com perfil de agente exibindo capacidade de aprendizado e transferência de herança *cultural* mediante o emprego de *memes*, unidades de informação apreendidas e compartilhadas ao longo da vida dos agentes. A diferença básica entre AMs e AGs convencionais advém da inclusão de uma etapa de otimização aos agentes pela adição de operadores de busca local *independentes* (tipicamente, heurísticas

⁽¹⁰⁾Na terminologia de SIHs [180], as classes (1) e (2) de AEHs simbolizam, respectivamente, metodologias de *fusão* e *combinação* de subsistemas.

⁽¹¹⁾Soluções não-factíveis são, às vezes, chamadas de *letais* [45].

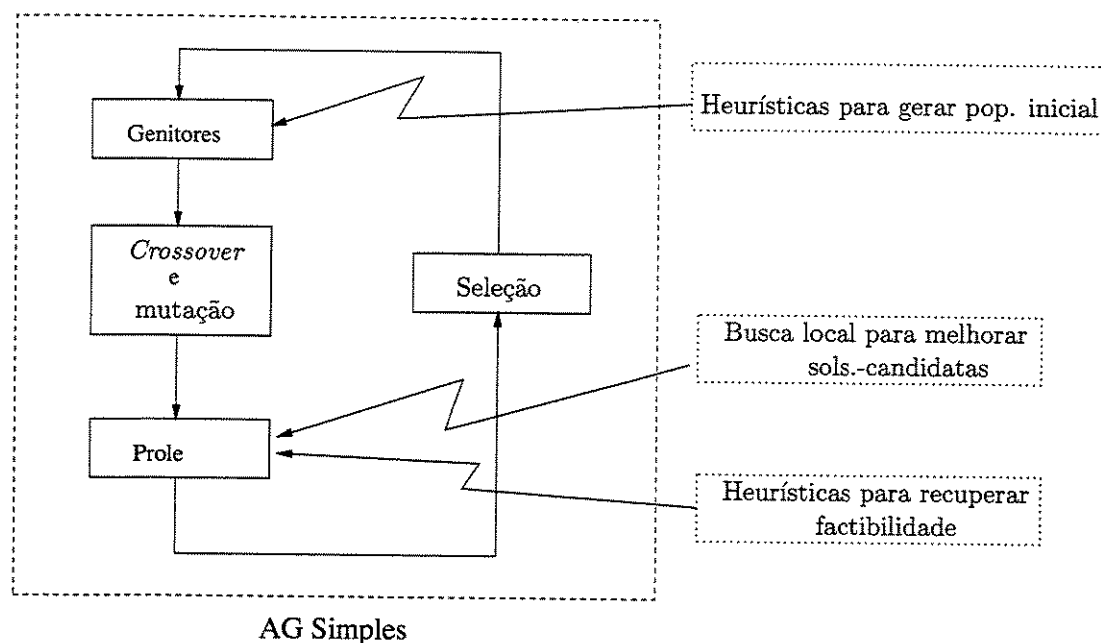


Fig. 6.2: Pontos de hibridização em um AG simples.

distintas customizadas ao problema-alvo), particularizando o “aprendizado” de cada indivíduo. Assim, as regras contidas nesses operadores locais (inseridos, quase sempre, em operadores de recombinação) *não* são transmitidas pelos genótipos dos indivíduos, ao passo que a informação cultural (memes), por não estar atada ao mecanismo de herança genética, pode ser disseminada de um único indivíduo a toda a população no decorrer de uma única geração⁽¹²⁾.

Enquanto os AMs se baseiam no fenômeno da transferência de informação sócio-cultural entre gerações de indivíduos, para os AGs híbridos que seguem a doutrina do Lamarckismo, o ponto-chave está na adoção da “herança dos caracteres adquiridos” como um *mecanismo* evolutivo *adicional* a ser “cominado” aos indivíduos da população. O mote central por trás da teoria de J.-B. Lamarck (quando contrastada com a de Darwin e Wallace) é que o processo evolutivo se daria mediante indivíduos se adaptando ao seu ambiente ao longo de seu tempo de vida (ou seja, modificando o seu fenótipo) e repassando tais adaptações adiante para a prole (já incorporadas no genótipo desta). Essa teoria foi demonstrada ser inconsistente, mais tarde, quando um estudo mais aprofundado, realizado por Weissmann, constatou que somente células germinativas (e não as somáticas) teriam capacidade de transmitir informação (via genes) para a prole. Mesmo desacreditadas no âmbito das ciências naturais, tais idéias passaram a ser implementadas, mais recentemente, em ambientes artificiais, dando origem a uma série de experimentações com propósitos diversos: desde para fins específicos de otimização (AGs híbridos) [48, 195, 400, 481] até para propósitos de simulação de vida artificial [7].

Em se tratando de AGs híbridos, a busca local lamarckiana emula um processo de “aprendizado”

⁽¹²⁾ Isso justifica o fato de AMs tenderem a ser mais eficientes, em termos computacionais, que AEs canônicos, quando aplicados sobre uma mesma classe de problemas [170].

(via método de otimização ou heurística relacionada ao problema) que ocorre em cada novo indivíduo da prole antes deste ter seu grau de aptidão atribuído (assim como em AMs). Esse tipo de aprendizado ocorre no espaço fenotípico e é mapeado de volta para o espaço genotípico⁽¹³⁾, ficando o resultado passível de ser transmitido, subsequente, via reprodução — em contraposição aos AMs, em que os memes não são propagados pelo genótipo. Quando o mesmo princípio é aplicado ao treinamento de RNAs [47, 257, 258], por exemplo, os pesos ou outros parâmetros da rede atribuídos inicialmente pelo processo evolutivo passam a ser substituídos por aqueles “aperfeiçoados” pelo algoritmo de aprendizado local empregado, sendo este tipicamente representado por métodos de otimização baseados em informação de primeira ou segunda ordem do gradiente da função-custo.

Quase um século após as primeiras postulações de Lamarck, James Baldwin [35] fornecia uma perspectiva alternativa e inovadora para a questão de como se dariam as influências das mudanças resultantes do processo de aprendizado dos indivíduos em seus cursos de vida (ontogenia) sobre as transformações evidenciadas pela evolução da espécie correspondente (filogenia). A conclusão a que Baldwin chegou foi a de que o aprendizado *individual* facilita a *assimilação indireta* de novas informações genéticas ao longo de gerações, *direcionando* o processo evolutivo mediante a atribuição de crédito (*credit assignment*) parcial a soluções genéticas também *parciais* (que serão refinadas e acumuladas, passo a passo, para formação de estruturas mais complexas). De um outro modo: a *plasticidade fenotípica* dos indivíduos, ao orientar as populações para níveis de adaptação ótimos, passa a selecionar quais das variações determinadas pelos genótipos serão mantidas nas gerações subsequentes; posteriormente, numa escala de tempo evolucionária, os traços aprendidos tornar-se-ão instintivos por algum mecanismo de assimilação genética. Conhecida atualmente como “Efeito Baldwin” (EB), essa teoria alternativa procura escapar tanto da crença na herança direta dos caracteres adquiridos, conforme idealizava Lamarck, quanto da fé dogmática neodarwinista que recusa atribuir um papel significativo na evolução das espécies à plasticidade comportamental dos indivíduos ao longo de suas trajetórias de vida.

O primeiro modelo quantitativo a demonstrar as potencialidades do Efeito Baldwin foi construído e explorado somente mais recentemente por Hinton e Nolan [48, 204], os quais, mediante simulação computacional via AGs, passaram a averiguar os efeitos do aprendizado individual sobre a evolução arquitetural de uma população de redes neurais. Na ocasião, a codificação genética empregada especificava uma topologia de RNA pela indicação de quais conexões, dentre 20 potenciais, seriam usadas para ligar um conjunto de neurônios. Para isso, lançou-se mão de um alfabeto ternário, com possibilidade de habilitar (‘1’), desabilitar (‘0’), ou deixar não-especificada (‘?’) cada uma das 20 conexões. No mais, uma função de *fitness* especial foi concebida indicando um alto valor de aptidão para a rede “correta” e um único valor de *fitness* mais baixo para todas as outras redes não-ótimas (i. e., a superfície de *fitness* apresentava-se como um largo platô com um único grande pico). Como resultado dos experimentos, os autores constataram que “[o aprendizado] altera o formato do espaço de busca sobre o qual a evolução opera e, desse modo, municia caminhos evolucionários adequados em direção a alelos coadaptados”. O segundo aspecto por trás do EB, o da assimilação

⁽¹³⁾ A aplicabilidade desses algoritmos fica, portanto, circunscrita a somente aqueles problemas para os quais esse mapeamento reverso seja possível.

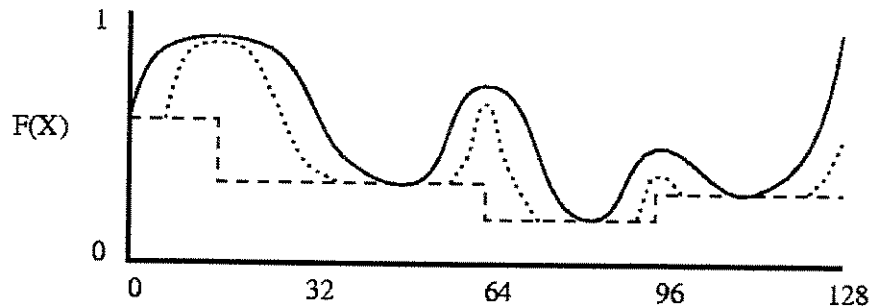


Fig. 6.3: O efeito da busca local (aprendizado) sobre uma superfície de *fitness* unidimensional. Legenda: linha cheia indica função de *fitness* original; linha pontilhada indica superfície de *fitness* após k passos da aplicação do algoritmo de aprendizado local; e linha tracejada refere-se à nova superfície de *fitness* depois de se “descer” até o ótimo local. (Fontes: Gruau & Whitley [185] e Whitley *et al.* [481].)

genética, foi também manifestado nos experimentos pela transformação (mutação) progressiva dos alelos mais flexíveis (?) em alelos com papel genético definido (fixo).

A Fig. 6.3, retirada de Whitley *et al.* [481], ilustra como o processo de aprendizado local pode realmente alterar o formato de uma superfície de *fitness* qualquer. O efeito conseguido após k passos de otimização é o de se alargar as “bases de atração” dos ótimos locais, tornando a função mais aplainada em seu entorno. Caso as buscas locais converjam para os pontos de mínimo local, então a nova superfície se tornará totalmente nivelada em cada base. Deve-se notar que cada uma dessas bases de atração pode atribuir um valor de *fitness* potencialmente diferente daquele relativo ao mínimo local original. A suavização da superfície de *fitness*, de acordo com esse processo, passa a impactar sobremaneira a qualidade da busca genética (global), uma vez que incrementa a probabilidade de se alocar mais pontos amostrais do espaço em regiões mais promissoras (haja vista suas bases de atração serem maiores). Ademais, como há maior facilidade de se saltar de uma base de atração para outras adjacentes, passa-se a combater, assim, o problema de convergência prematura para ótimos locais.

Apesar de métodos lamarckianos serem normalmente mais rápidos e efetivos que AGs simples [47, 185, 258, 400], Whitley *et al.* [481] constataram a existência de algumas funções em que ambas as estratégias promoveram convergência para ótimos locais, à medida que AGs explorando o Efeito Baldwin conseguiram convergir para ótimos globais. O fato é que, até o presente, não se conseguiu definir precisamente quais características devam ser apresentadas por uma dada classe de funções para que esta seja resolvida mais apropriadamente por métodos lamarckianos ou baldwinianos. Isso legitima, portanto, a necessidade de acareações empíricas sistemáticas entre tais métodos para cada possível domínio de aplicação.

6.7 Simbiose e coevolução: Do biológico ao artificial

Na natureza, vários mecanismos de realimentação (*feedback*) entre indivíduos sob pressão seletiva provêem forças diretivas intensas que acabam por contribuir com o aumento de complexidade no ecossistema [245]. Não somente os componentes abióticos exercem influência sobre que espécies estão presentes e em que abundância, mas também as próprias espécies são passíveis de serem modificadas, genotípica e fenotipicamente, como resultado de suas *interações* — também chamadas de relações ou coações — freqüentes com outras espécies. Na Ecologia, o termo *simbiose*, cunhado em 1879 por Anton de Bary e significando, originalmente, a “associação obrigatória entre dois seres vivos, em que ambos se beneficiam” [112], passou a assumir, mais recentemente, uma conotação mais ampla, designando “qualquer tipo de associação íntima entre seres vivos” [241], abarcando também aquelas em que uma das partes sofra algum tipo de prejuízo.

A Tab. 6.2 traz uma possível sistematização de alguns dos tipos de relações simbióticas mais comumente estudados e citados na literatura [15, 127, 241]. Um primeiro critério de classificação dessas interações ecológicas seria de “custo-benefício”, acarretando a divisão entre *relações harmônicas*, também chamadas de *interações positivas*, em que não há dano para nenhum dos agentes considerados e há vantagem para pelo menos um deles, e *relações desarmônicas* (ou *interações negativas*), em que pelo menos uma das partes sofre prejuízo⁽¹⁴⁾. Por outro lado, quando a interação estiver ocorrendo entre indivíduos de espécies distintas, fala-se que esta é de gênero *interespecífico* (ou *heterotípico*), ao passo que, quando os organismos envolvidos pertencerem à mesma espécie, diz-se que a coação é de caráter *intra-específico* (ou *homotípico*).

No contexto da CE, uma parcela crescente de pesquisadores vem se servindo de noções de simbiose para fins de simulação de comportamento coletivo emergente em ecossistemas virtuais (p. ex., dentro do tema de robótica evolutiva [85, 348]) ou para a confecção de técnicas de resolução de problemas mais robustas e flexíveis (foco da discussão desta seção). Com efeito, Daida e colaboradores [101, 102] traçam uma ponte bem interessante entre os mundos biológico e artificial ao proporcionar uma análise contemplativa de várias abordagens proeminentes, fazendo menção àquelas investigando o papel exercido pelas relações simbióticas sobre o processo evolutivo como um todo. Dentre as linhas de atuação que se enquadram nesse contexto, passa-se a seguir a caracterizar aquelas explorando aspectos coevolucionários. Antes, porém, é oportuno discorrer um pouco sobre a idéia por trás do conceito de coevolução.

Dizem alguns que o termo *coevolução* apareceu pela primeira vez na literatura no trabalho publicado por C. J. Mode num fascículo de 1958 do jornal *Evolution*, trabalho este intitulado “*A Mathematical Model for the Co-evolution of Obligate Parasites and their Hosts*”. O termo passou a ser mais bem explorado, refinado e difundido, posteriormente, por Ehrlich, Gilbert e Raven [135, 176], com o propósito de capturar as inúmeras influências evolucionárias que plantas e insetos (sobretudo borboletas) exercem uns sobre os outros⁽¹⁵⁾. Em verdade, como acontece com quase todos os conceitos recém-adotados que descrevem fenômenos

⁽¹⁴⁾ Alguns autores empregam o termo *neutralismo* para indicar a ausência de qualquer influência de um organismo sobre outro.

⁽¹⁵⁾ Cogita-se, por exemplo, que a capacidade de voar de alguns insetos surgiu como resposta direta ao processo seletivo ocorrido sobre formações arbóreas, que tiveram que elevar o seu porte físico (caule e copa) para terem

TIPOS DE RELAÇÕES ECOLÓGICAS			
Relações harmônicas	Interespecíficas	Mutualismo	Associação <i>necessária</i> à sobrevivência de duas ou mais espécies em que todas se beneficiam
		Protocooperação	Relação <i>não-obrigatória</i> à sobrevivência, porém em que todas se beneficiam
		Comensalismo	Associação em que uma das espécies se beneficia, alimentando-se dos restos alimentares de uma outra, que não é prejudicada
		Inquilinismo	Associação em que uma das espécies se fixa ou se abriga em outra, porém sem prejudicá-la
	Intra-específicas	Colônia	Relação entre indivíduos da mesma espécie que se mantêm ligados entre si, formando uma unidade estrutural
		Sociedade	Associação entre indivíduos da mesma espécie, não ligados anatomicamente, que se agrupam por divisão de trabalho e se organizam de modo cooperativo
Relações desarmônicas	Interespecíficas	Competição interespecífica	Relação entre indivíduos de espécies diferentes que concorrem pelos mesmos fatores do ambiente, existentes em quantidade limitada
		Parasitismo	Associação em que uma das espécies, tipicamente a menor, vive sobre ou dentro da outra, alimentando-se dela, porém, geralmente sem matá-la
		Predatismo	Relação em que uma das espécies, a predadora, mata a outra para dela se alimentar
		Amensalismo	Relação em que uma das espécies <i>inibe</i> o crescimento ou reprodução da outra
		Esclavagismo	Associação em que uma das espécies se aproveita das atividades ou do trabalho da outra
	Intra-específicas	Competição intra-específica	Relação entre indivíduos da mesma espécie que concorrem pelos mesmos fatores do ambiente, existentes em quantidade limitada

Tab. 6.2: Classificação das interações simbióticas de espécies biológicas. (Fonte: Silva Jr. e Sasson [241].)

biológicos, a noção de coevolução não é tão recente assim. O próprio Darwin já havia feito referência à idéia de "... coadaptações de seres orgânicos uns aos outros ..." em sua obra seminal de 1859 [105]. Atualmente, o preceito de coevolução passou a assumir uma perspectiva mais abrangente, incluindo todos os processos simbióticos (positivos ou negativos) entre *populações* separadas de indivíduos (espécies ou mesmo outros grupos taxonômicos de mais alto nível) que tenham suas superfícies de *fitness* e trajetórias evolucionárias de alguma maneira acopladas umas às outras, em consonância com uma dinâmica tal que mudanças na composição genética de um grupo *tenderão*, fortemente, a ocasionar o surgimento de variações em outros grupos [166, 245, 449]. Dessa exposição, fica evidente que o estudo de fenômenos coevolucionários se encontra na interseção das duas áreas, Evolução e Ecologia, na medida em que encara as relações simbióticas como instrumento inseparável (e, em parte, até responsável) do próprio processo evolucionário⁽¹⁶⁾. De fato, enquanto a definição de simbiose envolve qualquer associação (mesmo aquelas não tão persistentes temporalmente) entre diferentes *indivíduos* vivos, o escopo coevolutivo é mais restrito, envolvendo somente as interações simbióticas perenes e recíprocas entre *grupos* (espécies) desses indivíduos. Nas palavras de Morrison, "na presença de simbiose, a seleção natural produz a coevolução" [337].

Assim como para o conceito de simbiose, não há um consenso geral quanto à melhor aceção do termo "coevolução", sendo este, muitas vezes, empregado de forma incorreta; por exemplo, como sinônimo puro e simples de "coadaptação" e/ou "interação"⁽¹⁷⁾. A Tab. 6.3 traz uma lista de definições proferidas por militantes de renome na área, algumas mais específicas, outras menos rígidas. Janzen [230] foi um dos primeiros a levantar critérios para a delimitação da abrangência do tema, apontando para situações frequentes onde adaptações sequenciais e concorrentes apresentadas por espécies naturais que coabitam o mesmo espaço (biótopo), mas que não interagem no plano evolutivo, são interpretadas erroneamente como sendo de natureza coevolucionária. Ademais, como salientado por Van Valen [459], "nem tudo que parece ser coevolução está realmente coevoluindo, embora a distinção possa ser, freqüentemente, difícil de ser feita na prática". De todo modo, é fato que a "coevolução é um dos processos mais importantes influenciando os padrões de variação e adaptação do meio" [345], sendo comum se distinguir níveis de caracterização dos sistemas coevolucionários, estes diferenciados pela ênfase colocada na *especificidade* e *reciprocidade* das interações [449]: (a) *coevolução específica* (ou gene-a-gene), em que há coadaptação entre duas espécies, com correspondência exata entre partes de seus genótipos, como, por exemplo, a que ocorre entre parasitas e seus hospedeiros; (b) *coevolução difusa* (ou em guildas), que descreve mudanças evolutivas recíprocas entre grupos, e não entre pares, de espécies, sendo a especificidade mais ampla do que no caso anterior — uma determinada característica em uma ou mais espécies de uma guilda (associação mutualística) pode evoluir em resposta a uma característica

maiores chances de sobreviver (pela captura de mais energia solar e água das chuvas). Por outro lado, as estruturas reprodutivas das plantas (ou seja, as flores) teriam surgido como consequência da evolução de tais insetos voadores, aproveitando-os para fins de polinização [166, 176].

⁽¹⁶⁾ O termo *simbioticismo* é comumente empregado para representar o conjunto de teorias que suportam essa premissa [101]. Como exemplo, consoante a hipótese conhecida como *endosimbiose* [67, 307], cogita-se que organismos eucarióticos são fruto de associações mutualísticas entre procariontes e algumas células infectadas por eles.

⁽¹⁷⁾ A mesma observação procede quando se leva em consideração algumas abordagens computacionais que se dizem coevolucionárias, mas que, em essência, não apresentam o elemento-chave de acoplamento de *fitness* entre as entidades em evolução para serem consideradas como tal. Confundir coevolução com simples paralelismo entre populações é um bom exemplo disso.

Tab. 6.3: Algumas definições para o termo “coevolução” ordenadas cronologicamente. (Fontes: Janzen [230], Nitecki [345] e Thompson [449].)

Fonte	Definição
(Van Valen 1973)	“Coevolução ocorre quando as interações diretas e indiretas de duas ou mais <i>unidades em evolução</i> produzem uma resposta evolucionária em cada uma delas”
(Smith 1975)	“As mudanças em duas ou mais espécies que estão atuando como forças seletivas em cada uma delas”
(Ricklefs 1979)	“O desenvolvimento de traços geneticamente determinados em duas espécies para facilitar alguma interação, usualmente de benefício mútuo”
(Roughgarden 1979)	“A evolução simultânea de populações interagentes”
(Janzen 1980)	“Coevolução pode ser utilmente definida como uma mudança evolucionária em um traço dos indivíduos de uma população em resposta a um traço dos indivíduos de uma segunda população, seguida por uma resposta evolucionária da segunda população às mudanças da primeira”
(Kiester <i>et al.</i> 1983)	“Os efeitos seletivos conjuntos sobre caracteres de grupos taxonômicos interagentes, com base na variação herdável desses caracteres”
(Slatkin 1983)	“A evolução de duas ou mais espécies como consequência das suas interações umas ou com as outras”
(Thompson 1994)	“Qualquer mudança evolucionária induzida reciprocamente entre duas ou mais espécies ou populações interagentes”

ou conjunto delas em várias outras espécies de uma outra guilda; e (c) *coevolução escape-e-radiação*, em que uma certa espécie de predador passa a interagir com presas não-similares em épocas distintas, alternando entre elas com o tempo, como consequência direta da diversificação genética (e divergência) apresentada pelas presas.

No âmbito computacional, sistemas coevolucionários artificiais têm sido, mais recentemente, objeto de intensa investigação, partindo-se do pressuposto de que a modelagem de possíveis interdependências entre os indivíduos (i. e., soluções do problema) em evolução incorrerá em ganhos compensatórios em termos de eficiência e eficácia. Nesses ambientes, o grau de aptidão de um certo indivíduo passa a depender diretamente dos valores de *fitness* de seus pares, quer sejam estes da mesma ou de outra espécie. Almeja-se, com isso, a introdução de maior *dinamicidade* na avaliação das soluções, na medida em que a superfície de *fitness* associada a uma certa espécie (população) passa a ser moldada no curso do próprio processo evolutivo [245]. Tal “evolução acoplada” provê algumas vantagens relevantes sobre AEs tradicionais, tornando a abordagem coevolucionária bastante apropriada para lidar com problemas computacionais de maior complexidade. Dentre tais vantagens, pode-se ressaltar que a coevolução artificial [358, 363]: (i) favorece a descoberta de soluções complexas sempre que soluções complexas são requeridas; (ii) estimula a manutenção de diversidade; (iii) favorece o papel de funções de *fitness* computacionalmente eficientes; e (iv) é naturalmente propícia para ser

implementada em ambientes paralelos.

Como exemplo de ambiente para computação coevolucionária, Morrison e Oppacher [337, 338] propuseram um *framework* conceitual genérico, chamado de “*Modelo Linear de Simbiose*”, em que as relações ecológicas sistematizadas na Tab. 6.2 passam a ser formalmente definidas como operadores binários atuando sobre os valores de *fitness* dos indivíduos de uma população comum de AG. Um conjunto de tais relações binárias vem a formar uma teia (grafo) conectando, de alguma forma, os indivíduos uns aos outros, de modo que a influência que um dado organismo *A* exerce em um outro *B* no tempo *t* poderá repercutir, em uma geração posterior $t + p, p \geq 1$, sobre o *fitness* relativo apresentado por um terceiro ente *C*, mesmo que entre *A* e *C* não haja conexão direta no grafo. A despeito de haver uma única população em evolução (ou seja, espécie), tal modelo pode ser considerado como pertencente ao gênero coevolucionário pelo simples fato de haver alto acoplamento entre os valores de *fitness* dos indivíduos, como fruto direto de suas relações simbióticas. Morrison e Oppacher confrontaram o desempenho de diferentes instâncias de seu modelo simbiótico ante implementações de AGs simples, tendo como referência um conjunto-*benchmarking* de funções multimodais de otimização numérica. Os resultados comprovaram a tese inicial de que a modelagem de influências evolutivas entre indivíduos pode sim catapultar o desempenho de AEs tradicionais em ambientes mais complexos.

Afora esse *framework* genérico de Morrison e Oppacher, a maioria dos sistemas coevolucionários artificiais pode ser categorizada em duas frentes de atuação complementares, referidas como *Coevolução Competitiva* e *Coevolução Cooperativa*, modelando interações negativas e positivas, respectivamente, de indivíduos da mesma espécie ou de espécies distintas. Um primeiro esforço de unificação dessas duas vertentes sob um mesmo plano conceitual foi empreendido por Paredis [358], embora o arcabouço proposto não seja de todo universal — não contempla, por exemplo, as FFCs discutidas na seqüência.

6.7.1 Coevolução artificial competitiva

As abordagens competitivas seguem a lógica de que o sucesso de um lado deva ser sentido pelo outro como um risco iminente que precisa ser respondido à altura e a tempo a fim de se evitar a sua possível extinção [19, 203]. Idealmente, espécies artificiais em competição tentarão superar umas às outras continuamente (p. ex., como acontece na natureza, tornando-se mais rápidas, maiores ou mais letais), levando ao surgimento de verdadeiras “corridas armamentistas” (*arms races*) [111] ilimitadas (*open-ended*) entre elas. Esse processo de coadaptação recorrente *pode* resultar em um incremento progressivo na sofisticação das estratégias adotadas tanto por “predadores” (p. ex., restrições num problema) como por suas “presas” (soluções). Isso, entretanto, nem sempre é afofável. A teoria da “Hipótese da Rainha Vermelha”, proposta por Van Valen [458, 459] em alusão a uma passagem da fábula “*Alice no país dos espelhos*”⁽¹⁸⁾, de Lewis Carroll, estabelece o contraponto de que, muitas vezes, *não* há “ganho” evolutivo algum ocorrendo em

⁽¹⁸⁾Nessa passagem, Alice foge do exército (de cartas de baralho) da Rainha Vermelha, mas não consegue se distanciar de seus perseguidores. Nesse momento, é advertida pela Rainha Vermelha: “Aqui, veja, você precisa correr o máximo possível, para se manter no mesmo lugar”. Alice só seria pega se parasse de correr.

espécies participantes de uma escalada armamentista, pois elas estariam incrementando seus “arsenais” (p. ex., novas habilidades comportamentais ou fisiológicas) apenas para se *manterem* no mesmo ponto (estado de estabilidade) de suas trajetórias evolutivas. Mais recentemente, esse tipo de fenômeno “contraproducente” passou a ser também observado em experimentos envolvendo implementações computacionais coevolutivas em ambientes de aprendizado [147] ou comportamento emergentes [84, 360].

No geral, abordagens coevolutivas competitivas podem ser agrupadas em dois segmentos alternativos, o primeiro concernente ao aprendizado contínuo de novas habilidades (estratégias) por intermédio de torneios competitivos entre indivíduos da mesma população (baseando-se, pois, em conceitos de Teoria dos Jogos — ver Cap. 5) e o segundo compreendendo abordagens multipopulacionais. O modelo precursor de Hillis [203] se encaixa nessa segunda linha, sendo, às vezes, referenciado como “coevolução parasita-hospedeiro”, já que estimula a especificidade das interações dos genótipos dos agentes em coadaptação. Aqui, duas populações, uma de soluções para um certo problema (no caso, de ordenação) e outra de instâncias de testes de avaliação, passam a ter suas superfícies de *fitness* entrelaçadas de tal forma que um escore positivo para um lado represente um escore negativo para o outro (ou seja, são complementares). Esse trabalho foi estendido posteriormente por Paredis, que introduziu o conceito de “*life-time fitness evaluation*” [358] para a retenção de soluções de boa qualidade ao longo da execução, aplicando tal variação quando da resolução de problemas não-triviais de satisfação de restrições [357]. Tanto Hillis como Paredis constataram que, ao longo das execuções de seus experimentos, a manutenção separada de uma população de testes (parasitas) promoveu o surgimento gradual de soluções cada vez mais eficazes, que tiveram que lidar com nuances e obstáculos (impostas pelos testes adaptativos) cada vez mais peculiares e sofisticadas. Comprovava-se, daí, o notável potencial da abordagem coevolutiva em evitar dinamicamente o problema de convergência prematura típico de AEs tradicionais.

Por outro lado, o artigo de Angeline e Pollack, intitulado “*Competitive Environments Evolve Better Solutions for Complex Tasks*” [19], é representativo do outro segmento de trabalhos supracitado, de sorte que faz uso da concepção de *funções de fitness competitivas* (FFCs) para o projeto automático de especialistas em jogos de soma zero (no caso, “jogo da velha”). Em vez de se aferir os graus de aptidão dos indivíduos via competição em “*round-robin*”, os autores adotaram o instrumento de torneios de eliminação simples como forma adaptativa de se “relativizar” e ordenar tais valores. Uma vantagem desse método é que somente duas estratégias são aferidas em cada torneio, eliminando a necessidade, por parte do projetista, de elaborar uma função de *fitness* “perfeita” para se discriminar eficientemente entre estratégias de jogo adequadas ou não-adequadas. Aqui, todos os membros da população que perderam em um nível particular do torneio são considerados como equiparados em sua aptidão, proporcionando, daí, uma ordenação simples que ajuda a evitar o problema de convergência prematura. Além do fato de esse método coevolucionário conseguir gerar estratégias mais eficazes (em termos do número de partidas ganhas) que aquelas produzidas por meio de uma função de *fitness* estática preconcebida por um especialista humano no domínio de estudo, outra vantagem marcante pôde ser apreendida por Angeline e Pollack, qual seja a de que a coevolução artificial via FFCs não requer nenhum conhecimento particular *a priori* acerca do domínio do problema em questão.

Outras empreitadas passaram a estender os resultados teórico-experimentais obtidos por Angeline & Pollack, Hillis e Paredis, de modo a evitar os problemas de estagnação do processo de corrida armamentista levantados acima (tais como o surgimento de estados “estáveis e medíocres” [147]). Dentre esses trabalhos destacam-se os de Darwen e Yao [104] e de Rosin e Belew [399], ambos lançando mão dos conceitos de *niching* e de formação dinâmica de subpopulações como mecanismos eficientes de preservação de diversidade no ecossistema e de geração de estratégias de jogo com maior poder de generalização (i. e., com capacidade de derrotar diferentes “estilos” de jogadores). Tais abordagens podem ser consideradas, portanto, como um misto de abordagens competitivas multipopulacionais e de aprendizado por FFCs.

6.7.2 Coevolução artificial cooperativa

Com relação às abordagens coevolutivas cooperativas, estas, na sua maioria, são de natureza multipopulacional e inclinadas a seguir mais à risca a filosofia do “dividir-para-conquistar”. Isso porque se fundamentam, de antemão, na decomposição de problemas complexos em subtarefas de menor granulosidade, a serem tratadas por módulos adaptativos cujas interrelações harmônicas passarão a guiar o curso do próprio processo evolutivo. Exemplos de propostas que seguem essa perspectiva mais “mutualística” são descritos na Tab. 6.4, dentre os quais se destaca o “modelo de referência” concebido por Potter e De Jong [374, 375], que serviu de fonte de inspiração para alguns dos projetos desta tese, sendo alvo das próximas considerações.

Tab. 6.4: Diferentes trabalhos no segmento de Coevolução Cooperativa ordenados cronologicamente.

Fonte	Descrição
Husbands & Mill [216, 217]	Aparentemente, constituem os primeiros trabalhos a empregar noções de Coevolução Cooperativa, sendo voltados para problemas de escalonamento de processos manufatureiros. Aqui, cada população fica responsável por evoluir um plano de escalonamento (i. e., seqüência de operações e recursos a serem alocados) de um segmento específico do chão de fábrica. A cada simulação, vários planos (cada um vindo de uma população) são agregados e o conjunto deles é avaliado sobre o problema. Existe uma população especial à parte, composta de <i>árbitros</i> , estes destinados a resolver conflitos de interesse entre os planos (p. ex., uso simultâneo dos mesmos recursos físicos). O valor de <i>fitness</i> dos árbitros depende de quão bem o grupo de planos por ele coordenado trabalhe junto, vindo, daí, a noção de cooperação — mesmo que os valores de <i>fitness</i> dos planos de escalonamento sejam atribuídos de maneira independente da dos demais.

Continua na próxima página

Tab. 6.4: Diferentes trabalhos em Coevolução Cooperativa. (cont.)

Potter & De Jong [238, 375, 376, 377]	Modelo arquitetural padronizado que, ao contrário do caso anterior, estabelece, de antemão, que o grau de aptidão de cada subcomponente (i. e., solução parcial do problema) dependa diretamente do desempenho dos demais. Aqui, “a união faz a força” (evolutiva).
Paredis [358, 359]	Esquema alternativo em que duas populações, uma de <i>representações de soluções</i> e outra de soluções propriamente ditas, passam a interagir cooperativamente com o intuito de se abordar automaticamente um dado problema — aliviando do projetista, inclusive, a carga de ter que especificar <i>a priori</i> o formato de codificação da solução. Não há, porém, estímulo explícito algum para a decomposição da solução em módulos.
Puppala <i>et al.</i> [380]	Variação do modelo de Potter e De Jong que introduz uma memória compartilhada para se armazenar, explicitamente, as melhores soluções completas (ou seja, combinações dos subcomponentes) encontradas ao longo das execuções. Embora custosa computacionalmente, pois restringe o número total de espécies e de soluções no <i>buffer</i> , trata-se de uma proposta interessante, pois tem propensão a gerar indivíduos com papéis mais genéricos no ecossistema, haja vista que estes passam a ser avaliados de acordo com a sua capacidade de cooperar bem em diferentes combinações (de alta qualidade) existentes na memória.
Wiegand [482]	Ao que parece, trata-se da primeira abordagem na literatura a implementar um modelo de coevolução difusa, estendendo a arquitetura de Potter e De Jong para esse fim.
Zhao <i>et al.</i> [511, 512]	Os autores especificam um <i>framework</i> multiresolucional e formal com o intuito de abarcar todas as possíveis variações de modelos coevolutivos cooperativos. Como resultado, tal <i>modelo de sociedade</i> , como é chamado, peca por ser altamente complexo e de difícil implementação, devendo ser encarado mais como um substrato conceitual para se discutir e analisar aspectos particulares — p. ex., influência das várias populações aninhadas umas sobre as outras.

Continua na próxima página

Tab. 6.4: Diferentes trabalhos em Coevolução Cooperativa. (cont.)

Wiegand <i>et al.</i> [484, 486]	Trata-se de um estudo sistemático e experimental, conduzido sobre a arquitetura coevolutiva de Potter e De Jong, que mediu e contrastou o desempenho de diferentes esquemas de parceria, tendo em vista uma suíte de problemas de otimização numérica multimodal. Na ocasião, foram levantados alguns atributos interessantes para a caracterização desses esquemas de parceria, tais como (a) o <i>grau de gulodice</i> (<i>greediness</i>) na escolha dos parceiros; (b) o <i>número de possíveis parceiros</i> ; e (c) o método de atribuição de <i>fitness</i> (<i>credit assignment</i>), dado que um conjunto de resultados de desempenho advindos de diferentes parcerias esteja disponível. Foi avaliado também o possível impacto que representações/decomposições inadequadas possam vir a exercer sobre a escolha dos parceiros para cooperação.
Aickelin & Bull [10, 11]	É proposto um modelo coevolutivo hierárquico (também de gênero multiresolucional) em formato de pirâmide, vindo daí o nome de <i>coevolução piramidal</i> . Nessa arquitetura, à medida que se passa de um nível para outro acima, diminui-se a resolução do problema, aumentando-se, por conseguinte, o espaço de busca. Subpopulações em diferentes patamares têm que cooperar entre si de modo a que se consiga encontrar soluções completas adequadas. Daí, os autores também investigam o efeito exercido por diferentes esquemas de seleção de parceiros sobre o desempenho final da arquitetura, levando em conta problemas de otimização combinatorial de múltipla escolha — p. ex., montagem de turnos de trabalho para a enfermaria de hospitais.
Iorio e Li [220]	Esse trabalho investiga os possíveis efeitos positivos carreados com o emprego de parâmetros de controle auto-adaptativos (tais como o tamanho da população e taxas de operadores de variação) para cada uma das espécies em coevolução no modelo de Potter e De Jong.
Keeratitumrong <i>et al.</i> [248]	Aparentemente, constitui a primeira tentativa de se empregar uma abordagem coevolutiva cooperativa para a resolução de problemas de otimização multiobjetivo. Para tanto, a arquitetura coevolutiva de Potter e De Jong passou a servir como algoritmo de busca para o método MOGA [116] (ver Seção 6.5), dando origem à variante MOCCGA (de <i>Multi-Objective Cooperative Coevolutionary Genetic Algorithm</i>).

Continua na próxima página

Tab. 6.4: Diferentes trabalhos em Coevolução Cooperativa. (cont.)

Sofge <i>et al.</i> [438]	Essa abordagem lança mão tanto da arquitetura original de Potter e De Jong como do modelo de AGs distribuídos. Aqui, cada espécie em coevolução é associada a um <i>conjunto</i> de subpopulações (subespécies) executando em paralelo, sendo que indivíduos de subpopulações distintas podem ser intercambiados periodicamente, ao longo da execução, via migração. Na fase de avaliação, tais subpopulações passam a ser combinadas numa só. Em vez da manutenção de diversidade no nível das subpopulações, a idéia por trás do mecanismo de migração é a de promover uma maior mistura entre os componentes alocados às subespécies nas últimas gerações do processo evolutivo, facilitando o desemaranhar das possíveis interações epistáticas entre os componentes da solução.
Wiegand <i>et al.</i> [485, 487]	Constitui-se numa primeira tentativa de se analisar mais formalmente a dinâmica por trás dos algoritmos coevolutivos cooperativos, com o intuito de caracterizar as suas propriedades e de elucidar para quais domínios de aplicação tais algoritmos seriam mais indicados. Para tanto, os autores fizeram uso do ferramental analítico propiciado pela Teoria dos Jogos (equilíbrio de Nash, particularmente) e introduziram o conceito de <i>jogos simétricos multipopulacionais</i> para fins de modelagem das interações dinâmicas entre os componentes em coevolução.
Panait <i>et al.</i> [356]	Trata-se de um primeiro esforço em caracterizar uma abordagem alternativa à definição <i>a priori</i> do método de parceria para avaliação dos módulos. A idéia é a de introduzir certas “polarizações” no algoritmo coevolutivo cooperativo de Potter e De Jong de tal modo que ele encontre por si só a melhor forma de se estipular as parcerias ao longo de sua execução. Para tanto, o <i>fitness</i> de um certo indivíduo passa a ser visto como uma combinação linear convexa de dois termos, o primeiro dado pelo seu desempenho em um grupo escolhido de acordo com um esquema de parceria fixo (p. ex., composto pelos melhores candidatos das outras populações) e o segundo dado por uma <i>estimativa</i> de como seria o seu desempenho tivesse ele a chance de integrar a parceria <i>ideal</i> (i. e., composta por parceiros ideais). Os autores investigaram aqui a sensibilidade do algoritmo coevolutivo a diferentes escolhas do parâmetro δ , $0 \leq \delta \leq 1$, que controla o peso de cada um dos termos acima. A aplicabilidade dessa abordagem, porém, fica condicionada a somente aqueles domínios de aplicação em que se possa construir, para cada módulo, a sua parceria (time) ideal — a tarefa de otimização analisada correspondeu a uma instância de um problema artificial criado pelos próprios autores.

No cenário vislumbrado por Potter e De Jong [374, 375], cada módulo (componente) adaptativo fica responsável por apenas uma *solução parcial* do problema e passa a ser associado a uma espécie (população

isolada) evoluindo (via um AE dedicado) no ecossistema⁽¹⁹⁾. O objetivo de cada indivíduo de uma dada espécie passa a ser duplo: ao mesmo tempo que deve otimizar uma parte específica da solução completa, ele deve *cooperar* com indivíduos (provenientes de outras espécies) responsáveis por outras partes, de tal maneira que o *time* como um todo (i. e., solução completa) seja bem sucedido. A hipótese por trás desse modelo mutualístico é que a estratégia de se empregar diversas buscas paralelas procurando por diferentes soluções parciais para um problema complexo é mais *eficiente* e *eficaz* que a estratégia comum de se adotar um único processo de busca geral à procura de uma única solução completa (quase-)ótima [335]. O Alg. 4 traz a estrutura básica do *framework* coevolutivo concebido por Potter e De Jong.

Algoritmo 4 Algoritmo Coevolutivo Cooperativo

Entrada: n_p , número de populações, Θ_e e $\{\mu, \lambda, \Theta_r, \Theta_m, \Theta_s\}_{1}^{n_p}$, parâmetros de controle para cada população.

Saída: $a^* = \{a_1^*, \dots, a_{n_p}^*\}$, melhor solução completa (i. e., grupo de soluções parciais) encontrada.

Computação:

$t \leftarrow 0$;

for $i = 1$ to n_p **do**

$P_i(t) \leftarrow \text{INICIAR\_POPULACAO}(\mu_i)$;

$R_i(t) \leftarrow \text{ESCOLHER\_REPRESENTANTES}(P_1(t), \dots, P_{i-1}(t), \dots, P_{i+1}(t), \dots, P_{n_p}(t))$;

$F_i(t) \leftarrow \text{AVALIAR\_GRUPO}(P_i(t), \mu_i, R_i(t))$;

end for

while $(e(\{P_i(t)\}_{i=1}^{n_p}, \Theta_e) == \text{falso})$ **do**

$t \leftarrow t + 1$;

for $i = 1$ to n_p **do**

$P'_i(t) \leftarrow \text{RECOMBINAR}(P_i(t-1), F_i(t-1), \Theta_r^i)$;

$P''_i(t) \leftarrow \text{ALTERAR}(P'_i(t), \Theta_m^i)$;

end for

for $i = 1$ to n_p **do**

$R_i(t) \leftarrow \text{ESCOLHER\_REPRESENTANTES}(P''_1(t), \dots, P''_{i-1}(t), \dots, P''_{i+1}(t), \dots, P''_{n_p}(t))$;

$F_i(t) \leftarrow \text{AVALIAR\_GRUPO}(P''_i(t), \lambda_i, R_i(t))$;

$P_i(t) \leftarrow \text{SUBSTITUIR}(P''_i(t), F_i(t), \mu_i, \lambda_i, \Theta_s^i)$;

end for

end while

Peça vital nesse modelo arquitetural é o mecanismo de atribuição de crédito (ver Subseção 5.2.3) a ser empregado para distinção entre módulos apropriados ou não. A Fig. 6.4 mostra um ecossistema fictício composto por n_p espécies e evidencia o modo como cada AE deve computar o grau de aptidão de seus indivíduos. Como exemplo de aplicação mais concreto [377], suponha que se esteja otimizando uma função f

⁽¹⁹⁾Veja que há a possibilidade de dois ou mais módulos adaptativos estarem associados a uma mesma espécie, como ocorre, por exemplo, com a abordagem SANE [334], analisada na Seção 6.9. Na discussão que segue, porém, presume-se que cada componente esteja associado a uma espécie distinta.

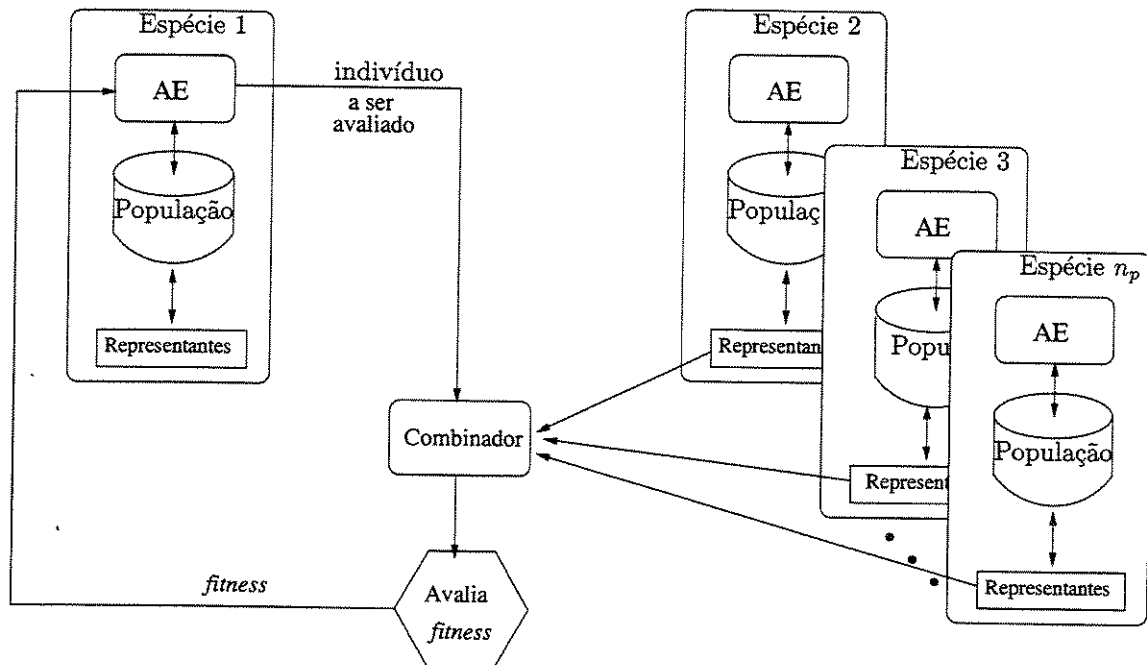


Fig. 6.4: Arquitetura coevolucionária cooperativa proposta por Potter e De Jong. O esboço mostra o processo evolutivo sob a ótica da espécie #1. O indivíduo (módulo) em avaliação é combinado com um ou mais representantes de outras espécies presentes no ecossistema de modo a se formar uma ou mais soluções completas a serem testadas no domínio de aplicação. O *fitness* desse indivíduo dependerá da qualidade dessas soluções. (Fontes: Peña-Reyes [363] e Potter [374].)

definida por n_p parâmetros, isto é, $f(x_1, \dots, x_{n_p})$. A decomposição mais trivial do problema é a de se associar cada argumento (módulo, aqui) a uma espécie distinta. Daí, cada uma das n_p populações segue um ciclo evolutivo separado. Contudo, no momento de se atribuir *fitness* a seus indivíduos, elas são “sincronizadas”, de tal modo que cada espécie avalie seus próprios indivíduos e também disponibilize representantes para a avaliação dos indivíduos das demais espécies. Esse processo de definição de quem interage com quem para fins de atribuição de *fitness* (capturado pela rotina “ESCOLHER_REPRESENTANTES” no Alg. 4) segue, via de regra, um esquema predefinido, chamado por alguns autores [10, 66] de *método de parceria* e por outros [375, 484] de *método de colaboração*⁽²⁰⁾. Um exemplo simples de método de parceria é o de escolher os melhores indivíduos atuais das demais populações como seus representantes. Outra possibilidade é a de escolher parceiros aleatórios e compor não um só mas vários times. Sendo o caso, o *fitness* do indivíduo será dado pelo supremo ou pela média dos índices de desempenho dos grupos dos quais participa sobre o problema-alvo.

Outro ponto relevante aqui diz respeito à etapa de decomposição da tarefa e concepção dos módulos. Na grande maioria das aplicações envolvendo a abordagem coevolutiva cooperativa de Potter e De Jong [119, 140, 334, 364, 376, 377, 380], a estrutura de divisão e recomposição dos módulos envolvidos no sistema coevolutivo passa a ser predefinida pelo projetista. Essa estratégia é atraente nos casos em que já se tem *a priori* uma visão clara de como se pode destrinchar o problema de uma maneira *eficiente*, que facilite o emprego e execução da arquitetura. Contudo, nem sempre esse é o caso, devendo-se, daí, recorrer a estratégias de particionamento *emergente* da tarefa em módulos. Mais recentemente, Potter e De Jong passaram também a considerar tal alternativa de projeto mais seriamente. Nas palavras dos autores, “a pressão seletiva para aumentar o *fitness* total do ecossistema pode prover o estímulo necessário para a emergência de um número apropriado de subcomponentes interdependentes que venham a cobrir múltiplos nichos, evoluir até um nível apropriado de generalidade, e se adaptar à medida que tanto o número como os papéis de seus pares mudem com o tempo” [375]. Nesse sentido, com o intuito de evitar possíveis estagnações do processo coevolutivo, os autores imaginaram a possibilidade de se introduzir, *dinamicamente*, novas espécies (com populações iniciadas aleatoriamente) no ecossistema, estipulando uma condição que deva ser satisfeita para esse fim [375]:

$$f(t) - f(t - L) < G,$$

onde $f(t)$ corresponde ao valor de aptidão da melhor parceria (solução completa) no tempo t , G é um limiar de comparação para avaliação do grau de melhoria entre $f(t)$ e $f(t - L)$, e L é uma constante especificando o tamanho de uma janela temporal em que se avalia se houve ou não estagnação no incremento do *fitness* relativo à melhor solução completa. Embora se utilizando de um artifício um pouco distinto para criação de novas espécies, a abordagem coevolutiva cooperativa apresentada no projeto da Seção 8.4 também pode ser considerada como de decomposição emergente.

De um modo geral, pode-se levantar um elenco de propriedades distintivas exibidas pelo *framework* coevolutivo proposto por Potter e De Jong [363, 375]:

⁽²⁰⁾Pelas razões colocadas na Subseção 5.2.1, adotou-se aqui a primeira denominação.

1. A evolução de espécies em populações isoladas se caracteriza como uma solução simples, mas bem eficaz, para o problema de preservação de diversidade no ecossistema, garantindo níveis suficientes para suportar a coadaptação progressiva dos módulos.
2. Enquanto há forte pressão seletiva para a convergência de uma espécie, uma vez que ela encontre um nicho (papel) útil para a resolução do problema geral, *não* há nenhum estímulo para que várias (ou mesmo todas) espécies converjam para o *mesmo* nicho.
3. A evolução de espécies em populações isoladas também elimina possíveis efeitos destrutivos oriundos do casamento entre espécies (blocos construtivos de maior granulosidade) distintas, em referência à idéia dos letais mencionada anteriormente. A arquitetura somente trata as interdependências entre os componentes mediante a avaliação de cada espécie no contexto das demais.
4. Como cada espécie evolui separadamente, pode-se paralelizar e customizar cada instância de AE de modo a arcar eficientemente com as peculiaridades da subtarefa (módulo) correspondente. Assim, diferentes esquemas de codificação, ou mesmo diferentes dialetos de AEs, podem ser empregados em uma instância da arquitetura. Além disso, tal abordagem usualmente requer menos cálculo computacional que AEs tradicionais, haja vista que o tamanho das populações das espécies *tende* a ser reduzido e a convergência do processo como um todo (em termos do número requerido de gerações) *tende* a ser mais rápido.
5. Finalmente, os resultados apresentados por vários trabalhos aplicando, estendendo ou formalizando esse *framework* [46, 140, 220, 248, 356, 376, 377, 438, 484, 485, 487], compreendendo desde aqueles voltados ao projeto automático de sistemas nebulosos [119, 363, 364] até outros destinados à evolução de sistemas neurais [167, 334, 335, 510, 512], vêm a corroborar a expectativa de que sistemas coevolutivos artificiais podem, de fato, alavancar os benefícios e potencialidades promovidos pela CE.

6.8 Computação evolutiva e sistemas emergentes

Em contraposição a sistemas construídos sob o paradigma *top-down*, que normalmente contam com uma (ou poucas) “divisão executiva” inteligente integrando as atividades realizadas pelos módulos em operação, sistemas *bottom-up* resolvem problemas intrincados mediante a coordenação descentralizada de massas de elementos relativamente simplórios [236]. Nesses sistemas, tais elementos primitivos (isto é, agentes com capacidade restrita), que residem em um patamar de organização inferior (dito *explícito* ou *mecânico*), passam a apresentar, como fruto de um sem-número de *interações* locais ocorrendo continuada e paralelamente, um macrocomportamento perceptível e interpretável (por um observador externo, é claro) que pertence a uma escala de organização superior (denominada *implícita* ou *emergente*).

Num jargão mais técnico, sistemas naturais ou artificiais apresentando tais propriedades costumam ser nomeados de *sistemas adaptativos complexos* (SACs), exemplos dos quais encerram o cérebro humano, as

sociedades de formigas e os complexos demográficos urbanos [236]. Por sua vez, o fenômeno que captura o surgimento espontâneo e gradual de padrões funcionais (de comportamento ou mesmo de computação [154]) no nível macroscópico⁽²¹⁾, em função das regras do nível explícito, passou a ficar referenciado como *emergência*. Correlatas ao fenômeno da emergência encontram-se outras propriedades (fenômenos) exibidas por SACs, a saber: (i) *auto-organização*, definida por alguns como sendo a “emergência espontânea de ordem” a partir de sistemas inicialmente aleatórios [245]; (ii) *coletividade*, relativa ao surgimento e manutenção de padrões interativos estáveis e suas implicações globais; (iii) comportamento *cooperativo*, em corroboração com a vertente mais holística de que “o todo é mais que a soma das partes”; e (iv) realimentação negativa (importante em sistemas auto-regulados, onde perturbações são respondidas com ações de efeito contrário) e positiva (onde alterações passam a ser amplificadas com o decorrer do tempo).

Forrest [154] considera um AG simples como um exemplo claro de modelo computacional de caráter emergente, fruto tanto das propriedades coletivas exibidas pelos indivíduos da população ao longo das gerações como do seu processo de busca *dual*, este ocorrendo tanto em um patamar mecanístico, relativo ao *framework* cíclico de “reprodução-variação-seleção”, como em um nível emergente, relativo ao paralelismo implícito decorrente da busca pelo espaço de alta dimensão formado pelos esquemas. Como exemplos de SACs, encontram-se os Sistemas Classificadores (ver Seção 6.1), cujo comportamento emergente é resultado direto das interações competitivas entre as inúmeras regras (classificadores) mediadas por um processo de microeconomia simulada, o qual atribui crédito em proporção ao grau de utilidade de cada regra para o sistema como um todo (aprendizado por reforço) [155]. Outros trabalhos aplicando AEs com o objetivo de estudar comportamento ou computação emergente abrangem: o treinamento de autômatos celulares, em que AGs, ou variações coevolutivas destes, são empregados para encontrar regras de atualização de autômatos de modo a induzir um mapeamento pré-especificado, indo desde a sua configuração inicial até a configuração de atrator [325, 360]; a emergência de cooperação em grupo ou de normas sociais complexas em ambientes simulando conflitos de interesse ou outros comportamentos estratégicos — como o recurso da imitação (*mimicry*) — via modelos evolucionários da Teoria dos Jogos [29, 74, 104, 434, 457]; e a decomposição e resolução emergentes de problemas complexos mediante a coevolução de espécies simbióticas [203, 217, 219, 375]. O projeto apresentado na Seção 8.4 enquadra-se nessa última frente de pesquisa, abordando o problema de coordenação em sistemas multiagentes (Cap. 5).

6.9 Aplicações de algoritmos evolutivos em sistemas inteligentes modulares

O emprego de técnicas evolucionárias para a configuração de sistemas inteligentes modulares começou a ser explorado mais sistematicamente a partir de meados da década de 1980, com os primeiros trabalhos combinando CE e RNAs ou CE e SMAs [109], e ganhou maior corpo e respaldo ao longo da década seguinte, sendo esta uma corrente de pesquisa ainda muito em voga por estes dias. Nesta seção, passa-se a avaliar,

⁽²¹⁾Tais padrões implícitos de mais alta resolução costumam ser chamados de *epifenômenos* [154].

concisamente, algumas das várias abordagens que se inserem nessa conjuntura, tendo-se como referência a classe de sistemas modulares estudada ao longo dos Caps. 2 a 5 desta tese. Esta revisão, que não pretende ser de todo exaustiva, é também suportada pelas seções introdutórias e de revisão bibliográfica constando dos artigos do Cap. 8.

Redes neurais. Dentre os sistemas modulares aqui pesquisados, as RNAs, com suas inúmeras variações, constituem, inegavelmente, a subclasse com maior número de trabalhos envolvendo aplicações de técnicas evolutivas. Yao [501] provê um esquadramento bem considerável desses trabalhos, apondo, grosso modo, três principais níveis de atuação⁽²²⁾: (i) *evolução dos pesos* das conexões sinápticas, apresentando-se como uma estratégia alternativa a métodos de treinamento calcados em informação do gradiente; (ii) *evolução da regra de aprendizado*, compreendendo tanto a otimização de parâmetros de controle (taxa de aprendizado, taxa de *momentum*, etc.) como a descoberta dinâmica de novas regras de aprendizado; e (iii) *evolução da topologia*, envolvendo tanto o projeto otimizado dos padrões de conexão (número de camadas e suas interconexões) como a definição automática das não-linearidades do modelo resultante (funções de ativação e seus parâmetros associados). A Fig. 6.5 traz um arcabouço genérico esboçado por Yao [501] ressaltando possíveis inter-relações entre os três níveis de adaptação supracitados — nessa ilustração, quão mais interna a camada, mais rápida é a escala de tempo evolucionária inerente. Passa-se, em seguida, a descrever alguns aspectos vinculados aos primeiro e último níveis, comentando sobre algumas propostas relacionadas.

A motivação para se otimizar pesos via AEs é a de se apelar para a sua “máquina” de busca adaptativa e (semi-)global de sorte a lidar efetivamente com as peculiaridades impostas pelas superfícies de *fitness* associadas às funções-custo comumente adotadas (ver Subseção 2.4.2); peculiaridades estas (tais como não-diferenciabilidade e multimodalidade) de difícil trato por métodos de otimização mais tradicionais. A estratégia evolucionária para o ajuste de pesos consiste, quase sempre, de duas grandes etapas [501]. A primeira diz respeito a decidir que tipo de codificação dos parâmetros utilizar, binária ou não-binária. A segunda se refere a projetar os operadores de variação a serem empregados em concordância com a representação escolhida — determinando, por conseguinte, a “feição” do AE. As vantagens de se dispor de representações binárias se resumem à sua *simplicidade* e ao seu alto grau de *generalidade*, além do fato de já se ter disponível um “arsenal” de operadores genéticos prontos para serem aplicados. Um dos argumentos contrários a esse tipo de representação é que ele é *não-escalável*, ou seja, comumente implica a manipulação de longas cadeias de *bits*, acometendo, pois, a eficiência da busca evolutiva. Outro forte agravante passou a ser capturado pelo que se denomina *problema de permutação*, também conhecido como *problema de convenções competitivas* (do inglês, “*competing conventions problem*”) [480]: qualquer permutação de nós escondidos irá produzir RNAs funcionalmente equivalentes mas com diferentes representações cromossômicas, tornando, muitas vezes, ineficaz o papel do operador de recombinação utilizado⁽²³⁾. Essa ineficácia se deve à possibilidade de

⁽²²⁾Outros possíveis pontos de aplicação de AEs em RNAs incluem a seleção (ou geração) adaptativa de atributos [278] e de modelos [246], a iniciação mais otimizada dos pesos das conexões, a seleção automática das amostras mais significativas (e relevantes para o treinamento) do conjunto de dados disponível [310] e a extração de regras inteligíveis a partir de modelos neurais já treinados (Seção 2.2).

⁽²³⁾Está-se assumindo, aqui, o uso de uma mesma função de ativação para todos os neurônios escondidos e que não

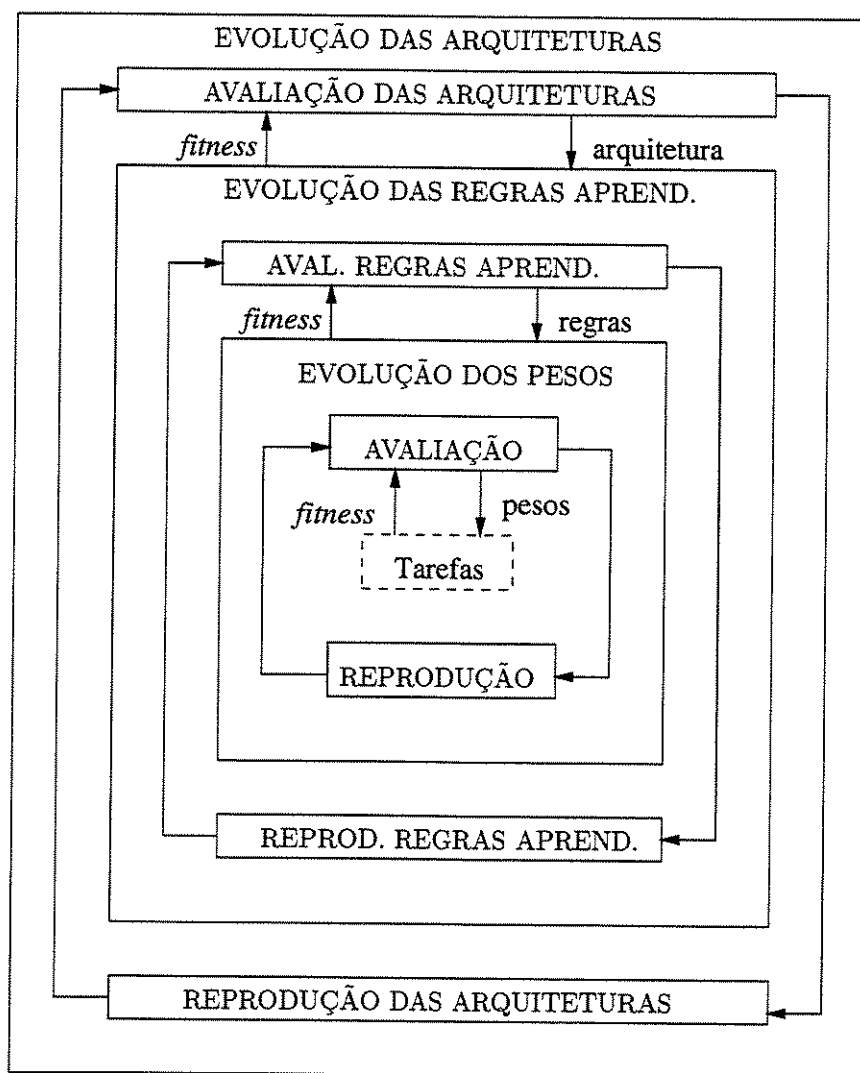


Fig. 6.5: Um arcabouço geral para caracterização de RNAs evolutivas. (Fonte: Yao [501].)

se gerar prole herdando blocos de genes idênticos (logo, redundantes) vindos de posições diferentes de pais funcionalmente equivalentes — muito similar ao caso de geração de letais comentado anteriormente. Como forma de suplantar tais obstáculos, passou-se a recorrer, com maior frequência, ao uso de representações em ponto flutuante, ocasionando a necessidade de se empregar operadores genéticos mais customizados. Nesse sentido, Montana e Davis [331] foram uns dos primeiros a estipular um bom repertório de operadores para AGs incorporando várias heurísticas existentes acerca do treinamento de RNAs *feedforward*. Os autores constataram que o AG com codificação real foi superior, tanto em eficiência (menor tempo de treinamento) como em eficácia (menores taxas de erro), àquele binário. Por outro lado, desde que são adequados, particularmente, para o tratamento de problemas de otimização contínua, tanto algoritmos embasados em PE como em EEs passaram a ser igualmente empregados para o ajuste dinâmico de pesos em RNAs, apresentando-se como alternativas, às vezes mais robustas, aos AGs tradicionais [151, 182, 371, 408].

É válido comentar neste espaço sobre alguns estudos explorando o treinamento híbrido de RNAs, com a intenção de incrementar a eficiência de busca dos AEs mediante a sua combinação com outros métodos não-populacionais mas com maior habilidade de aprimoramento localizado das configurações de redes pesquisadas (ver Seção 6.6). Nesse grupo, encontram-se a abordagem memética proposta recentemente por Abbass [4] e os trabalhos sondando as potencialidades e os deméritos por trás de técnicas lamarckianas e baldwinianas [47, 185, 258]. Ao treinarem RNAs booleanas usando árvores gramaticais, Gruau e Whitley [185] verificaram que a busca genética baldwiniana (i. e., que lançou mão do aprendizado somente para mudar a superfície de *fitness*) foi tão efetiva quanto a estratégia de busca lamarckiana, sendo ambas mais eficientes e eficazes que a busca genética convencional. Porém, a mesma constatação não pôde ser depreendida por Ku e colaboradores [257], quando avaliando, comparativamente, o desempenho de estratégias lamarckianas e baldwinianas embutidas em AGs celulares (ver Seção 6.5) para a difícil tarefa de treinamento de RNAs recorrentes. Os autores conjecturaram que a ineficiência da estratégia baldwiniana se deveu ao fato de que muitos pesos de uma dada instância de RNA podem ser alterados simultaneamente no curso de um só ciclo de aprendizado, tornando bastante dificultoso para os operadores genéticos realizarem a assimilação genética correspondente — quer dizer, reproduzirem os efeitos de tais mudanças fenotípicas no genótipo do indivíduo. Num estudo correlato [258], os mesmos autores passaram a investigar diferentes variações de estratégias lamarckianas sob o mesmo domínio de aplicação, ou seja, treinamento de RNAs recorrentes para problemas de controle e de reconhecimento de seqüências. Os resultados obtidos comprovaram a tese de que, quando confrontados com AGs celulares sem busca local, as variações lamarckianas não somente promoveram elevada redução no tempo computacional médio exigido para realizar a busca como também proporcionaram uma melhora significativa na qualidade das soluções encontradas.

Em se tratando da aplicação de AEs para o desenho automático do leiaute estrutural de RNAs, Miller *et al.* [322] foram um dos primeiros a apontar as vantagens de tais métodos frente a outros algoritmos construtivos ou subtrativos (ver Seção 2.5). Na ocasião, os autores caracterizaram a superfície de *fitness* inerente ao processo de busca pela arquitetura ótima como sendo, tipicamente, (i) *infinitamente vasta*, já que o número de conexões e unidades escondidas pode ser ilimitado; (ii) *não-diferenciável*, uma vez que mudanças há evolução concomitante da topologia da rede.

no número de unidades e conexões devam ser de natureza discreta, gerando efeitos de descontinuidade em termos de desempenho; (iii) *complexa e ruidosa*, visto que tende a ser independente das condições iniciais; (iv) *deceptiva*, pelo fato de que redes estruturalmente similares podem ter desempenhos bem dissimilares; e (v) *multimodal*, pois genótipos bastante distintos podem apresentar desempenhos bem similares.

Do mesmo modo que para a evolução de pesos, duas grandes etapas estão envolvidas na evolução topológica de RNAs, compreendendo a definição do esquema de codificação genotípica das arquiteturas e a configuração dos operadores de variação a serem implementados [501]. Um dos aspectos-chave com relação à fase de codificação das soluções é o de se determinar o quanto de conhecimento acerca do formato da arquitetura deva ser especificado *a priori*. De um lado do espectro, estão aqueles esquemas de representação, conhecidos como de *codificação direta*, que estipulam que todo conhecimento prévio acerca dos nós e conexões deva ser, de alguma forma, já embutido no formato da representação. Do outro lado, encontram-se os métodos de *codificação indireta*, que estabelecem que somente os parâmetros mais importantes devam ser representados (tais como, número de camadas e neurônios escondidos), deixando a decisão sobre outros detalhes arquiteturais à mercê de um segundo processo de treinamento a ser utilizado. Esquemas de codificação direta podem ser ainda divididos em duas subclasses compostas por: (a) aqueles que separam a evolução da arquitetura do processo de correção dos pesos; e (b) aqueles que consideram a evolução simultânea de arquitetura e pesos num só processo de otimização/aprendizado. Exemplos de trabalhos pertencentes à subclasse (a) encerram aqueles de primeira geração [322, 373, 410, 480]. Exemplos de abordagens aderentes à segunda subclasse (mais genérica) são mais recentes e abrangem tanto aquelas para redes *feedforward* [142, 282, 305, 479, 503] como para redes recorrentes [20, 315]. Por outro lado, além de serem considerados por alguns como “mais biologicamente plausíveis” [252], esquemas de codificação indireta tendem a gerar cadeias genotípicas mais compactas, ao preço de nem sempre assegurarem boa capacidade de generalização para os modelos resultantes [501]. Exemplos de abordagens provendo diferentes esquemas indiretos de representação topológica encerram o modelo de representação fractal proposto por Merrill e Port [319], o modelo de AGs estruturados [106, 108], discutido na Seção 6.5, e os vários *métodos desenvolvimentistas* [184, 252], assim chamados por codificarem regras, ou grafos, ou mesmo gramáticas, a serem evoluídos, especificando como as arquiteturas de RNAs devam ser construídas para fins de avaliação.

Ainda concernente a estratégias não-convencionais para codificação das estruturas arquiteturais, é pertinente mencionar algumas abordagens que, por decisão de projeto, já estipulam pela decomposição modular da representação das soluções, incorrendo em ganhos salientes em termos de eficiência (desempenho computacional) e eficácia (generalização e precisão). Andersen e Tsoi [17], por exemplo, parecem ter sido os primeiros a trabalharem com uma população de neurônios escondidos em vez de uma população de RNAs completas, concebendo uma abordagem evolucionária construtiva parecida com o algoritmo CasCor (Seção 2.5). Nesse modelo, usa-se somente uma instância de RNA *feedforward*, sendo que cada camada escondida dessa rede (pode-se ter mais de uma) passa a ser construída automaticamente mediante o emprego de um AG configurado com compartilhamento de *fitness*. Esse AG especial encoraja a formação de diferentes detectores de características, quer dizer, espécies de neurônios escondidos, na população. Outras propostas que aderem ao conceito de espécies de neurônios escondidos são aquelas de gênero coevolutivo. Nesse âmbito,

encaixam-se a abordagem “cooperativa-competitiva” endereçada por Whitehead e Choate [479] — voltada para a definição automática dos parâmetros das funções de base radial compondo modelos RBF, para fins de previsão de séries temporais — e a abordagem SANE (de *Symbiotic Adaptive Neuroevolution*), de autoria de Moriarty e Miikkulainen [334] — destinada a tarefas de neurocontrole via aprendizado por reforço. Na seqüência, passa-se a descrever com mais detalhes a versão hierárquica dessa última abordagem [335, 336], a qual serviu de referência para os projetos apresentados no Cap. 7 e na Seção 8.1.

Na versão hierárquica de SANE, duas populações, uma de neurônios escondidos e outra de *blueprints* (“esquemas”) de redes *feedforward* com uma camada intermediária, são evoluídas concorrentemente. O motivo em se ter uma população separada de neurônios (camada inferior) é o de possibilitar a *decomposição dinâmica* do espaço de busca de topologias de RNAs, na medida em que grupos de neurônios passam a se especializar em partes diferentes das soluções geradas (ver Subseção 6.7.2). Desse modo, um maior grau de diversidade passa a ser assegurado e o algoritmo não se torna tão vulnerável a problemas de convergência prematura como em AGs convencionais. Já a população de *blueprints* (patamar de cima) fica responsável pela busca de combinações efetivas dos neurônios produzidos pela outra população. A manutenção desses esquemas de redes conduz a avaliações mais precisas dos neurônios, ao mesmo tempo que promove a concentração do esforço de busca sobre regiões mais promissoras por meio da alocação de mais oportunidades de reprodução (*trials*) a neurônios que “trabalham” bem em grupo. A Fig. 6.6 contrasta os passos adotados pela abordagem SANE com aqueles tradicionalmente seguidos para o treinamento evolutivo de arquiteturas de redes neurais.

Uma iteração do processo evolutivo por trás da versão hierárquica de SANE pode ser dividida em duas fases, de *avaliação* e de *reprodução*. Durante a fase de avaliação, cada *blueprint* (i. e., conjunto de ponteiros para neurônios da outra população) é interpretado como uma instância de arquitetura de RNA cujo desempenho passa a ser avaliado em consonância com a tarefa em vista. Por outro lado, cada indivíduo (neurônio escondido) da população do nível de baixo é codificado por uma série de definições de conexões (e seus pesos). Tais definições descrevem, no todo, um ou mais caminhos que se originam num dado neurônio de entrada e terminam num dado neurônio de saída, e perpassam, necessariamente, pelo neurônio escondido representado por esse indivíduo. No mais, cada entidade da população inferior tem seu *fitness* calculado como sendo a *média* dos graus de aptidão das N_r melhores redes em que participa como componente. Já na fase de reprodução, ambas as populações passam a ter seus indivíduos ordenados e selecionados em função de seus valores de *fitness* (via *fitness ranking*). Para cada indivíduo pertencente à elite (i. e., $N_b = 25\%$ melhores cromossomos no momento), escolhe-se aleatoriamente um parceiro (também da elite) para que se gere um filho via *crossover* de um ponto. O outro filho do casal será um clone direto de um dos pais, o que é estipulado aleatoriamente. A prole passa em seguida a ocupar o lugar dos piores indivíduos da população atual a fim de se compor a próxima geração. Finalmente, com uma taxa de $\Theta_m = 1\%$, aplica-se mutação sobre cada posição de cada cromossomo de ambas as populações. Em suma, o processo evolutivo que particulariza a SANE hierárquica abarca duas buscas evolucionárias interdependentes: uma, de essência explorativa (“diversificativa”), destinada a encontrar neurônios (blocos construtivos) que realizem subtarefas úteis e complementares; e outra, de gênero “intensificativo”, dirigida a manter e aprimorar continuamente

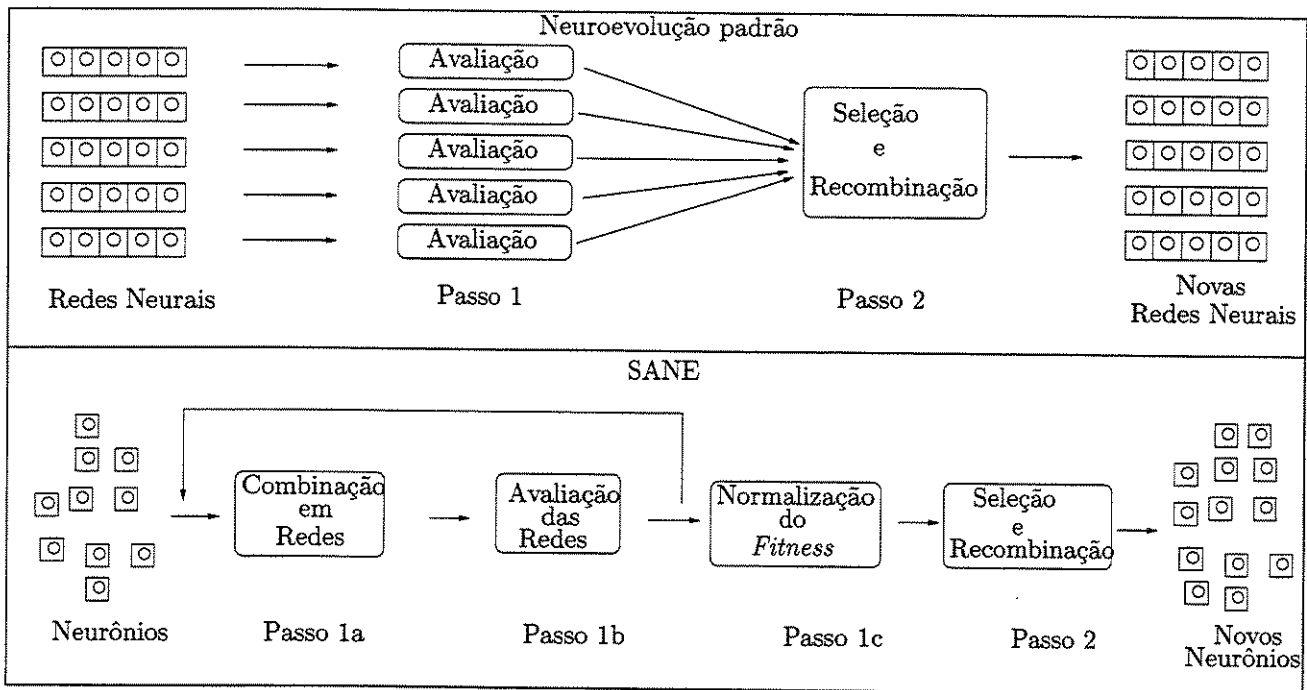


Fig. 6.6: Uma ilustração comparativa entre o processo neuroevolutivo “canônico” e aquele induzido por SANE. A abordagem convencional mantém uma população de RNAs, estas avaliadas independentemente. Por outro lado, SANE mantém uma população de neurônios, os quais são avaliados conjuntamente via montagem de instâncias de RNAs (times) — na versão hierárquica, tais instâncias de redes passam a ser determinadas por esquemas específicos de combinação de neurônios (*blueprints*), os quais integram uma segunda população com ciclo evolutivo em separado. Cada neurônio recebe seu *fitness* normalizado com base no desempenho das redes (ou esquemas) das quais participa. (Fonte: Moriarty & Miikkulainen [335].)

os melhores esquemas de justaposição de blocos construtivos cooperativos.

Finalmente, cabe ainda fazer uma breve alusão às abordagens que promovem a evolução conjunta dos padrões de conexão estrutural das redes com os formatos das funções de ativação de seus neurônios. Uma primeira motivação para tal empreendimento já foi dada na Subseção 1.2.1, ao passo que justificações mais formais ou experimentais para o projeto de redes neurais com não-linearidades heterogêneas podem ser encontradas em outras referências [221, 295, 311, 465]. Uma amostra significativa desses trabalhos é analisada no artigo que discorre sobre a arquitetura HCGA (Seção 8.1). Esta encerra uma extensão do modelo SANE que lida com o problema de especificação automática das não-linearidades de RNAs heterogêneas mediante o emprego de populações especializadas de neurônios escondidos (ou seja, várias espécies diferentes no nível inferior).

Máquinas de vetores-suporte. Contrastando com o grande volume de trabalhos já publicados em redes neurais evolutivas, ainda muito incipiente é a linha de investigação voltada à aplicação de técnicas

evolutivas para o tratamento de aspectos vinculados ao treinamento de SVMs. Apenas dois trabalhos recentes puderam ser recuperados no curso de levantamento bibliográfico. O primeiro, conduzido por Lee *et al.* [280], dispõe de máquinas de vetores-suporte para a tarefa não-trivial de reconhecimento de faces, sendo o conjunto de atributos das amostras de treinamento das máquinas gerado, via AGs, a partir de características de mais baixa resolução. Os autores reportaram altas taxas de reconhecimento de faces com essa integração de técnicas, incorrendo também em ganhos de memória. A segunda abordagem, proposta por Saavedra e colaboradores [404], adota heurísticas fundamentadas em AGs com vista à seleção de modelos de SVM. Isso foi possível mediante a adaptação dos parâmetros associados às funções-*kernel* empregadas pelo algoritmo de otimização *Kernel-Adatron*, este discutido na Seção 3.2.

Comitês de máquinas. Salvo o mecanismo HGT-GAME detalhado na Seção 8.3, parece que somente Hong *et al.* [208] se propuseram a investigar o potencial de uma técnica evolutiva quando do treinamento adaptativo de MEs (no caso, do tipo localizado). Por sua vez, no âmbito mais abrangente de sistemas conexionistas modulares, García-Pedrajas *et al.* [168] conceberam um modelo coevolutivo cooperativo, chamado de COVNET e baseado no *framework* de Potter & De Jong [375] e na abordagem SANE [335], o qual foi destinado à evolução completa (pesos + topologias) de arranjos compostos de vários subcomponentes neurais. Assim como a variante com aninhamento da abordagem HCGA (ver Subseção 8.1.1), COVNET lança mão de uma arquitetura com dois níveis de evolução simultâneos: no nível de cima, encontra-se uma população de configurações de redes neurais, enquanto no nível de baixo está uma população de subcomponentes neurais (chamados de *nódulos* pelos autores). Tais *nódulos* são compostos de um conjunto de nós exibindo interconexões arbitrárias entre eles⁽²⁴⁾, apresentando, portanto, maior granulosidade que os neurônios simples usados na versão original de HCGA (artigo da Seção 8.1). Já um indivíduo da população de cima é interpretado como uma combinação de um número finito de *nódulos*, sendo esse número fixado no decorrer do processo evolutivo e considerado o mesmo para todas as combinações. A saída de uma RNA é dada pela *soma* das saídas dos *nódulos* que a constituem, todos eles “enxergando” o mesmo espaço de entrada — destarte, os mapeamentos gerados pelos *nódulos* não são redundantes como em *ensembles*. Assim como HCGA, COVNET adota várias subpopulações de *nódulos* no nível inferior como forma de preservar diversidade — cada subpopulação evoluindo isoladamente, ou seja, *não* intercambiando soluções parciais via processos de migração. Diferentemente da versão aninhada de HCGA, além de adotar PE para a evolução dos *nódulos*, COVNET *não* investiga a evolução simultânea de nós heterogêneos, ou seja, todos os nós têm a mesma função de ativação associada.

Estendendo as potencialidades do modelo COVNET, García-Pedrajas *et al.* [167] conceberam a abordagem MOBNET. O ponto-chave de MOBNET está na modelagem do processo de atribuição de crédito aos *nódulos* como um problema de otimização multiobjetivo, lançando mão dos conceitos de dominância e de fronteira de Pareto (ver Cap. 5). Desse modo, tornou-se possível a determinação do valor de *fitness* de um *nódu*lo tomando por base diferentes critérios de desempenho, estes possivelmente conflitantes entre si. Os autores adaptaram o método NSGA (ver Seção 6.5) para o escopo de PE e investigaram o potencial de várias

⁽²⁴⁾Há a possibilidade, inclusive, de se gerar recorrência interna entre os nós de um *nódu*lo. Sendo o caso, os autores projetaram um procedimento que transforma tais *nódulos* com recorrência local em *nódulos feedforward* equivalentes.

métricas para atuarem como possíveis critérios de desempenho (ou seja, objetivos) quando da avaliação da aptidão dos nódulos. Uma vez que tal estratégia mostrou-se pertinente em experimentações preliminares, García-Pedrajas *et al.* [167] também passaram a levantar métricas para a avaliação multiobjetivo das redes (população superior). Tanto COVNET como MOBNET foram avaliadas perante vários outros métodos (inclusa aqui a abordagem ME com especialistas lineares [226]) sobre algumas instâncias de problemas de classificação, estas presentes na Tab. C.1. De acordo com os resultados obtidos a partir de uma série de execuções [167], MOBNET mostrou-se ser a melhor abordagem em termos de acuidade (taxa de erro) e complexidade (número de neurônios).

Em se tratando de *ensembles*, por sua vez, é mais comum notar o emprego de AEs (sobretudo AGs e PE) como ferramenta alternativa para a agregação automática de múltiplos estimadores. A maioria dos trabalhos publicados nessa linha de *ensembles* evolucionárias gira em torno da configuração otimizada dos modelos individuais de estimadores, geralmente RNAs, a serem combinados. Nesse contexto, encaixa-se a abordagem proposta por Liu e Yao [296, 502, 504], que estipula a combinação (via voto majoritário, ou média ponderada, ou versão recursiva do método de quadrados mínimos, ou um AG específico) das instâncias completas de redes neurais relativas aos cromossomos compondo a última geração de uma execução do processo evolutivo EPNet [503]. O que justifica esse procedimento é que outros indivíduos da população de redes *feedforward* poderão conter informações adicionais a serem usadas para auxiliar na melhoria da capacidade de generalização da melhor rede encontrada. Entretanto, a principal deficiência dessa estratégia é que as redes-componentes da última geração tendem a ser muito similares umas às outras (geralmente, por serem clones alterados da melhor rede encontrada), algo decorrente da perda constante de diversidade ao longo das gerações. Apesar de demonstrarem altas taxas de reconhecimento, tais estimadores são propensos a exibirem dificuldades para compor um time de sucesso, haja vista seus padrões de erro serem altamente correlacionados [369, 397]: onde um erra, a maioria, senão todos, irá errar.

Nesse sentido, Liu, Yao e Higuchi [298] se ocuparam em estender a abordagem anterior embutindo, como parte inerente do processo evolutivo, a idéia de especiação via compartilhamento de *fitness* entre as RNAs da população e a idéia de aprendizado por correlação negativa, esta já proposta, anteriormente, pelos mesmos autores [297] (ver Subseção 4.1.2). A nova versão, de cunho mais cooperativo e denominada ENNCL (de *Evolutionary Ensembles with Negative Correlation Learning*), encoraja diferentes RNAs individuais a se especializarem em partes ou aspectos distintos do conjunto de treinamento. Ao mesmo tempo, ela explora as interações entre os padrões de erro apresentados pelas redes como meio de guiar a sua evolução e, posteriormente, auxiliar na seleção do número adequado de componentes da *ensemble* final. O número final de componentes será dado, após a execução do processo evolutivo, ou por todos os indivíduos da última geração ou pelo número de *clusters* encontrados via aplicação do algoritmo *K-Means* [197] sobre essa população. Nesse último caso, a melhor rede (ou seja, com maior valor de *fitness*) de cada *cluster* definirá cada componente. Mais recentemente, Liu *et al.* [299] se ocuparam de avaliar, formalmente, a conexão entre a idéia de compartilhamento de *fitness* e o conceito de *informação mútua explícita*, este relacionado a noções da área de Teoria da Informação [198]. O objetivo foi o de lançar mão do critério de informação mútua entre cada RNA *k* e o resto da população como um fator de correção a ser aplicado sobre o seu grau de aptidão

fit_k , em proporção ao número de membros do nicho atual a que tal indivíduo pertence. A nova versão de ENNCL foi denominada ELMI, de *Evolutionary Learning with Mutual Information*.

Essa estratégia evolutiva proposta por Liu e colaboradores guarda certa similaridade com aquela concebida paralelamente por Friedrich [162], no sentido de que ambas empregam técnicas evolutivas incrementadas com noções de treinamento descorrelacionado das redes. Entretanto, em vez de ser o algoritmo de aprendizado por correlação negativa, o algoritmo de descorrelação adotado por Friedrich [162] teve como base aquele proposto por Rosen [397]. Ademais, a abordagem de Friedrich recorre ao método de generalização por empilhamento [492] como instrumento de coligação das redes evoluídas num primeiro estágio. Lee *et al.* [281] também advogaram o uso de AGs com compartilhamento de *fitness* como uma abordagem eficiente para a geração de redes com papéis complementares, cada uma proveniente de um nicho distinto. Em vez do critério de informação mútua adotado por Liu *et alia* [299], Lee *et al.* empregaram a distância euclidiana entre as médias dos graus de suporte e duas outras métricas estatísticas com base em Teoria de Informação (coeficiente de correlação de Pearson e entropia cruzada) como fatores de correção dos valores de *fitness* e definição das espécies — essas métricas são descritas na Tab. 4.1. O recurso de compartilhamento de *fitness* foi também considerado por Bruce e Miikkulainen [65] quando da evolução em conjunto de diferentes redes especialistas, cada uma dotada de um instrumento especial que indica o seu grau de confiança (entre zero e um) para com cada amostra de entrada sob análise — similar, portanto, à arquitetura Pandemonium [433]. A rede com maior grau de confiança para uma dada amostra produzirá a saída do grupo, sendo que a “função de confiança” de uma certa rede especialista passa a ser evoluída conjuntamente com seu mapeamento de E/S.

O algoritmo ADDEMUP (de *Accurate anD Diverse Ensemble-Maker giving United Predictions*), proposto por Opitz e Shavlik [353], também emprega um AG com o intuito de produzir uma população de estimadores — no caso, modelos neurais baseados em conhecimento, os quais podem ser facilmente traduzidos em regras “IF-THEN” — exibindo alta capacidade de predição e pouca sobreposição em suas distribuições de erro. A fim de se assegurar a diversidade do grupo de RNAs em evolução, o valor de *fitness* do cromossomo R_k (i. e., uma instância de rede qualquer) passa a ser calculado como sendo $Fitness(R_k) = Accuracy(R_k) + \lambda Diversity(R_k)$, em congruência com a equação clássica de desacoplamento do erro total de uma *ensemble* estipulada por Krogh & Vedelsby [256] (Eq. (4.9)). Com efeito, a medida de diversidade ($Diversity(R_k)$) é obtida por meio de uma estimativa do desvio entre a saída produzida pela k -ésima rede e aquela gerada pela média das saídas produzidas por todo o grupo — similar, portanto, ao conceito de ambigüidade definido por Krogh & Vedelsby. Quão mais diversa a *ensemble*, maior será o *fitness* de seus componentes. O parâmetro λ deve prover um balanço conveniente entre os dois critérios e, segundo Opitz e Shavlik, pode ter seu valor atribuído dinamicamente em concordância com os níveis atuais de acuidade/diversidade dos elementos da população. O algoritmo ADDEMUP é dito tentar gerar *ativamente* bons candidatos a componentes na medida em que põe mais ênfase (peso no *fitness*) sobre aquelas amostras não acertadas pela população corrente durante o seu treinamento pela metodologia de retropropagação do erro [354].

Zhou *et al.* [514, 515], seguindo a idéia de que, para alguns casos, a agregação de *alguns* componentes mais apropriados pode ser uma estratégia mais eficaz do que justapor *todos* aqueles disponíveis (ver Subseção 4.1.4), deram luz a uma abordagem calcada em AGs, denominada GASEN (de *Genetic Algorithm based Selective Ensemble*), destinada à seleção adaptativa do melhor subconjunto de RNAs para a composição da *ensemble* final. De acordo com essa proposta, treina-se, primeiramente, um certo número de instâncias de RNAs, tornando-se estas possíveis candidatas para integrar a *ensemble*. Posteriormente, atribuem-se arbitrariamente pesos aleatórios a essas redes e emprega-se um AG com codificação em ponto flutuante para encontrar o conjunto ótimo de pesos que indiquem o grau de importância de cada instância de rede para o grupo. Somente aquelas redes cujos pesos associados são significativos (ou seja, maiores que uma limiar preestabelecido) deverão compor a *ensemble* final. Tal abordagem foi testada tanto em problemas de classificação como de regressão, sendo que as *ensembles* resultantes apresentaram desempenho superior àquele conseguido por combinações indiscriminadas de todos os componentes disponíveis (i. e., sem seleção adaptativa), mesmo quando tais combinações foram produzidas pelas técnicas de *bagging* e *boosting*. Outro trabalho nessa direção foi dirigido por Ruta & Gabrys [403], aplicando um AG com codificação binária customizada (esta também usada no artigo da Seção 8.2) para a seleção automática de componentes dentre vários modelos treinados antecipadamente, cada um oriundo da aplicação de um algoritmo de indução diferente — 15, no total. Paralelamente, Sirlantzis *et al.* [432] também aplicaram um AG simples para a configuração automática de sistemas multiclassificadores, com o intuito de avaliar o desempenho das *ensembles* produzidas perante classificadores individuais que haviam sido configurados otimadamente para as tarefas em questão — tais classificadores *não* pertenciam ao grupo de 12 possíveis componentes, estes oriundos de quatro classes distintas de indutores. Assim como no trabalho de Zhou *et al.* e Ruta & Gabrys, ficou comprovado, experimentalmente, que há sim fundamento em se adotar a seleção adaptativa de componentes para *ensembles*, algo que vem a corroborar os bons resultados apresentados pelo método evolutivo confeccionado para o projeto da Seção 8.2.

No mais, ainda no âmbito das *ensembles*, há outros (poucos) trabalhos investigando o uso de AEs para (i) a seleção automática de atributos; (ii) a seleção automática do melhor subconjunto de amostras de treinamento; ou (iii) a definição automática dos combinadores. A abordagem ERC (de *Evolutionary Resample and Combine*), proposta por Huber e Mayer [213], serve como exemplo representativo da segunda classe de trabalhos. De acordo com esse método, um AG dedicado evolui conjuntos de dados de treinamento para todos os estimadores, selecionando amostras (via reamostragem com reposição) guardadas em um repositório genérico. O *fitness* de cada cromossomo (codificado como uma lista de índices, cada índice apontando para uma amostra do repositório) é calculado de acordo com o desempenho do estimador (no caso, RNA) treinado com suas amostras associadas sobre um conjunto de validação à parte. Os autores também seguiram a idéia de Liu e Yao [504] de fazer uso de todas as redes correspondentes aos conjuntos de treinamento da última geração para o propósito de composição das *ensembles*. Outra estratégia adotada foi a de recrutar a rede correspondente ao melhor conjunto de treinamento (cromossomo) de cada geração; nesse caso, o número de componentes é igual ao número de gerações do AG. Pode-se pensar a abordagem ERC como sendo uma primeira versão evolutiva dos métodos estatísticos de reamostragem e combinação

adaptativas (*Arcing*) [63, 160].

Por outro lado, Kuncheva [259] investigou a aplicação de AGs para a redução da dimensionalidade dos padrões de treinamento dos componentes via seleção de atributos. No caso, cada cromossomo binário c_k do AG codifica um subconjunto de características em particular, de sorte que seu respectivo classificador seja construído e treinado usando somente esse subconjunto. A única forma de se promover diversidade aqui foi a de identificar o melhor subconjunto de classificadores de cada geração, retraindo os seus cromossomos correspondentes, sem levar em conta seus valores de *fitness*. Mais recentemente, Kuncheva e Jain [266] passaram a codificar em um só cromossomo os atributos correspondentes a todos os K componentes da *ensemble*. Para tanto, duas versões de AG foram propostas. Na primeira versão, cada componente usa um subconjunto *disjunto* dos atributos, sendo que o tamanho do cromossomo é igual ao número total de atributos (ou seja, cada posição guarda o índice $k \in \Gamma = \{1, \dots, K\}$ do componente a que tal atributo pertence). Na segunda versão, é permitido haver *sobreposição* entre os subconjuntos de atributos, sendo que cada posição do cromossomo (cujo tamanho continua a ser igual ao número total de atributos) pode assumir um dentre 2^K valores possíveis. Nesse caso, cada valor k representa um subconjunto $D_k \in 2^\Gamma$, onde 2^Γ alude ao conjunto-potência de Γ .

Finalmente, no que tange à evolução de combinadores, vale mencionar o trabalho conduzido por Harald & Kamstra [194], o qual empregou PE com parâmetros auto-ajustáveis para a evolução dos parâmetros de configuração de RNAs *feedforward* destinadas à combinação não-linear de diferentes preditores de séries temporais; no caso, o estudo focou sobre preditores de volatilidade em séries econômicas gerados pelas abordagens GARCH (*Generalized Autoregressive Conditional Heteroskedasticity*) e MAV (*Moving Average Variance*) [22]. Os resultados obtidos mostraram que as RNAs-combinadoras evoluídas apresentaram melhores desempenhos de generalização frente a combinadores triviais, como média, algoritmo dos quadrados mínimos e estimadores não-paramétricos baseados em *kernels* suavizantes [197].

Sistemas multiagentes. Assim como para sistemas conexionistas, a sinergia entre CE e SMAs tem sido explorada sobremaneira nos últimos anos. Na seqüência, é proporcionada uma rápida “pinçada” sobre algumas das frentes de investigação, com maior foco nos trabalhos explorando aspectos de coordenação em SMAs. Axelrod [28] foi um dos precursores no estudo da emergência de comportamento cooperativo em situações de conflito envolvendo múltiplos agentes egoístas (ver Cap. 5), analisando, mediante simulações controladas empregando AGs, o processo de evolução de estratégias para o conhecido jogo do “dilema-do-prisioneiro iterativo” (DPI). Vários trabalhos passaram a estender as “condições de contorno” analisadas por Axelrod, avaliando, por exemplo, (i) o emprego de autômatos (máquinas de estado finito) [323] e de RNAs [152] como mecanismos mais adequados para representar as estratégias dos jogadores; (ii) a possibilidade de estudar outros tipos de jogos, como os de coordenação e *chicken* [152, 457]; e (iii) a influência exercida por restrições espaciais (topológicas) sobre a taxa de disseminação de estratégias mais adaptadas pelo ambiente ou sobre a formação de novos padrões de cooperação entre os agentes [349].

Seredynski [421, 422] trouxe uma nova perspectiva para a configuração de AEs distribuídos ao considerar os próprios indivíduos da população em evolução como sendo agentes (nesse ponto, similar a AMs),

cada um com função-*payoff* e conjunto de ações próprios. Tais agentes exibem padrões de interação limitados, que são modelados via jogos não-cooperativos de N jogadores e determinados pelas vizinhanças topológicas predefinidas entre os indivíduos. Quando da aplicação do sistema coevolucionário resultante para propósitos de mapeamento e escalonamento de tarefas em ambientes paralelos, os agentes passaram a ser modelados tanto via autômatos como via AGs e SCs especiais (intitulados de “fracamente acoplados”). Ainda na linha da modelagem de comportamento social via AEs, é pertinente citar a abordagem MAZCS, introduzida recentemente por Hercog e Fogarty [202] e direcionada ao casamento entre sistemas classificadores e SMAs para lidar com o problema do “El Farol Bar”, sendo este uma generalização do interessante “jogo da minoria” [74].

Duas abordagens relevantes explorando a combinação “teoria de jogos e CE” para fins de coordenação em SMAs foram apresentadas, independentemente, por Bazzan [44] e por Seo *et al.* [420]. A primeira abordagem introduz um algoritmo de aprendizado por reforço evolutivo (ou seja, evolução de estratégias como forma de se alcançar equilíbrios em jogos não-cooperativos) como mecanismo descentralizado, escalável e de comunicação mínima para a coordenação das atividades de agentes não-benevolentes — no caso, motoristas interagindo em uma rede de cruzamentos de trânsito. Já a segunda abordagem trata da formação emergente de coalizões de estratégias jogando NDPI, uma variação do DPI em que N jogadores interagem simultaneamente. O interesse aqui é o de se permitir que estratégias mais complexas possam ser formadas, e também dissolvidas, dinamicamente (ao longo dos torneios) pela agregação de estratégias mais simples mas que, de alguma forma, sejam complementares entre si. Como resultado, obtêm-se, ao final das simulações, alianças entre jogadores exibindo alto grau de *generalização*, na medida em que (i) somente a jogadores com *fitness* acima da média da população é permitida a formação (inserção) de (em) grupos; (ii) tais alianças também competem entre si (por coevolução competitiva) e podem ser combinadas (via *crossover* atuando sobre seus membros) para a geração de novas alianças; e (iii) a cada jogador pertencendo a uma coligação são atribuídos um grau de confiança⁽²⁵⁾ e um posto (*rank*), usados como peso para ponderar a influência de sua ação individual (i. e., “cooperar” ou “delatar”) quando da determinação do próximo movimento (*move*) do grupo ante a um certo oponente.

Afora a proposta de Seo *et al.* [420], outras empreitadas também vêm investigando o emprego de AEs para o desenho automático (emergente) de organizações compostas por múltiplos agentes autônomos (vide Seção 5.2). Uma parte considerável dessas abordagens estima por combinar a grande flexibilidade de representação das soluções trazida com a PG com a alta capacidade de coadaptação dinâmica proporcionada pelos modelos de coevolução cooperativa, almejando, com isso, a geração customizada de times completos de agentes interdependentes, ou seja, desempenhando papéis diferentes no grupo. Encaixam-se nesse cenário os experimentos pioneiros de Haynes e Sen [199, 200], avaliando as potencialidades de diferentes tipos de operadores de variação e analisando as estratégias de caça coletiva emergentes em ambientes “presa-predador”; de Iba [218], realizando acareações sobre o desempenho exibido por times formados por agentes homogêneos ou heterogêneos; de Luke e Spector [300, 301], tratando tanto de estratégias de *breeding* (geração) dos mem-

⁽²⁵⁾ O grau de confiança pode ser também otimizado ao longo do processo evolutivo, como resultado de jogos DPI contra tanto estratégias aleatórias como outras bem conhecidas, tais como “TFT” (*Tit-for-Tat*), “Trigger”, “TF2T” e “AllD” [420].

bro do grupo como de mecanismos para sua coordenação em ambientes “presa-predador” e de futebol de robôs; e de Hara e Nagao [193], os quais se utilizaram da capacidade de encapsulação automática de módulos em PG para a definição emergente de (sub)grupos homogêneos dentro de times heterogêneos. Finalmente, vale ainda fazer menção ao recente trabalho de Goldman e Rosenschein [179], que investigou o potencial de se emergirem organizações completas em domínios orientados para informações (e não para tarefas, como os anteriores)⁽²⁶⁾. Tal investigação teve como suporte um modelo inspirado em autômatos celulares e fundamentou-se na possibilidade de transferência e evolução da própria informação capturada pelo grupo de agentes em conformidade com preceitos meméticos. Tal modelo “evolutivo” destinou-se a induzir a dinâmica de formação de padrões de (re)organização de agentes autônomos (no caso, para recuperação e agrupamento de documentos HTML) em consonância com o estado corrente do ambiente (ou seja, novos documentos sendo armazenados ou retirados dinamicamente da *Web*).

6.10 Últimas ponderações

A extensão deste capítulo está em direta proporção ao grau de atenção que vem sendo dado, nos últimos anos, a ferramentas computacionais de cunho evolutivo, atentando tanto para aspectos mais teóricos, relacionados à justificação das propriedades inerentes ao arquétipo conceitual introduzido na Seção 6.1 ou à concepção de novas abordagens derivadas, como para aspectos mais práticos, dirigidos à aplicação (e customização) dessas técnicas em um sem-número de domínios de pesquisa [30, 51, 178, 320]. Na sequência, passa-se a apresentar os projetos que definem as contribuições desta tese, os quais se servem diretamente do material apreciado neste capítulo.

⁽²⁶⁾ Goldman e Rosenschein [179] fazem distinção entre duas classes de modelos de organização. Nos casos em que a organização é *dirigida por tarefas*, novas tarefas podem requerer a formação dinâmica de novos padrões organizacionais para a sociedade. Por outro lado, quando a organização é *dirigida por informações*, ela pode manter a sua estrutura constante no decorrer das execuções das múltiplas tarefas que se servem da mesma base de informações.

Capítulo 7

Uma arquitetura coevolucionária, hierárquica e híbrida para *ensembles* heterogêneas de redes neurais artificiais

“Se você tem uma maçã e eu tenho uma maçã e nós as trocarmos, então você e eu continuaremos a ter, cada um, uma maçã. Mas, se você tem uma idéia e eu tenho uma idéia e nós as trocarmos, então cada um de nós terá duas idéias.”

George Bernard Shaw

Este capítulo discorre especificamente sobre COHENN⁽¹⁾, uma arquitetura evolucionária hierárquica, que é fundamentada nos conceitos de coevolução cooperativa e de treinamento híbrido e orientada para a síntese automática de *ensembles* heterogêneas de redes neurais *feedforward*. O material exposto aqui lança mão diretamente daquele veiculado nos segundo, quarto e sexto capítulos e nos três adendos deste documento. A organização deste espaço é a que se delineia a seguir. Na primeira seção, são alinhavadas algumas razões que motivaram o desenvolvimento do modelo coevolutivo COHENN, delimitando o seu escopo de investigação atual. Num segundo momento, na Seção 7.2, é encaminhada uma visão mais pormenorizada acerca das funcionalidades bem como dos elementos que caracterizam tal modelo arquitetural. Como forma de apreciação (e comparação com outras abordagens mais “estabelecidas”) das potencialidades exibidas por COHENN, um número bem extenso de simulações experimentais passa a ser reportado nas duas subseções que dão corpo à Seção 7.3. Por fim, fica destinada a Seção 7.4 a apresentar uma síntese dos principais pontos abordados ao longo do capítulo e a propalar algumas extensões ao modelo arquitetural passíveis de investigação futura.

⁽¹⁾Do inglês, “*Coevolutionary Heterogeneous Ensembles of Neural Networks*”.

7.1 Motivação e escopo de investigação

Um primeiro fator motivante por trás da concepção de COHENN diz respeito ao desempenho considerável exibido pelas abordagens SANE [334, 335] e HCGA (ver Seção 8.1) quando da evolução coadaptativa de módulos neurais homogêneos/heterogêneos, fato este corroborado por estudos correlatos relatados na literatura [167, 168, 479, 512]. Almejando-se a expansão dos horizontes de aplicação dessa estratégia evolucionária hierárquica de cunho simbiótico, passa-se, com COHENN, a investigar de forma sistemática aspectos de manutenção dos níveis de eficiência e eficácia em face da tarefa mais desafiadora de configuração customizada de arranjos completos de *ensembles* de RNAs.

Uma motivação complementar à anterior foi a de tentar cunhar uma metodologia “universal” dirigida ao tratamento coeso das três etapas comumente envolvidas no projeto de *ensembles* efetivas: criação, seleção e combinação de estimadores (ver Seção 4.1). A hipótese primal aventada neste capítulo é a de que os insumos providos pelo maquinário evolutivo (notadamente aquele incrementado com noções de simbiose e hibridismo) são particularmente bem apropriados para esse propósito. O fato é que, ao se tomar conta do conjunto já substancial de investigações conduzidas na linha de *ensembles* evolucionárias [162, 194, 213, 259, 266, 281, 296, 298, 353, 504, 515], pode-se dizer que a veracidade dessa suposição já vem sendo, pelo menos em parte, comprovada. Contudo, como evidenciado na Seção 6.9, a grande maioria desses trabalhos trata de questões *isoladas* relativas à atividade de construção automática de *ensembles* de estimadores, focando, quase sempre, em aspectos relacionados a uma (ou, quando muito, duas) das etapas supracitadas.

O escopo de investigação por trás da abordagem coevolucionária COHENN é ilustrado na Fig. 7.1, a qual aponta, de maneira clara, os “pontos de ajuste” (*hotspots*⁽²⁾) abarcados por esse modelo arquitetural. Pode-se verificar que aspectos relacionados aos estágios de criação, seleção e combinação de estimadores passam a ser tratados de maneira *uniforme* e *integrada* quando da síntese não-manual de *ensembles* de estimadores. (No decorrer deste capítulo, tais estimadores serão destinados exclusivamente a tarefas de classificação, embora isso não seja uma restrição, já que o mesmo ferramental pode ser empregado para outras tarefas relacionadas ao aprendizado supervisionado, tais como regressão e previsão.) Este é o mote distintivo por trás de COHENN: em vez de ter que estabelecer *a priori* a *ordem* de ajuste dos *hotspots*, o projetista transfere esse “encargo” para um procedimento de busca (aprendizado) *auto-regulado* (mediante as forças coevolutivas empreendidas dinamicamente entre os módulos), o qual “desvelará”, de forma automática, quais são as verdadeiras *interdependências* entre tais *hotspots*.

Trata-se de uma propriedade de grande apelo, haja vista que a decisão sobre quais *hotspots* manter fixos e quais devam ser ajustados em função dos primeiros é algo notadamente difícil de ser arbitrado. Muitas vezes, não se sabe, de antemão, quais pontos são realmente os mais “relevantes” para a instância de problema em vista; isto é, para quais pontos uma calibração fina bem apropriada se torna imprescindível de modo a

⁽²⁾O termo “*hotspot*” é oriundo da área de Engenharia de *Software*, sendo comumente usado para representar os pontos de configuração/extensão (geralmente, via sobrecarga de operações ou herança de classes) de *frameworks* computacionais [144]. No contexto deste capítulo, tal termo foi adotado num sentido semelhante, conquanto mais relaxado, como meio de se capturar os pontos de ajuste adaptativo abraçados pela arquitetura coevolutiva COHENN.

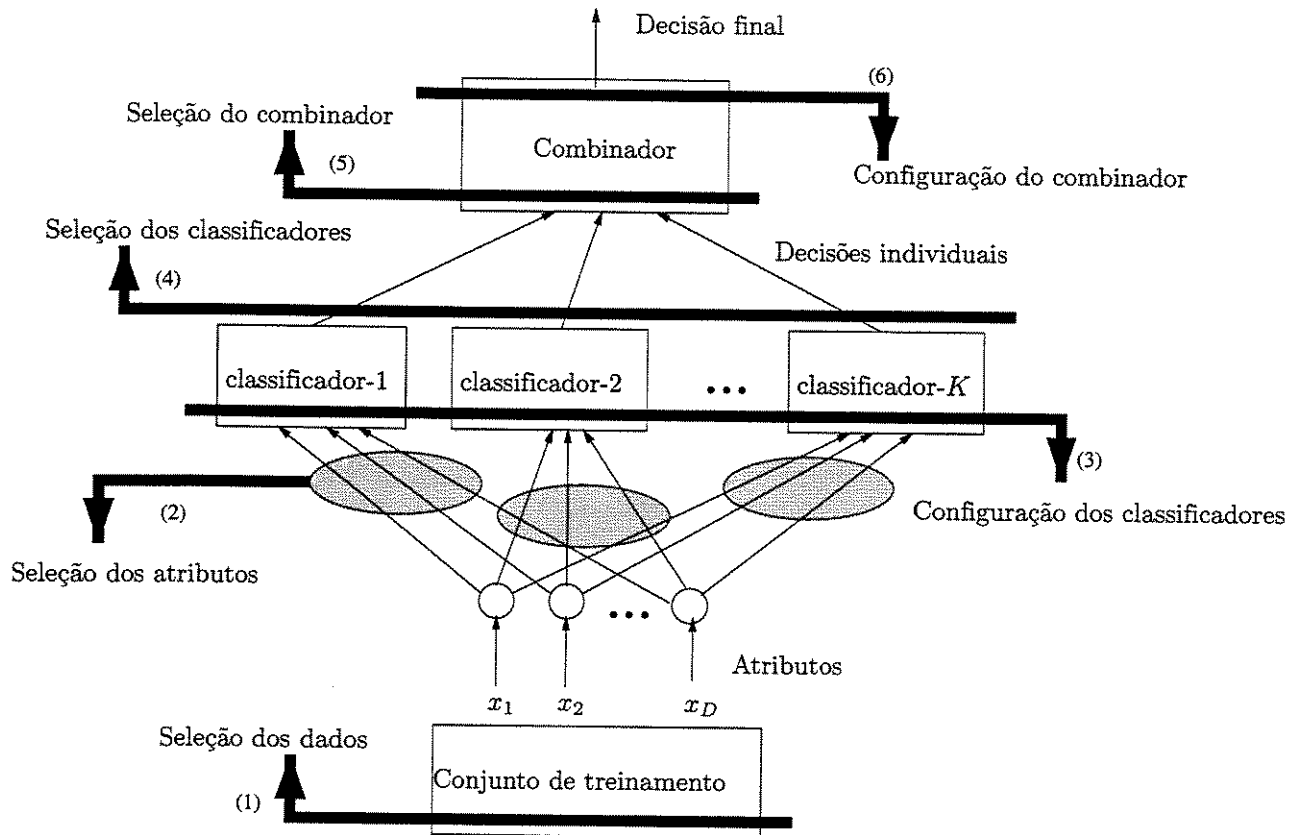


Fig. 7.1: Escopo de investigação de COHENN. As linhas e elipses cheias indicam os pontos de ajuste adaptativo (*hotspots*) configurados, de maneira integrada, ao longo do processo coevolutivo.

se obter ganhos compensatórios em termos de desempenho do modelo final. Tão importante quanto isso é descobrir quais pontos necessitam de pouco, ou mesmo nenhum, ajuste, de modo a se evitar desperdício de tempo e outros recursos computacionais.

Ademais, torna-se quase impossível prever com antecedência como as escolhas atribuídas aos pontos fixos passarão a influenciar as calibrações dinâmicas a serem feitas para os pontos maleáveis. Por exemplo, na abordagem proposta por Kucheva [259], destinada à seleção adaptativa de atributos para treinamento das redes-componentes (ver à página 161), a população de indivíduos selecionados para a próxima geração engloba uma parcela composta pelos cromossomos-filhos atuais com maior acuidade individual (maior grau de aptidão) e uma parcela composta pelos cromossomos-pais cujas respectivas redes apresentem o maior nível de acuidade quando em combinação. Para a definição dessa segunda parcela, adotou-se tão-somente o operador “Voto majoritário” (ver Apêndice B) para a fusão das decisões dos estimadores, algo que influenciou sobremaneira a qualidade dos modelos finais obtidos para as redes-componentes. Situações semelhantes envolvem várias outras abordagens de gênero evolucionário, dentre as quais aquela proposta por Harrald

e Kamstra [194], em que as redes-combinadoras são ajustadas por EP somente após os modelos-preditores terem sido configurados, separadamente, numa primeira fase, e a abordagem GAsel (apresentada à Seção 8.2), destinada à seleção (e combinação) adaptativa de modelos SVMs pré-configurados.

Uma das grandes desvantagens de se moldar independentemente ou seqüencialmente os vários *hotspots* está na perda da capacidade de exploração das possíveis interrelações entre eles. Em alusão a esse processo comumente adotado de desenho de *ensembles* em dois passos (i. e., primeiro gerar e depois combinar os componentes), Liu *et alia* [298] argumentam: “As possíveis interações entre elas [redes-componentes] não podem ser exploradas até o estágio de combinação. Não há, pois, *realimentação* do estágio de combinação para o estágio de projeto das redes individuais. É possível que algumas das redes projetadas independentemente não provenham contribuição significativa para todo o sistema”. Paradoxalmente, como apontado à página 158, os próprios autores só passaram a analisar as potencialidades dos três métodos de combinação investigados (“Média simples”, “Voto majoritário” e “Vencedor-leva-tudo”) *após* a fase de evolução (e determinação) das espécies de RNAs. Ou seja, a abordagem EENCL [298] proposta por eles não promoveu, em realidade, realimentações entre os estágios de combinação e projeto de redes-classificadoras.

Dada a sua natureza simbiótica e evolucionária, COHENN enfatiza tanto a *especialização* como a *harmonia* entre os vários *hotspots*. O preceito defendido aqui (e perquirido via extensas simulações computacionais) é o de que, ao se permitir que os vários pontos de ajuste possam exercer *influências mútuas* ao longo do processo adaptativo, maiores ganhos, em termos de eficiência (i. e., rapidez na convergência às melhores soluções) e de eficácia (i. e., grau de acuidade e generalização do modelo final), podem ser alcançados. A razão que justifica essa estratégia é a de que COHENN pode ser encarado como um modelo “generalizado” que congrega, num só arcabouço conceitual, as *virtudes complementares* exibidas por várias abordagens proeminentes destinadas à confecção de combinações efetivas de estimadores, abordagens estas cuja eficácia vem sendo atestada tanto teórica como experimentalmente — exemplos dessas abordagens são apresentados no decorrer da Seção 4.1.

7.2 COHENN: Propriedades e arquitetura

Como já frisado, COHENN é um modelo arquitetural de cunho inerentemente coevolutivo cooperativo, uma vez que segue de perto as diretrizes estabelecidas pelo *framework* conceitual de Potter & De Jong [374, 375]. Assim sendo, em COHENN, diferentes espécies passam a ter suas trajetórias evolutivas interrelacionadas de forma mutualística como meio de se gerar a solução final (no caso, *ensembles* de estimadores). Em consonância com o modelo hierárquico de SANE [335], COHENN exhibe dois níveis de evolução concorrentes — vide Fig. 7.2. No nível superior, encontra-se uma população de *descrições de ensembles*, ao passo que o nível inferior encerra uma população de *redes-componentes* e uma população de *redes-combinadoras*. Desse modo, COHENN promove acoplamento das partes quando da avaliação de desempenho dos protótipos de *ensembles* (para fins de cálculo dos valores de *fitness*) e desacoplamento das mesmas quando da sua configuração dinâmica e concorrente. Antes de esquadrihar os elementos de cada uma das populações e expor os passos do

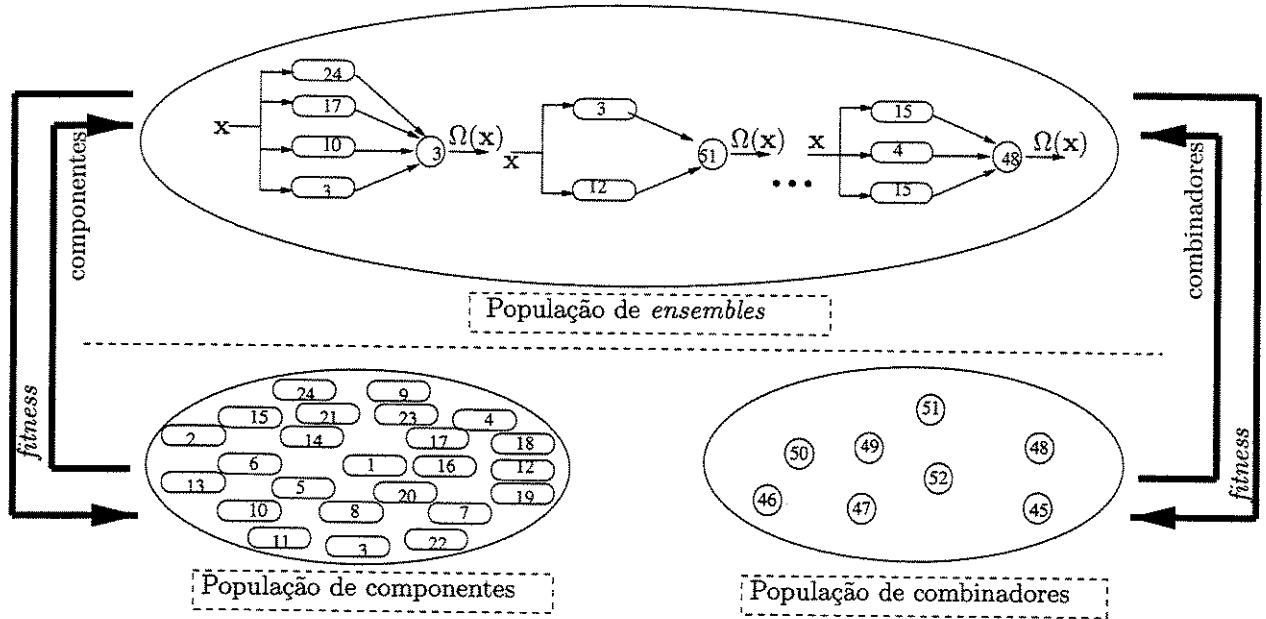


Fig. 7.2: Modelo arquitetural hierárquico COHENN. As três populações coexistentes evoluem concorrentemente, sendo que as do patamar inferior fornecem módulos para as composições do nível superior.

processo coevolucionário relativo a COHENN, passa-se, na sequência, a enumerar algumas das funcionalidades intrínsecas a esse modelo arquitetural.

7.2.1 Funcionalidades

O primeiro ponto a ser salientado diz respeito à questão mui importante de manutenção dos níveis de diversidade (ambigüidade) quando da etapa de criação das redes-componentes que integrarão os arquétipos de *ensembles* do nível superior. Conforme mencionado na Subseção 4.1.2, existem várias abordagens passíveis de serem adotadas para esse fim. Mediante o ajuste integrado dos *hotspots* (1), (2) e (3) indicados na Fig. 7.1, COHENN passa a encorajar a geração e a preservação constantes de diferentes padrões de generalização para as redes-componentes. Enquanto o ponto de ajuste (2) estimula variações (com possíveis sobreposições) nos conjuntos de atributos, o ponto de ajuste (1) incita variações (também com possíveis sobreposições) nos conjuntos de amostras para quando do treinamento individual de cada rede (ver a seguir). Dessa forma, diferentes padrões de generalização passam a ser induzidos via a decomposição progressiva e complementar do espaço de entradas, tanto em termos do número de dimensões que cada classificador “enxerga” como em termos das regiões em que eles se especializam. Sob esse último aspecto, COHENN passa a exibir funcionalidades similares àquelas promovidas pelas técnicas estatísticas de *Bagging* e *Boosting*, uma vez que também estimula a reamostragem aleatória e adaptativa dos dados amostrais de treinamento das máquinas — isso feito no curso do seu processo coevolutivo (em similaridade, portanto, com a abordagem evolutiva ERC [213]).

Complementando os efeitos viabilizados nos pontos anteriores, o ponto de ajuste (3) da Fig. 7.1 promove variações concernentes ao procedimento de indução das várias redes-componentes. Cada uma dessas redes pode ser refinada localmente mediante uma etapa de treinamento híbrido lamarckiano amparada na execução do algoritmo *Backpropagation* (Subseção 2.4.2) sobre seu conjunto de treinamento associado — este demarcado em razão do que tenha sido especificado nos pontos (1) e (2). A taxa de aprendizado de cada rede passa a ser um parâmetro determinado dinamicamente, haja vista ser codificada em seu próprio cromossomo. Ademais, parâmetros arquiteturais são também ajustados ao longo do processo evolutivo associado à população de redes-componentes: tanto os pesos sinápticos (entre as camadas de entrada e intermediária e entre as camadas intermediária e de saída) como o número e tipo de neurônios da camada intermediária de cada rede. O tipo (único) dos neurônios escondidos de cada rede é definido a partir da escolha dinâmica de sua função de ativação. Do mesmo modo que com a abordagem HCGA, com COHENN, pode ser possível se investigar qual(is) o(s) tipo(s) de neurônio(s) utilizado(s) tende(m) a ser mais indicados para lidar com a instância de problema em mãos. Caso se tenha um número apropriado de instâncias de problemas (quanto mais, melhor, em termos estatísticos), pode-se até levantar *indícios* sobre quais tipos de neurônios sejam mais adequados para a tarefa de classificação como um todo. Para esse propósito, basta estimar a frequência com que cada tipo de neurônio aparece nos arranjos das redes-componentes pertencentes aos modelos de *ensembles* vencedores finais de cada execução de COHENN. Outrossim, para o mesmo propósito, pode-se levar em conta o tipo de neurônio relativo à melhor rede-componente gerada ao longo de todo o ciclo evolutivo ou os tipos de neurônios presentes à população de redes-componentes da última geração.

Embora não mostrado explicitamente na Fig. 7.1, um quarto ponto de indução à diversidade dos padrões de erros das redes-componentes está embutido na fase de avaliação (atribuição do grau de aptidão) das descrições de *ensembles* da população superior. Como se verá adiante, uma parcela da função de *fitness* associada a essa população diz justamente respeito ao nível de diversidade inerente a cada protótipo de agregação de estimadores codificado por cada cromossomo da população. Por conta do mecanismo de *credit assignment* intrínseco ao paradigma coevolutivo cooperativo (ver Subseção 6.7.2), os valores de *fitness* das redes-componentes passam a depender diretamente daqueles atribuídos às descrições de *ensembles* em que participam como módulos, passando a haver, por conseguinte, um estímulo indireto para que tais redes-componentes se tornem cada vez mais não-correlacionadas em seus padrões de erro/generalização; ou seja, há estímulo para se tornarem cada vez mais especialistas em partes diferentes do problema. De acordo com essa mecânica, em vez de apelar para instrumentos de formação *explícita* de nichos e espécies (tais como os de compartilhamento de *fitness* adotados por EENCL [298] e pela abordagem de Lee *et al.* [281]), COHENN lança mão da sua própria natureza coevolucionária como instrumento de formação *implícita* dos nichos de atuação das redes-componentes. Conforme García-Pedrajas *et al.* salientaram, no contexto da abordagem correlata COVNET [168], “...na medida em que a combinação de diversos nódulos exibindo o mesmo comportamento e recebendo as mesmas entradas não irá produzir redes com um bom valor de *fitness*. Daí, não há necessidade de se calcular uma medida de compartilhamento de *fitness* para guiar o processo evolutivo”.

Assim como as abordagens GASEN [514, 515] e GASE1, COHENN também promove a seleção adaptativa dos módulos componentes — ver *hotspot* (4) à Fig. 7.1. Em vez de recorrer a um critério preestabelecido para

decidir quais redes-componentes manter e quais retirar — ver critérios sugeridos por Zhou *et al.* e Perrone & Cooper à Subseção 4.1.4 —, COHENN seleciona os melhores “times” de redes-componentes produzidos ao longo do processo evolutivo mediante a manutenção da população de descrições de *ensembles*. De um universo de *NIND.COMP* módulos compondo correntemente a população de redes-componentes, cada uma das descrições de *ensembles* do nível superior irá selecionar apenas *N.COMP* delas. As redes-componentes pertencentes às melhores descrições de *ensembles* encontradas ao longo da execução, por se beneficiarem dos altos graus de aptidão atribuídos a estas, tenderão a ter seu material genético preservado, gerando, por conseguinte, prole com características cada vez mais adaptadas às nuances do problema. Como forma de garantir *explicitamente* essa manutenção de blocos construtivos de alta qualidade, tanto as redes-componentes como a rede-combinadora (se for o caso) pertencentes à melhor *ensemble* atual passarão a integrar diretamente a próxima geração do ciclo evolutivo de suas populações⁽³⁾.

Além disso, COHENN viabiliza também um mecanismo alternativo destinado à *poda evolucionária* do número de elementos a compor as *ensembles*: a possibilidade de se gerar redes-componentes *nulas* (ou seja, contendo nenhum neurônio escondido). Destarte, em existindo na população de componentes redes dessa natureza, para que ocorra a poda de componentes basta que as descrições de *ensembles* passem a apontar para tais componentes nulos. Uma vez que tais componentes são avaliados de modo similar àqueles não-nulos, ao se observar o número de redes-componentes nulas a cada geração, pode-se ter uma noção de como COHENN está promovendo *parcimônia* de recursos (parâmetros livres) no decorrer de sua execução.

Por outro lado, em cada arquétipo de *ensemble* projetado por COHENN, há também a chance de se ter, simultaneamente, várias cópias de uma mesma rede-componente. Poder-se-ia encarar tal situação como um segundo subterfúgio de COHENN dirigido à poda de estimadores, visto que, muitas vezes, as várias cópias existentes podem ser substituídas, sem prejuízo, por uma única rede-componente. Contudo, existem situações em que o operador de combinação adotado lança mão de informações quantitativas — por exemplo, concernentes ao número de votos ou ao grau de suporte agregado às classes —, a fim de produzir a decisão final do grupo. Assim, pode ser que várias cópias de uma mesma rede (clones) estejam sendo usadas por uma única *ensemble* somente com o propósito de se prover maior “peso” às suas decisões frente às decisões de outros componentes do grupo. Nesse sentido, nas simulações reportadas na próxima seção, optou-se por não eliminar tais “redundâncias de classificadores” quando da interpretação fenotípica das *ensembles* resultantes.

Além das etapas de criação e seleção de componentes, COHENN também lida, de maneira adaptativa, com as tarefas de seleção e configuração do módulo combinador. Da mesma forma que parece não existir um único indutor capaz de apresentar o melhor desempenho para todos os domínios de problemas possíveis (em concordância com o que é apregoado pelos teoremas NFL [493]), parece não existir também um único esquema de combinação capaz de superar os demais em todas as ocasiões. Essa hipótese soa bem plausível uma vez que o efeito produzido pela atuação do combinador pode ser encarado, por si só, como a geração de um novo estimador cujo comportamento (i. e., padrão de generalização) passa a ser definido por meio

⁽³⁾ Caso exista mais de uma *ensemble* com melhor grau de aptidão na primeira geração, a escolha de qual delas salvar os componentes é feita aleatoriamente. Ao longo da execução, caso surjam descrições de *ensembles* com *fitness* igual ao da melhor atual, a decisão aleatória é feita somente entre as novas, como forma de evitar convergência prematura.

da fusão dos comportamentos dos estimadores-componentes. No que tange à tarefa de seleção adaptativa do algoritmo de combinação (*hotspot* (5) da Fig. 7.1), COHENN lança mão tanto de operadores fixos como ajustáveis — ver Subseção 4.1.5. No caso dos combinadores fixos, atualmente, a arquitetura adota um repertório composto por 44 operadores⁽⁴⁾, muitos dos quais representando variações (via configuração de parâmetros particulares) de um mesmo algoritmo de combinação. Uma descrição de alto nível desses 44 operadores é propiciada no Apêndice B, sendo essa lista passível de ser facilmente estendida com a introdução de novos operadores. No caso dos operadores ajustáveis, optou-se por implementá-los também como RNAs *feedforward*, haja vista a capacidade de aproximação universal apresentada por esses modelos [194]. Tais combinadores ajustáveis passam a ser projetados de modo automático no curso do processo coadaptativo existente entre a população de redes-combinadoras e as demais populações. Desse modo, COHENN pode ser visualizado como uma alternativa evolucionária para as abordagens *stacking* [450, 492] e ERNM [500].

Complementando os esforços empreendidos pelo engenho de busca genética (i. e., de caráter global), em COHENN, toda vez que um novo arranjo de *ensemble* mais adaptado que o melhor atual é encontrado, entra em ação um mecanismo de *busca local* que examina a possibilidade de existir, no elenco atual de combinadores (fixos + ajustáveis), algum que seja mais apropriado que o combinador corrente, ou seja, que incremente ainda mais o grau de aptidão dessa nova melhor *ensemble*. Em existindo, o índice correspondente a esse operador passa a substituir aquele associado ao combinador corrente. Ademais, de maneira similar ao que foi comentado para os tipos de neurônios escondidos, com COHENN, há também a possibilidade de se investigar, empiricamente, quais os operadores de combinação tendem a ser mais adequados para arcar com os requisitos impostos por uma certa classe de problemas de estimação. Esse tipo de funcionalidade pode municiar o projetista que não faz uso de métodos automáticos de construção de *ensembles* com a informação (heurística) interessante sobre que tipo de combinador usar (ou não), em princípio, quando do tratamento de problemas exibindo sobreposições em suas características — p. ex., instâncias de problemas envolvendo algum tipo de diagnóstico médico.

Finalmente, com COHENN, existe a possibilidade de se averiguar ainda mais o potencial por trás do argumento defendido por Yao & Liu [296, 504] de que o nível de informação embutido numa população em evolução será sempre igual ou superior àquele embutido num único indivíduo gerado pelo mesmo processo evolucionário. Como discutido na Seção 6.9, os autores advogam que outros indivíduos de uma população em evolução podem conter informação adicional útil que venha a se somar àquela apreendida pelo melhor indivíduo gerado, de sorte a que se incremente a capacidade de generalização do sistema de aprendizado como um todo. Nesse contexto, COHENN vai um passo além da estratégia investigada por Yao & Liu [296, 504] de combinar as redes neurais pertencentes à última geração do processo evolutivo EPNet para compor *ensembles* de RNAs evolutivas. Com COHENN, em vez de descartar todas as *ensembles* geradas ao longo do processo evolutivo e manter apenas a vencedora final, o projetista passa a ter à sua disposição a funcionalidade mui interessante de geração automática de *ensembles* de *ensembles* evolutivas. A Fig. 7.3 traz

⁽⁴⁾Tal repertório de combinadores fixos abrange exemplos de operadores trabalhando com informações provenientes dos componentes nos três níveis de resolução levantados por Xu *et al.* [497]: abstrato, em *ranking* e de confiança — ver Subseção 4.1.5.

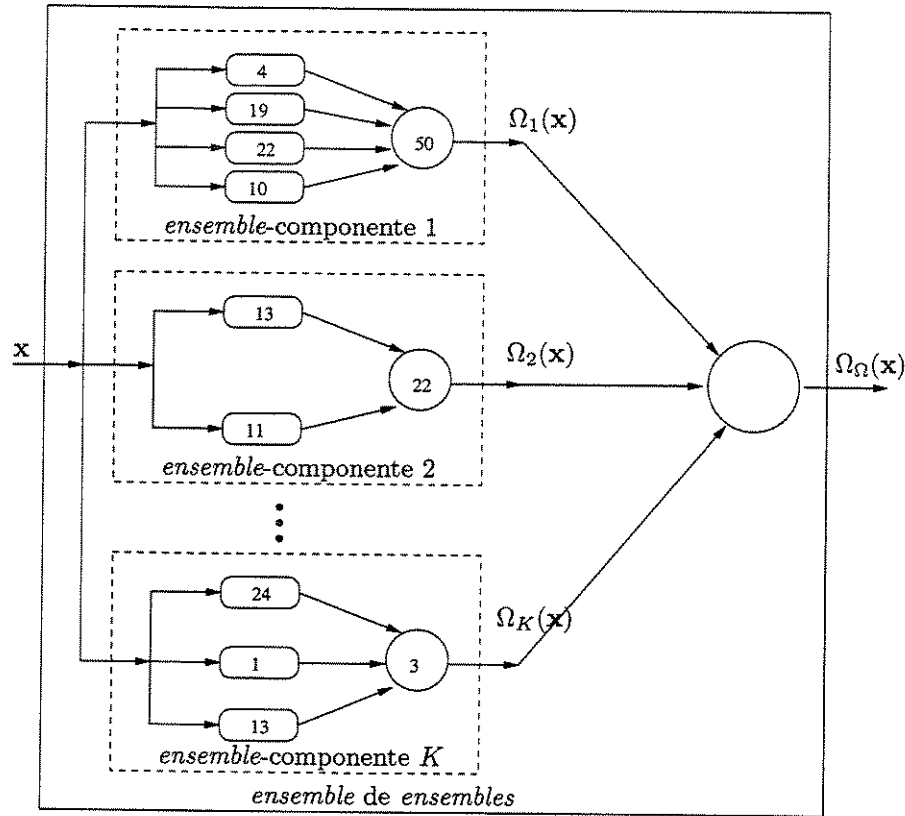


Fig. 7.3: Ensembles de ensembles evolucionárias.

um esboço desse recurso, em que K descrições de *ensembles* obtidas pelo processo coevolutivo passam a ser tratadas como módulos de maior granulosidade (denominadas *ensembles-componentes*) e passam a ser (não-)linearmente combinadas por um operador de combinação a ser arbitrado pelo projetista — provavelmente, um dentre os 44 operadores fixos do repertório disponível. Dois tipos particulares de *ensembles* de *ensembles* evolutivas foram consideradas num primeiro momento em COHENN. Enquanto no tipo (A) as *ensembles-componentes* são resgatadas diretamente da última geração da população de descrições de *ensembles* (ou seja, $K = N\_ENS$), no tipo (B), as *ensembles-componentes* são representadas pelas melhores *ensembles* de cada uma das MAX_GEN gerações do processo coevolutivo (ou seja, $K = MAX\_GEN$). Obviamente, outras configurações são passíveis de serem implementadas, tais como resgatar todas as *ensembles* evoluídas em *todas* as gerações ou resgatar as melhores *ensembles* pertencentes a cada um dos *clusters* oriundos da aplicação de um algoritmo de agrupamento qualquer (p. ex., *K-Means* [197], *Fuzzy C-Means* [52], ou mesmo uma rede neural com aprendizado não-supervisionado, como os mapas auto-organizáveis de Kohonen [198, 318]) sobre as descrições de *ensembles* da última geração.

7.2.2 Passos do processo coevolucionário e caracterização das populações

A Fig. 7.4 traz os passos do processo coevolucionário inerente ao modelo arquitetural COHENN. Após a etapa de geração das três populações, já envolvendo aqui um primeiro refinamento local das redes-componentes, o sistema entra num ciclo de operações, que alterna entre fases de reconfiguração das populações (mediante a aplicação dos operadores de variação bem como dos procedimentos de refinamento local) e de avaliação dos seus indivíduos (i. e., atribuição dos valores de *fitness*). No que segue, passa-se a caracterizar os principais elementos desse processo coadaptativo.

Iniciação das populações. As três populações são geradas inicialmente de forma aleatória, segundo uma distribuição uniforme sobre os valores permitidos para cada variável codificada, ou seja, de acordo com os tipos e intervalos definidos para os alelos de cada gene. Particularmente, com relação aos pesos sinápticos das redes-componentes e redes-combinadoras, recorreu-se à heurística discutida ao final da Subseção 2.3.3 (página 26) para a determinação do seu intervalo de possíveis valores. (A escolha dessa heurística em particular baseou-se em experimentações preliminares, onde ficou constatada a sua eficácia.) Para os experimentos realizados, $A = 2, 4$ (valor sugerido por Le Cun [386]) e o fator $\sqrt{n_s}$ passou a ser substituído por D , para as redes-componentes, e por $D + M \times N\_COMP$, para as redes-combinadoras, onde D é o número total de atributos de cada amostra do conjunto de treinamento, M denota o número de classes (inclusa a de rejeição) e N_COMP representa o número máximo de componentes de uma *ensemble* qualquer. Por sua vez, as taxas de aprendizado adaptativas das redes-componentes têm seu intervalo de possíveis valores definido como sendo $[0, 1]$.

Aspectos de codificação genotípica e interpretação fenotípica. A Fig. 7.5 ilustra como os indivíduos das três espécies são genotipicamente representados, indicando os tamanhos de seus blocos de genes. Cada descrição de *ensemble* guarda um conjunto de ponteiros sinalizando quais módulos recrutar como redes-componentes e qual operador usar (do repertório total de fixos e ajustáveis) como combinador. Como se vê, os cromossomos das redes-componentes e redes-combinadoras exibem codificações mescladas, obtidas pela justaposição de blocos de variáveis binárias, inteiras ou reais (ponto flutuante). Com relação ao número de saídas de uma rede-componente ou rede-combinadora, este é determinado de acordo com a codificação “1-of- M ” — ver Subseção 2.4.1. No que diz respeito ao número de entradas de uma certa rede-componente, este será definido de acordo com aqueles atributos que estejam ativados (valor “1”) em seu cromossomo. Ou seja, dentre D atributos possíveis, somente os ativados serão levados em consideração⁽⁵⁾. Dessa forma, mesmo que estejam sendo codificados $(D + 1) \times NMAX\_HID\_COMP$ valores de pesos de entrada (inclusive o de *bias*) em seu cromossomo, somente serão interpretados fenotipicamente aqueles relativos às dimensões de entrada ativadas. Além de diversidade, tal instrumento também estimula o uso parcimonioso de recursos quando da definição dos componentes. Já com relação ao número de entradas de uma rede-combinadora, este é dado por $(D + 1) + M \times N\_COMP$. No caso de se ter componentes nulos, as suas M saídas assumem valores nulos quando servindo de entrada aos combinadores ajustáveis, ao passo que, para os combinadores

⁽⁵⁾Não é considerado, porém, o caso em que todos os atributos estiverem desativados de uma só vez. Nessa circunstância, todas as dimensões de entrada serão interpretadas como ativadas.

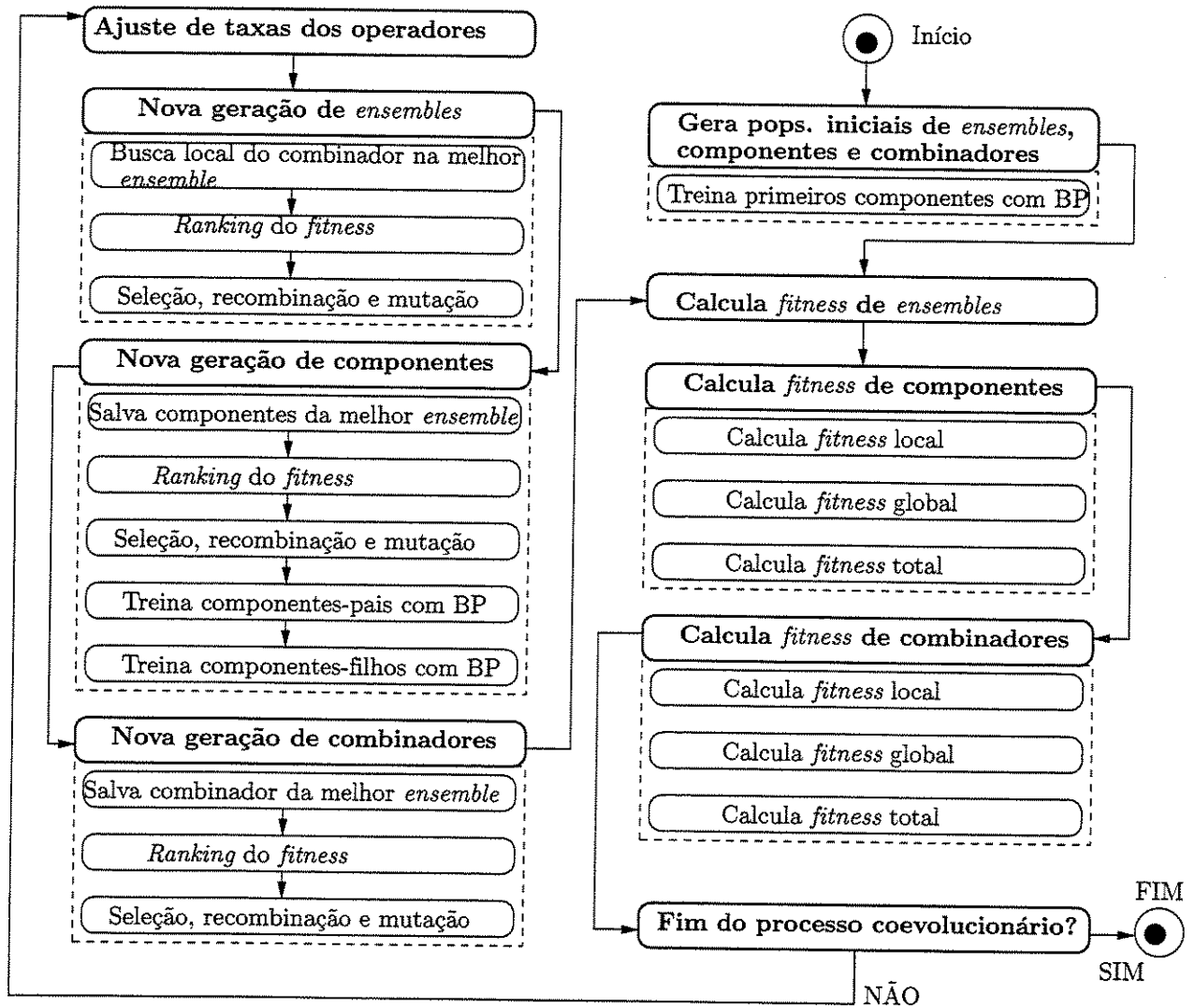


Fig. 7.4: Diagrama de blocos de alto nível indicando os principais passos do processo coevolucionário por trás de COHENN.

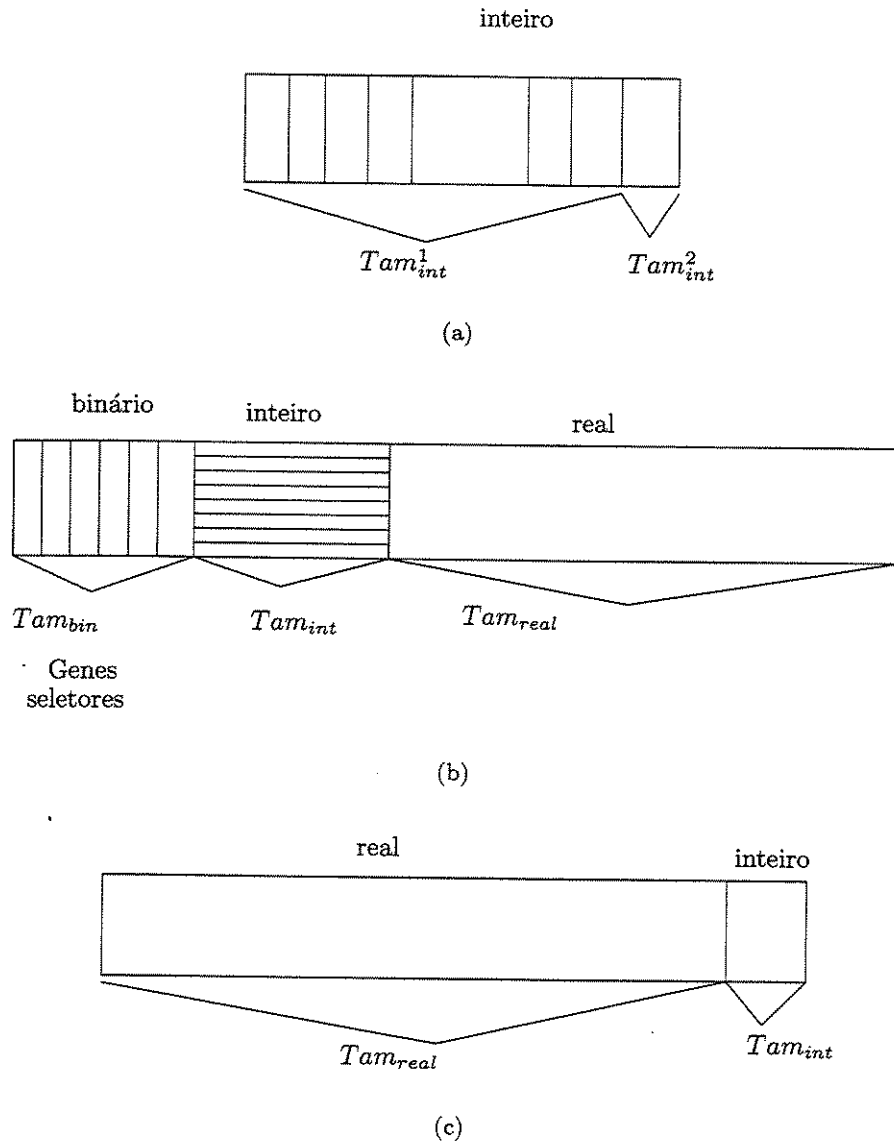


Fig. 7.5: Cromossomos dos três tipos de espécies em coevolução. (a) Descrição de *ensemble*: $Tam_{int}^1 = N\_COMP$ (índices dos componentes) e $Tam_{int}^2 = 1$ (índice do combinador); (b) Rede-componente: $Tam_{bin} = NMAX\_HID\_COMP + D$ (seleção do número de neurônios e dos atributos), $Tam_{int} = N_{tr} + 1$ (índices das amostras de treinamento e índice da função de ativação dos neurônios escondidos) e $Tam_{real} = (D + 1) \times NMAX\_HID\_COMP + (NMAX\_HID\_COMP + 1) \times M + 1$ (pesos sinápticos e taxa de aprendizado); (c) Rede-combinadora: $Tam_{int} = 1$ (índice da função de ativação dos neurônios escondidos) e $Tam_{real} = N\_HID\_COMB \times (D + (M \times N\_COMP) + 1) + (N\_HID\_COMB + 1) \times M$ (pesos sinápticos). D e M denotam o número de atributos e o número de classes (com a de rejeição), respectivamente.

fixos, tais valores são descartados. Por outro lado, para cada estimador não-nulo, as suas M saídas de ativação (i. e., graus de suporte às classes) passam a ser interpretadas via codificação bipolar pelas redes-combinadoras: valor “+1” indica a saída com maior ativação e valor “-1” é usado para as demais saídas. O intuito com essa transformação é o de facilitar o mapeamento a ser realizado pelas redes-combinadoras, já que estas não dispõem, na versão atual de COHENN, de etapas de treinamento local.

As funções relatadas na Tab. 2.1⁽⁶⁾ foram adotadas como funções de ativação tanto para as redes-componentes como para as redes-combinadoras, ficando a ordem de apresentação daquela tabela preservada para a definição dos seus índices associados. No que tange à delimitação do parâmetro relacionado ao número máximo de neurônios escondidos para as redes-componentes, $NMAX\_HID\_COMP$, lançou-se mão de uma variação da fórmula cunhada por Baum e Haussler [42], apresentada à Eq. (2.3):

$$NMAX\_HID\_COMP = 2 \times \left\lceil \frac{N_{tr} \times \alpha}{D + M} + 0,5 \right\rceil, \quad (7.1)$$

onde N_{tr} indica aqui o número de amostras de treinamento. No caso, foi adotado o valor 1/8 para a máxima tolerância admissível ao erro de generalização (α). (Testes preliminares com redes *feedforward* treinadas exclusivamente com *Backpropagation* para alguns dos problemas de classificação analisados no Apêndice C indicaram ser essa variação adequada para ser adotada por COHENN.) Ademais, para tais redes, foi resgatada a idéia estipulada pelos AGs estruturados [106] de se introduzir genes reguladores na sua codificação genotípica como forma de habilitar/desabilitar, dinamicamente, a expressão fenotípica dos seus neurônios escondidos. Assim, para cada um dos $NMAX\_HID\_COMP$ neurônios escondidos possíveis, somente serão considerados para efeito de composição da rede e propagação dos sinais à camada de saída aqueles cujos genes reguladores estiverem ativados (i. e., com valor “1”). Tal mecanismo encoraja o uso parcimonioso de recursos, como discutido na subseção anterior, viabilizando um instrumento de regularização adaptativa da complexidade das redes-componentes⁽⁷⁾. O fator “2” em (7.1) serve apenas para possibilitar uma maior “folga” para o intervalo de atuação desse mecanismo de poda. Já para as redes-combinadoras, como não há treinamento (nem poda evolucionária) destas, não se pôde utilizar a mesma receita teórica aventada por Baum & Haussler. Por esse motivo, o número (fixo) de neurônios escondidos de tais combinadores ajustáveis, NUM_HID_COMB , passou a ser dado por:

$$NUM\_HID\_COMB = \left\lceil \frac{D + N\_COMP}{M} + 0,5 \right\rceil. \quad (7.2)$$

Funções de aptidão. O processo de otimização conduzido em COHENN envolve a *minimização* de três funções de aptidão, cada uma relacionada a uma espécie. Nesse sentido, dizer que um certo cromossomo

⁽⁶⁾Embora a Tab. 2.1 aponte 15 funções, foram empregadas, em verdade, 16, já que a função Polinomial foi representada tanto com ordem dois como com ordem três. Ao contrário do que feito para a abordagem HCGA, que promove a evolução dos parâmetros associados às funções de ativação, com COHENN, tais parâmetros foram prefixados arbitrariamente — p. ex., nos experimentos, a largura (variância) das funções de base radial (Cauchy, Gaussiana, Multiquádrica e Multiquádrica inversa) foi dada por $\sigma = 0,5$.

⁽⁷⁾Ademais, o uso de tais genes reguladores, juntamente com o esquema de codificação mesclada, possibilita a geração de mapeamentos “um-para-um” entre genótipos e fenótipos, algo interessante para lidar com o problema de convenções competitivas atrelado à aplicação do operador de recombinação em RNAs *feedforward* (ver Seção 6.9).

apresenta um alto grau de aptidão relativa significa, em verdade, que ele apresenta um baixo *valor nominal de fitness*. Para a população de descrições de *ensembles*, a função de aptidão é determinada pela composição aditiva de três parcelas: uma referente ao seu grau de *acuidade*, outra referente ao seu grau de *parcimônia*, e uma terceira concernente ao grau de *diversidade* entre os padrões de generalização exibidos por seus componentes. Como o processo é de minimização, o termo de acuidade (*err*) é medido como sendo a taxa de erro total (de substituição + de rejeição) sobre o conjunto de treinamento total; quer dizer, tanto sobre as amostras da partição usada para o treinamento local das redes-componentes como sobre as amostras da partição usada para validação desse treinamento — a Tab. 7.3 apresenta os percentuais de amostras de cada partição adotados nos experimentos. Já a parcela relativa à parcimônia quantifica, em realidade, a complexidade da *ensemble* (*complex*), sendo dada por:

$$complex(\Omega_z) = \frac{\sum_k^{N\_COMP} \Upsilon(f_{kz})}{N\_COMP \times N_{tr}},$$

onde $\Upsilon(f_{kz})$ diz respeito ao número de pesos sinápticos (interpretados fenotipicamente) do k -ésimo componente integrando a z -ésima descrição de *ensemble*. Por sua vez, como medida do grau de diversidade intrínseco a uma descrição de *ensemble* qualquer, foi adotado o critério da média dos valores do coeficiente de correlação serial de Pearson [267, 281] aplicado aos pares de seus componentes⁽⁸⁾. Contudo, como o processo é de minimização, em vez dessa parcela ser calculada com base no inverso desse coeficiente (como colocado à Tab. 4.1), ela passa a ser medida em direta proporção a este — no lugar da diversidade, está-se mensurando, portanto, o nível de similaridade (*simil*). Em suma, o valor nominal de *fitness* da z -ésima descrição de *ensemble* Ω_z passa a ser dado pela seguinte combinação convexa:

$$fit_e(\Omega_z) = \lambda_d \times simil(\Omega_z) + \lambda_p \times complex(\Omega_z) + (1 - \lambda_d - \lambda_p) \times err(\Omega_z).$$

Para o caso excepcional de se gerar uma *ensemble nula* (ou seja, uma descrição apontando somente para componentes nulos), o seu valor nominal de *fitness* foi arbitrado como sendo “1,0”, pois, enquanto ela apresenta a menor complexidade possível, as suas taxas de acerto e diversidade são nulas.

Em se tratando de redes-componentes e redes-combinadoras, a determinação dos seus valores nominais de *fitness* baseia-se em duas parcelas complementares, uma de caráter local (acuidade individual) e a outra de natureza global (acuidade coletiva). Assim, o valor nominal de *fitness* total do k -ésimo elemento da população de redes-componentes, f_k , é dado por:

$$fit_{tot}(f_k) = \lambda_{COMP} fit_{glob}(f_k) + (1 - \lambda_{COMP}) fit_{loc}(f_k),$$

ao passo que o valor nominal de *fitness* total do l -ésimo cromossomo da população de redes-combinadoras, cb_l , é dado por:

$$fit_{tot}(cb_l) = \lambda_{COMB} fit_{glob}(cb_l) + (1 - \lambda_{COMB}) fit_{loc}(cb_l),$$

com λ_{COMP} e λ_{COMB} indicando parâmetros de controle cujos valores devem ser estabelecidos *a priori*.

⁽⁸⁾No caso de se ter componentes nulos, estes são desconsiderados nesse cálculo.

No caso das redes-componentes, a parcela de *fitness* local julga quão bem cada estimador é capaz de lidar por si só com a tarefa de classificação como um todo. Para esse propósito, cada indivíduo dessa população passa a ser avaliado após o seu treinamento local sobre um conjunto de validação à parte. O usuário de COHENN tem atualmente disponível para escolha quatro métricas que quantificam o grau de acuidade individual dessas redes, quais sejam, a taxa de erro, a medida-E (ver Seção A.1), a função-custo EQM (Eq. (2.5)) e a função-custo de entropia cruzada (Eq. (2.6)), cada uma explorando aspectos distintos de acuidade. Para o caso de redes-componentes nulas, à parcela de *fitness* local destas é atribuído o valor “1,0”, por razão semelhante àquela levantada anteriormente para as *ensembles* nulas.

Já para as redes-combinadoras, a parcela de *fitness* individual passa a ser delimitada em conformidade com o seguinte procedimento. São selecionadas, primeiramente, as *NUM_BEST_ENS* melhores descrições de *ensembles* da geração atual. Para cada um desses arquétipos, passa-se a substituir o seu combinador por cada uma das redes-combinadoras da geração atual e a se calcular as taxas de erro dos novos arranjos obtidos. O *fitness* local de cada rede-combinadora será dado pela média das taxas de erro dos *NUM_BEST_ENS* arranjos virtuais em que participou como combinador⁽⁹⁾. No caso excepcional de que uma dessas melhores *ensembles* seja nula, a taxa de erro arbitrada foi também de “1,0”, haja vista o papel a ser realizado pelo combinador, nessas circunstâncias, ser nulo.

Com relação à parcela de *fitness* global, que tenta dar uma indicação de quão bem um certo módulo (componente ou combinador) trabalha em equipe, ou seja, coopera com os demais módulos para que o desempenho do grupo seja o melhor possível, há dois esquemas simples de parceria passíveis de serem adotados atualmente em COHENN. O primeiro (esquema I) é de gênero guloso, pois calcula o *fitness* global de cada módulo com base somente na melhor descrição de *ensemble* corrente de que participa. O segundo (esquema II), por sua vez, é de escopo mais abrangente, já que se baseia na média de todas as descrições de *ensembles* de que é integrante. É pertinente ressaltar que, ao contrário do que é feito para a parcela local de *fitness* das redes-componentes nulas, a parcela global destas é calculada de maneira indistinta do que aquela feita para não-nulas. No caso daqueles módulos que não participam de nenhuma descrição de *ensemble* da geração corrente, o seu valor de *fitness* global passa a ser dado pela média dos valores de *fitness* global dos módulos que integram pelo menos uma descrição de *ensemble*. Isso vale tanto para cromossomos-pais como filhos, uma vez que esses últimos realmente não podem integrar nenhuma *ensemble* até a próxima geração. A razão por trás dessa decisão é a de que não é justo penalizar as redes (componentes ou combinadoras) que ainda não tiveram a chance de serem avaliadas globalmente. Uma possível alternativa, ainda não implementada, para esse caso seria a de se apelar para uma memória individual contendo os últimos (digamos, *N_LAST*) valores de *fitness* de cada cromossomo — para os cromossomos-filhos, esses valores seriam herdados dos pais. A média desses valores seria o seu *fitness* global atual, caso não participasse de nenhuma descrição de *ensemble*. Esse artifício guarda similaridade com o conceito de *life-time fitness evaluation*, proposto por Paredis [358] e discutido *in passant* à Subseção 6.7.1. Uma das desvantagens, porém, em adotá-lo é a da possibilidade de se introduzir mais carga computacional ao sistema.

⁽⁹⁾Vale frisar que esse procedimento é empregado para todas as redes-combinadoras, pais e filhas, mesmo para aquelas redes que já pertençam a uma das *NUM_BEST_ENS* melhores descrições de *ensembles* atuais.

Operadores de variação. Assim como a abordagem HGT-GAME (ver Seção 8.3), COHENN emprega, em suas três populações, (i) o algoritmo de amostragem universal estocástica (SUS) [33], (ii) um operador de rescalamento dos valores nominais de aptidão baseado em *ranking* linear, e (iii) a idéia de intervalo geracional (definido pelo parâmetro *GGAP*) para quando da seleção dos indivíduos (dentre pais e filhos atuais) a compor a próxima geração. Uma vez definidos os genitores da próxima geração, estes passam a ter seu material genético recombinado com o propósito de concepção da nova prole. Uma vez que os cromossomos das redes-componentes e redes-combinadoras apresentam codificações mescladas, para cada um dos blocos de variáveis de mesmo tipo (binário, real ou inteiro) é aplicado um operador de *crossover* em separado. Enquanto para as partes binárias e inteiras o operador *crossover* de dois pontos de corte (estes definidos aleatoriamente) foi adotado, para as partes reais, apelou-se para o operador *crossover* aritmético. No caso de blocos com um único gene, não existe recombinação das partes, somente a mutação destas⁽¹⁰⁾. Para blocos de genes inteiros ou binários, o operador mutação simples foi utilizado, ao passo que, para blocos de variáveis reais, apelou-se para um operador de mutação gaussiana especial proporcionado pelo AG estendido conhecido como “*Breeder GA*”, este proposto por Mühlenbein & Schlierkamp-Voosen [341]. A escolha de tais operadores, em particular, teve como base experimentações prévias conduzidas em problemas de otimização de menor complexidade, em que ficou evidenciada a sua eficácia e melhor desempenho frente a outros operadores-candidatos.

As taxas relativas aos operadores de recombinação e mutação são *adaptativas* para as três espécies. A correção dos valores atuais dessas taxas obedece a dois critérios complementares. O primeiro deles diz respeito ao nível atual de diversidade do material genético da população, ou seja, monitora se a população está apresentando problemas de convergência prematura ou não. A sua mecânica é de fácil compreensão: quando o grau de diversidade estiver muito elevado, os valores das taxas serão corrigidos para baixo (promovendo, portanto, maior intensificação), enquanto no caso de se ter níveis precários de diversidade, os valores das taxas serão incrementados (introduzindo material genético novo). O fator de ajuste ao valor das taxas adotado para esse critério fundamentou-se naquele sugerido por Srinivas & Patnaik [440]⁽¹¹⁾. Segundo tal fator de correção, as novas taxas de mutação e recombinação (aqui representadas genericamente por p_m e p_c) passam a ser calculadas como:

$$p_m = k_1 \frac{MUT\_MAX\_RATE}{|f_{min} - f_{med}|}$$

e

$$p_c = k_2 \frac{XOVER\_MAX\_RATE}{|f_{min} - f_{med}|},$$

onde *MUT_MAX_RATE* e *XOVER_MAX_RATE* representam, genericamente, os máximos valores das taxas para cada espécie (ver Tabs. 7.2 a 7.4), $k_1, k_2 \leq 1$ são duas constantes usadas somente para que os

⁽¹⁰⁾Por esse motivo, nesses casos, a taxa de mutação de tais blocos é o triplo do valor atual, como forma de se introduzir maior variabilidade.

⁽¹¹⁾No caso de COHENN, porém, o fator de correção é aplicado uniformemente a todos os membros da população, e não de maneira individual (ou seja, para cada indivíduo que venha a sofrer mutação ou para cada par de indivíduos escolhidos para recombinação), como proposto por Srinivas & Patnaik [440].

valores das taxas fiquem em $[0, 1]$, e f_{min} e f_{med} representam, respectivamente, o valor nominal de aptidão rescalado do melhor indivíduo e a média dos valores nominais de aptidão rescalados de todos os indivíduos para cada população corrente.

O segundo critério sinaliza se o grau de aptidão do cromossomo mais apto de cada população vem melhorando (ou seja, seu valor nominal de *fitness* vem decaindo) ou não ao longo das últimas gerações. Para verificar quanto a isso, calcula-se a média do valor nominal de *fitness* dos melhores cromossomos das últimas *MEMO* (Tab. 7.1) gerações e contrasta-se o resultado com o valor nominal de *fitness* do melhor cromossomo atual. Se a diferença percentual entre tais valores for maior que um limiar v prefixado, então não é feita nenhuma correção às taxas, conforme esse critério. Caso contrário, cada taxa tem seu valor incrementado por um fator proporcional ao produto do seu valor corrente pela diferença absoluta entre a diferença percentual encontrada e o limiar prefixado.

Treinamento híbrido. Como já mencionado, às redes-componentes de cada geração (pais e filhas) é aplicada uma execução individual do algoritmo *Backpropagation*, com a taxa de aprendizado η sendo customizável adaptativamente para cada rede. Assim como HGT-GAME, COHENN também adota um procedimento de ajuste dinâmico do número de iterações do algoritmo *Backpropagation* para quando do treinamento local (lamarckiano) dos cromossomos-pais. A fórmula de ajuste é a mesma daquela apresentada no pseudo-código da Fig. 8.12. Já o número de iterações para o ajuste dos pesos dos cromossomos-filhos se mantém constante ao longo da execução. Para as redes-componentes que participam da melhor descrição de *ensemble* atual, o resultado do seu aprendizado local só será efetivado (quer dizer, os novos valores para os seus pesos sinápticos só serão verdadeiramente incorporados ao cromossomo) caso o valor nominal de *fitness* da melhor *ensemble* seja de alguma forma decrementado. Não sendo o caso, os valores antigos dos pesos serão recuperados, bem como o valor nominal de *fitness* da melhor estrutura de *ensemble* atual.

Critério de parada. O ciclo de execução será interrompido quando um dos seguintes critérios for satisfeito: ou o número de gerações é maior que *MAX_GEN* ou o grau de acuidade da melhor descrição de *ensemble* atual atingiu a marca dos 100% de acerto, correspondendo, pois, a uma parcela de erro (do valor nominal de *fitness*) nula.

Decisão sobre a descrição de *ensemble* vencedora final. Finalizado o ciclo evolutivo, passa-se a avaliar mais uma vez todas as descrições de *ensemble* referentes à última geração, considerando, nessa situação, somente o desempenho das mesmas em termos de acuidade, ou seja, descartando-se as parcelas de parcimônia e diversidade. Aquela(s) descrição(ões) que obtiver(em) o melhor desempenho será(ão) eleita(s) “vencedora(s) final(is)” relativa(s) a todo o processo coadaptativo⁽¹²⁾.

⁽¹²⁾Portanto, a vencedora de todo o processo (denominada COHENN1) não é necessariamente a melhor descrição de *ensemble* da última geração do ciclo evolutivo (esta referida daqui em diante como “vencedora em potencial”). A razão dessa decisão está no fato de ter sido priorizado o critério da acuidade, em detrimento dos demais, para fins de análise de eficácia da abordagem – ver a seguir.

Tab. 7.1: Parâmetros de configuração comum a todas as populações.

Parâmetro	Interpretação	Valor
<i>MAX_GEN</i>	Número máximo de gerações do processo coevolucionário	20
<i>GGAP</i>	Percentual de novos indivíduos a serem criados por geração	90%
<i>NIND_COMB_f</i>	Número de operadores de combinação fixos	44
<i>MEMO</i>	Tamanho da janela de gerações passadas usada para fins de correção das taxas dos operadores genéticos	5
ν	Limiar percentual adotado para o propósito de correção das taxas dos operadores genéticos	20%

Tab. 7.2: Parâmetros de configuração da população de descrições de *ensembles*.

Parâmetro	Interpretação	Valor
<i>NIND_ENS</i>	Número de indivíduos da população	12
<i>N_COMP</i>	Número máximo de componentes formando uma <i>ensemble</i>	4
<i>XOVER_ENS_RATE</i>	Taxa inicial (e máximo valor) do operador de recombinação	70%
<i>MUT_ENS_RATE</i>	Taxa inicial (e máximo valor) do operador de mutação	10%
λ_p	Peso da parcela que mede a complexidade de uma <i>ensemble</i> sobre o seu <i>fitness</i> total	10%
λ_d	Peso da parcela que mede a diversidade de uma <i>ensemble</i> sobre o seu <i>fitness</i> total	20%

7.3 Resultados experimentais

Dois são os tipos de análise encaminhados neste espaço como forma de se aferir as potencialidades inerentes à abordagem COHENN. O primeiro tipo tenta apurar o desempenho do modelo arquitetural proposto em termos de sua eficiência computacional. Nesse âmbito são também examinadas algumas estatísticas interessantes, como a frequência relativa dos tipos de funções de ativação e a frequência relativa dos tipos de operadores de combinação fixos nas *ensembles* vencedoras finais do processo. Quanto ao segundo tipo de análise, passa-se a considerar aspectos concernentes à eficácia da proposta, avaliando, pois, o seu desempenho em termos dos critérios de acuidade e generalização. Aqui também há interesse em se confrontar o desempenho exibido pela nova abordagem frente àquele apresentado por outros métodos de AM supervisionado.

Os valores dos parâmetros de configuração de COHENN adotados para os experimentos conduzidos são mostrados nas Tabs. 7.1 a 7.5. Tais valores são comuns a todas as simulações realizadas sobre as 18 primeiras instâncias de problemas descritas nas Tabs. C.1 e C.2. (O elevado número de atributos da instância *MCR* inviabilizou a execução do protótipo da arquitetura para essa instância, por conta da alta demanda por recursos computacionais, fazendo com que uma única execução levasse mais de quatro ou cinco dias de execução.) Mesmo embora se saiba que uma calibração fina de tais parâmetros, em consonância com cada instância de problema em vista, possa vir a incorrer em um possível aumento de desempenho da técnica,

Tab. 7.3: Parâmetros de configuração da população de redes-componentes.

Parâmetro	Interpretação	Valor
<i>NIND_COMP</i>	Número de indivíduos da população	24
<i>XOVER_COMP_RATE</i>	Taxa inicial (e máximo valor) do operador de recombinação	70%
<i>MUT_COMP_RATE</i>	Taxa inicial (e máximo valor) do operador de mutação	10%
$\tau_1(0)$	Número (máximo) de épocas de treinamento dos componentes-pais	5
τ_2	Número (fixo) de épocas de treinamento dos componentes-filhos	10
<i>NUM_KERNEL_COMP</i>	Número de tipos de função de ativação	16
<i>NMAX_HID_COMP</i>	Número máximo de neurônios escondidos das redes-componentes	Eq. (7.1)
<i>NBITS_SEL_HID_COMP</i>	Número de genes seletores	<i>NMAX_HID_COMP</i>
λ_{COMP}	Peso do <i>fitness</i> global sobre o <i>fitness</i> total	60%
<i>LOC_FIT_MEAS</i>	Métrica usada para avaliação local	Medida-E
<i>PARTNERSHIP_COMP</i>	Esquema de parceria	Esquema II
$size_{tr}, size_v$	Tamanho das partições para treinamento e validação das redes-componentes em relação ao conjunto de treinamento inteiro, cujo total de amostras, N_{tr} , é dado de acordo com o problema	70%, 30%

Tab. 7.4: Parâmetros de configuração da população de redes-combinadoras.

Parâmetro	Interpretação	Valor
<i>NIND_COMB_a</i>	Número de indivíduos da população	8
<i>XOVER_COMB_RATE</i>	Taxa inicial (e máximo valor) do operador de recombinação	80%
<i>MUT_COMB_RATE</i>	Taxa inicial (e máximo valor) do operador de mutação	20%
<i>NUM_KERNEL_COMB</i>	Número de tipos de função de ativação	16
<i>NUM_HID_COMB</i>	Número (fixo) de neurônios escondidos das redes-combinadoras	Eq. (7.2)
<i>NUM_BEST_ENS</i>	Tamanho do conjunto de melhores descrições de <i>ensembles</i> atuais usado para calcular os valores de <i>fitness</i> local das redes-combinadoras	3
λ_{COMB}	Peso do <i>fitness</i> global sobre o <i>fitness</i> total	60%
<i>PARTNERSHIP_COMB</i>	Esquema de parceria	Esquema II

Tab. 7.5: Valores adotados para os parâmetros dos operadores fixos — vide Apêndice B.

Parâmetro	Valor
τ_P (Pluralidade)	10%
τ_{Bayes} (<i>Bayesiano</i> ingênuo)	50%
τ_{BKS} (BKS)	50%
τ_{DSA} (Dempster-Shafer-A)	50%
τ_{DSB} (Dempster-Shafer-B)	50%
τ_{seq} (Sequel)	50%

optou-se, por razões de uniformidade quando da análise de sua eficácia, em mantê-los fixos segundo uma configuração arbitrária, não necessariamente ótima. O mesmo procedimento foi adotado por García-Pedrajas *et alia* [168] e por Liu *et alia* [298, 299]. Vale atentar para o fato de que, para alguns parâmetros críticos, como o número máximo de gerações, o número máximo permitido de componentes para uma descrição de *ensemble* qualquer e os tamanhos das populações, os valores atribuídos foram, propositadamente, de baixa magnitude⁽¹³⁾. O objetivo com isso foi o de poder explorar ao máximo a capacidade da arquitetura quando tendo que operar num nível basal, ou seja, com poucos recursos computacionais à disposição. De uma outra forma, também em prol da análise de eficácia da abordagem, os parâmetros λ_d e λ_p passaram a assumir valores de baixa magnitude, ficando seus critérios associados preteridos em favor da parcela de acuidade. No mais, vale ressaltar que os resultados reportados foram obtidos com base no procedimento de validação cruzada estratificada (com número de *folds* igual a 10) destinado à definição das partições de treinamento (inclusa aqui a de validação local dos componentes) e teste — ver Seção A.2.

7.3.1 Análise de eficiência e frequencial

As Figs. 7.6 e 7.7 ilustram, de forma pictórica, exemplos de como se deu o processo de convergência à solução vencedora em potencial efetuado por COHENN quando lidando com os 18 problemas de classificação considerados⁽¹⁴⁾. Cada um desses gráficos faz alusão a uma partição qualquer (escolhida de forma discricionária) das 10 produzidas pelo método de validação cruzada para cada problema. Nessas ilustrações, além da taxa de erro relativa à melhor descrição de *ensemble* corrente, duas outras variáveis têm suas trajetórias evolutivas esmiuçadas. A primeira diz respeito à taxa média de erro da população atual de descrições de *ensembles*. A segunda se refere à taxa de erro do *oráculo* [205, 281]. O papel dessa abstração é o de indicar, para cada amostra sob análise, se há na população atual de redes-componentes algum estimador capaz de produzir a saída correta. Serve, portanto, como um bom “termômetro” para monitorar tanto o grau de

⁽¹³⁾Uma vez que as populações são de tamanho moderado, optou-se por fixar valores iniciais relativamente altos para as taxas de mutação, como meio de assegurar a possibilidade de manutenção freqüente dos níveis de variabilidade do material genético. O tamanho bem reduzido da população de redes-componentes (*NIND_COMB_a*) tem um motivo em particular: possibilitar uma disputa “leal” entre os tipos de operadores de combinação para quando da análise da freqüência dos operadores fixos — ver a seguir. Valores elevados para esse parâmetro poderiam prejudicar tal análise.

⁽¹⁴⁾Os experimentos foram conduzidos mediante a execução da ferramenta Matlab (Versão 6.0) em estações de trabalho Sun-UltraSparc (256MHz 512MB RAM) disponíveis em rede.

Tab. 7.6: Algumas medidas de desempenho. *TE*: tempo médio (em minutos) da execução de COHENN sobre uma dada partição gerada pelo procedimento de validação cruzada, considerando todo o ciclo coevolutivo e a posterior avaliação da última população de descrições de *ensembles*; *IRTE*: índice relativo de tempo de execução (ver Eq. (7.3)); *NG*: número médio de gerações necessárias para encontrar o protótipo de *ensemble* vencedor em potencial; e *NC*: número médio de componentes não-nulos das *ensembles* vencedoras finais. A ordem das instâncias de problemas segue aquela da Tab. C.2. Legenda: “DP”: desvio padrão.

Instância	TE		IRTE	NG		NC	
	Média	DP		Média	DP	Média	DP
1	1,16	0,84	0,12	8,3	4,67	3,4	0,84
2	9,96	1,5	0,03	12,5	2,46	3,73	0,64
3	34,86	11,96	0,15	16,5	3,13	4	0
4	6,93	1,26	0,05	14,7	3,94	4	0
5	3,21	0,49	0,02	7,3	3,83	3,2	0,79
6	15,76	11,79	0,17	15,1	4,75	4	0
7	3,55	0,64	0,14	8	4,028	2,9	0,7
8	11,94	1,47	0,20	11,6	5,32	3,76	0,63
9	2,73	0,45	0,06	8,8	5,27	2,43	0,57
10	2,89	0,47	0,02	6,6	2,72	3,7	0,54
11	0,27	0,092	0,03	2,9	1,37	3,65	0,47
12	2,90	0,49	0,26	12,8	4,34	3,8	0,42
13	1,81	0,83	0,16	10,7	4,54	3,9	0,32
14	4,19	0,66	0,02	10,9	6,17	3,05	0,68
15	2.050,2	239,41	0,90	3,3	4,4	4	0
16	9,43	1,61	0,04	16,2	3,29	3,5	0,85
17	0,73	0,29	0,02	2,9	1,10	3,9	0,32
18	3,37	0,52	0,13	10	6,51	3,45	0,68

acuidade como o grau de ambigüidade do conjunto de redes-componentes.

É interessante notar, primeiramente, que, em muitos dos casos, a melhor configuração de *ensemble* em potencial é logo encontrada nas primeiras gerações do ciclo coadaptativo. Por exemplo, para a instância de problema *Thyroid*, tal constatação fica patente para a partição em questão (Fig. 7.7(f)). Observando a Tab. 7.6, pode-se verificar que, para todas as instâncias de problemas, o número médio de gerações necessárias para encontrar a *ensemble* vencedora em potencial (*NG*) ficou abaixo do limite imposto por *MAX_GEN*. Esse resultado sugere, fortemente, que o maquinário por trás de COHENN consegue rapidamente produzir e localizar, já nas primeiras populações, protótipos de combinação bem efetivos para lidar com a tarefa de estimação em vista. Em alguns casos, como atestam os exemplos para as instâncias *Iris* (Fig. 7.7(b)), *Monk3* (Fig. 7.7(d)) e *Wine* (Fig. 7.7(h)), COHENN vai um pouco além, conseguindo projetar, em poucas iterações, vencedoras em potencial cujas taxas de erro são nulas, algo que faz abortar o seu ciclo coevolutivo.

Um segundo ponto passível de ser comprovado diz respeito à capacidade exibida freqüentemente por COHENN de preservação constante dos dois níveis de diversidade de interesse nesse estudo. No que

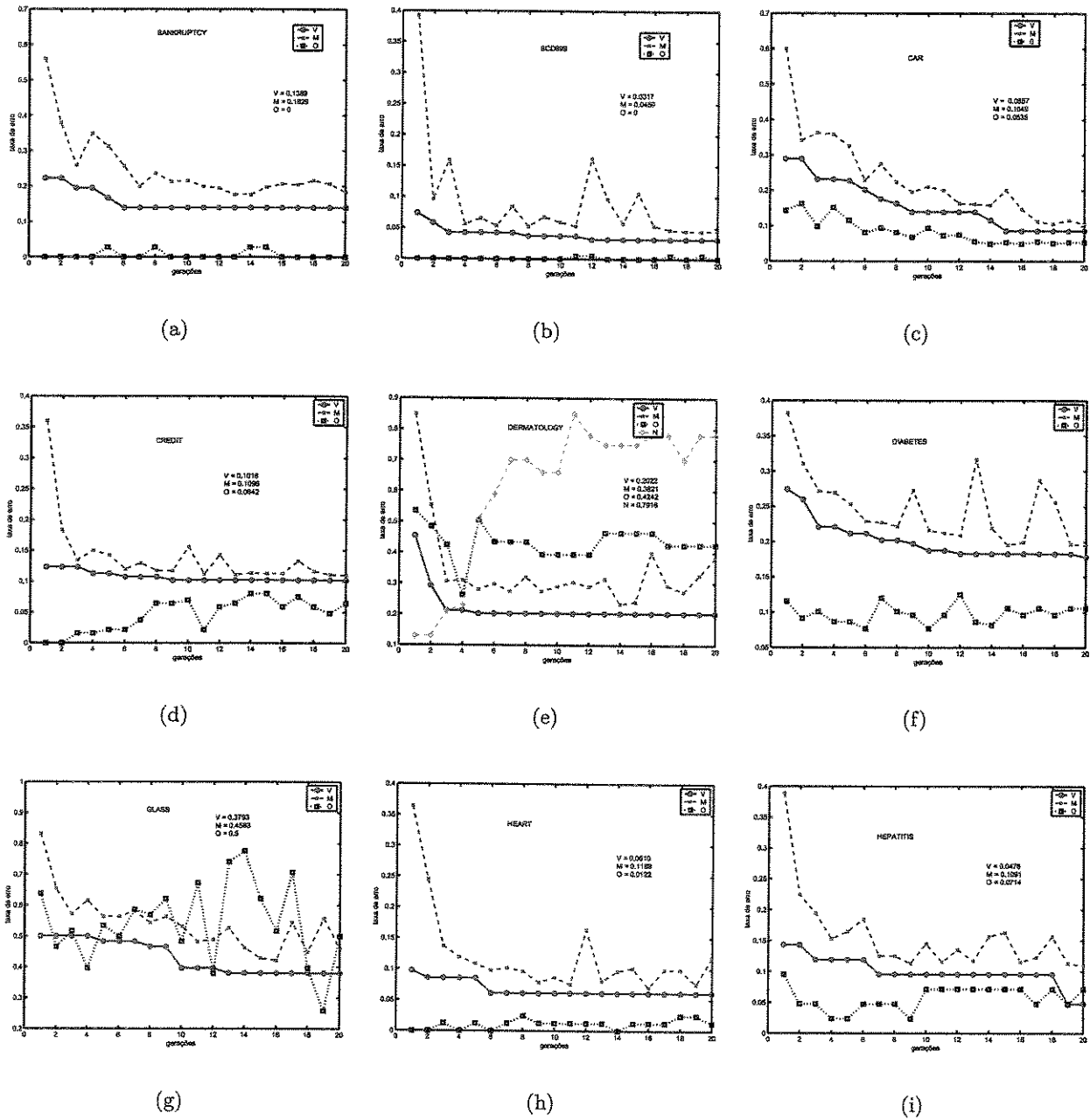


Fig. 7.6: Processo de convergência de COHENN para as nove primeiras instâncias de problemas, segundo a ordem estabelecida na Tab. C.2. “V” alude à taxa de erro da melhor descrição de *ensemble* da geração atual, “M” representa a taxa média de erro da população corrente de descrições de *ensembles*, “O” denota a taxa de erro do oráculo e “N” indica a taxa de componentes nulos na geração atual.

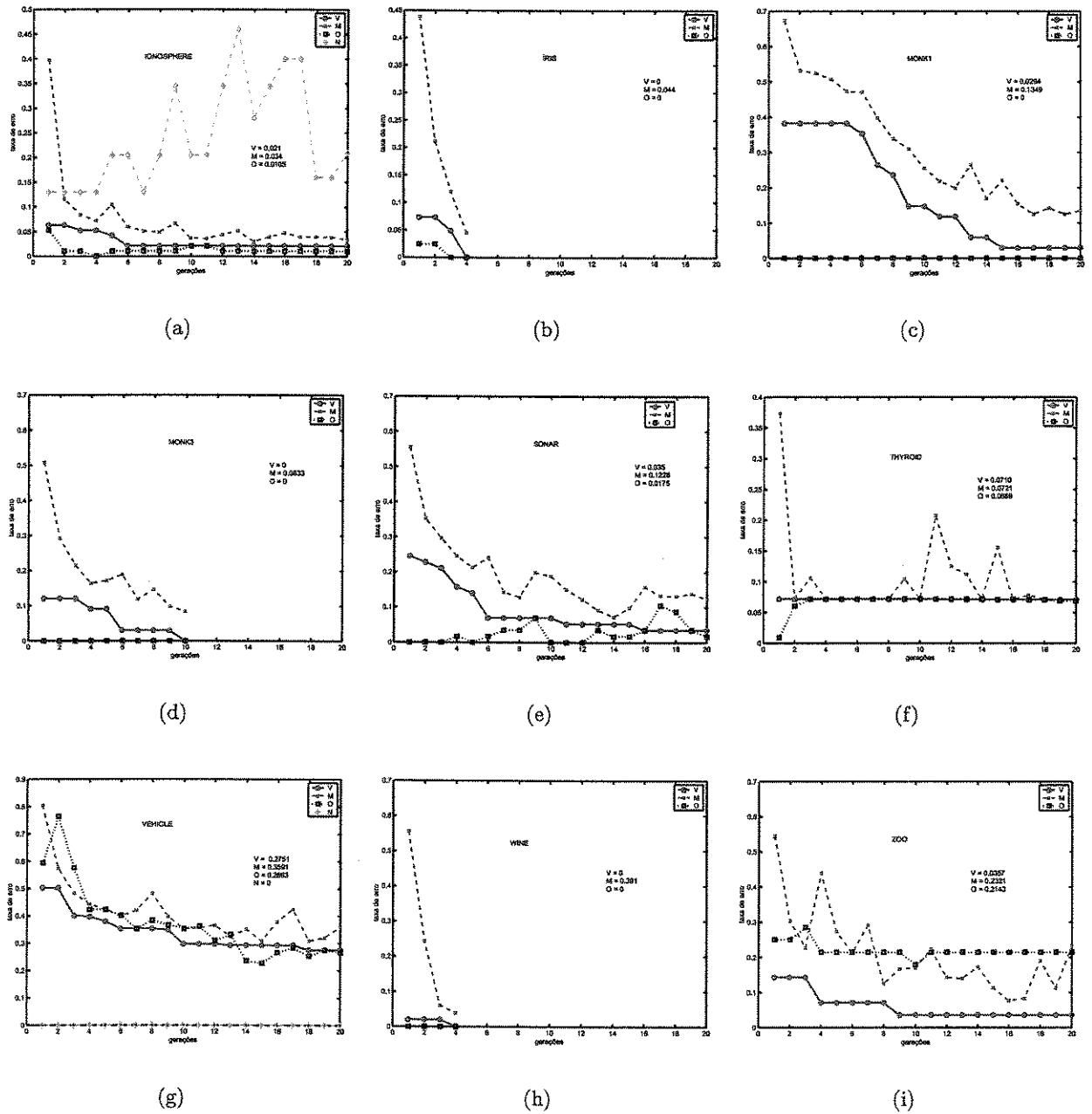


Fig. 7.7: Processo de convergência de COHENN para as nove últimas instâncias de problemas (desconsiderando a instância *MCR*), segundo a ordem estabelecida na Tab. C.2. “V” alude à taxa de erro da melhor descrição de *ensemble* da geração atual, “M” representa a taxa média de erro da população corrente de descrições de *ensembles*, “O” denota a taxa de erro do oráculo e “N” indica a taxa de componentes nulos na geração atual.

Tab. 7.7: Taxa média percentual de acerto, tanto sobre os dados de treinamento como os de teste, exibida pelo oráculo referente à população final de redes-componentes. A ordem das instâncias de problemas segue aquela da Tab. C.2. Legenda: “MIN”: mínimo valor; “MAX”: máximo valor; “DP”: desvio padrão.

Instância	Treinamento				Teste			
	Média	DP	MAX	MIN	Média	DP	MAX	MIN
1	99,72	0,88	100	97,22	100	0	100	100
2	99,78	0,38	100	98,94	98,55	1,18	100	97,1
3	93,75	1,61	97	91,65	95,17	1,16	96,51	93,02
4	95,19	1,65	97,86	92,51	95,36	1,5	97,1	92,75
5	50,91	6,64	60,61	36,36	66,67	3,7	72,22	58,33
6	92,36	2,08	95,67	89,42	89,74	1,94	93,42	86,84
7	37,56	19,56	66,67	8,62	47,38	11,58	61,9	28,57
8	98,54	1,26	100	96,34	98,67	1,72	100	96,67
9	92,38	2,19	95,24	88,1	91,99	5,25	100	86,67
10	99,16	0,97	100	96,84	99,14	1,38	100	97,14
11	100	0	100	100	100	0	100	100
12	99,41	1,24	100	97,06	99,17	2,63	100	91,67
13	100	0	100	100	100	0	100	100
14	96,67	2,54	100	91,23	99,5	1,58	100	95
15	93,18	0,09	96	92,8	93,12	0,1	96	92,78
16	63,32	14,6	80,79	40,61	58,45	11,31	73,81	39,29
17	100	0	100	100	100	0	100	100
18	66,43	11,69	78,57	46,43	79	11,97	90	60

tange à diversidade do material genotípico produzido ao longo das gerações do ciclo coevolutivo, pôde-se observar que o uso de taxas de mutação e recombinação adaptativas, aliado ao aspecto de segregação dos blocos construtivos em espécies distintas, lidou de maneira eficiente com o problema de convergência prematura, introduzindo, sempre que necessário, novo material genotípico às populações. Isso pode ser verificado visualizando a trajetória da taxa média de erro da população de descrições de *ensembles*. Por outro lado, no que se refere à diversidade dos padrões de generalização apresentados pelas redes-componentes geradas por COHENN, a atuação integrada dos *hotspots* (1), (2) e (3) indicados na Fig. 7.1 parece que surgiu efeito na maioria dos casos analisados. Para atestar isso, basta verificar que, para muitas das instâncias, a taxa de erro do oráculo no decorrer do processo evolutivo se manteve em patamares bem atraentes. Nesse contexto, a Tab. 7.7 traz a taxa média de acerto exibida pelo oráculo referente à última geração de indivíduos da população de redes-componentes, tanto sobre os dados de treinamento como sobre os de teste.

Muitas vezes, a curva relativa à taxa de erro do oráculo exhibe saltos bem significativos de uma geração para a outra, sinalizando ora a perda de diversidade das redes-componentes ora o aumento ou preservação dessa ambigüidade. As flutuações por si só não são problemáticas desde que essas taxas de erro sempre girem em torno de patamares razoáveis (como aconteceu com a simulação da Fig. 7.7(a)). Contudo, algumas situações “aberrantes” puderam ser detectadas nos experimentos conduzidos. Dois exemplos com condições semelhantes são aqueles ilustrados nas Figs. 7.6(e) e 7.6(g). A explicação para esses dois casos

é a mesma: o aumento, às vezes a ritmo crescente, do número de redes nulas compondo a população de redes-componentes. Por conta dos instrumentos adotados por COHENN de estímulo ao uso parcimonioso de recursos (em direção à redução da complexidade do modelo agregado final) e por conta do mecanismo de *credit assignment* (mediante as parcelas de *fitness* global e local), algumas vezes, tais componentes nulos podem vir a dominar sobremaneira a população de componentes. Entre dois componentes com a mesma acuidade local (p. ex., uma rede nula e uma não-nula que não acerte nenhum padrão de validação), aquele com menor complexidade tende a ser privilegiado, haja vista a existência da parcela de regularização de complexidade (*complex*) embutida na função de *fitness* das descrições de *ensembles*. Essa é uma deficiência da versão atual de COHENN, já que não se impõe limite algum sobre as taxas de crescimento do número de componentes nulos por geração.

Vale frisar também que, em alguns casos, a taxa de erro da melhor descrição de *ensemble* atual ficou (temporária ou perenemente) abaixo da taxa de erro exibida pelo oráculo. Duas razões existem para esse fato. A primeira é a de que, à medida que as gerações passam, as redes-componentes tendem a ficar cada vez mais sobreajustadas à sua partição de treinamento associada por conta de seu processo de aprendizado local. Como mencionado anteriormente, algumas vezes as redes-componentes integrando a melhor descrição de *ensemble* atual precisam ter seus pesos anteriores resgatados ao final da geração por se constatar que o novo ajuste dado a eles incorreu em perda de desempenho (acuidade) para a melhor *ensemble*. Como a medida da taxa de erro do oráculo não leva em conta essa possível recuperação dos pesos, pode ser que ela fique abaixo do valor da taxa de erro relativa à melhor *ensemble* encontrada até o momento. O segundo motivo parece corroborar a conjectura levantada por especialistas na área de AM de que a capacidade de generalização do modelo agregado pode sim ser bem superior àquela apresentada individualmente por seus componentes. Como já enaltecido à Seção 4.1, em muitos casos, a melhor hipótese (no caso, rede-componente) para o mapeamento desconhecido não está presente no espaço de hipóteses $\mathcal{H}$ atualmente sob consideração pelo indutor. Contudo, uma combinação apropriada das várias hipóteses extraídas de $\mathcal{H}$ (*ensemble*) pode vir a expandir esse espaço de funções representáveis, passando o espaço resultante a abarcar parte de, senão todo, o mapeamento desejado [123, 461].

A Tab. 7.6 também anuncia outras variáveis interessantes de serem perscrutadas para fins de análise de desempenho computacional da abordagem COHENN. Além do tempo médio de execução do ciclo coevolutivo (*TE*), que se trata de uma métrica temporal absoluta, é também apresentada uma métrica temporal relativa, chamada de *Índice Relativo de Tempo de Execução* (*IRTE*), cujo cálculo é dado como sendo:

$$IRTE = \frac{TE(\times 60)}{Ntr \times D} \left(\frac{s}{\text{Amostra} \times \text{Dimensão}} \right). \quad (7.3)$$

Ou seja, *IRTE* significa apenas uma normalização de *TE*, levando em conta parâmetros que podem caracterizar melhor as dificuldades (em termos de demanda por recursos computacionais) de cada instância de problema de classificação⁽¹⁵⁾. Pode-se constatar que para todos os casos, excetuando-se o da instância *Thyroid* (que apresenta o número mais elevado de amostras disponíveis), COHENN exibiu um padrão de desem-

⁽¹⁵⁾ Obviamente que tais parâmetros não devem ser interpretados como indicadores do grau de dificuldade intrínseco à tarefa de estimação associada a cada instância de problema.

Tab. 7.8: Número médio de pesos sinápticos ($\Upsilon(\Omega_{\text{COHENN1}})$) e número médio de neurônios escondidos ($\mathcal{U}(\Omega_{\text{COHENN1}})$) pertencentes aos arranjos COHENN1. *IRPS* significa índice relativo de pesos sinápticos (ver Eq. (7.4)). A ordem das instâncias de problemas segue aquela da Tab. C.2. Legenda: “MIN”: mínimo valor; “MAX”: máximo valor; “DP”: desvio padrão.

Instância	$\Upsilon(\Omega_{\text{COHENN1}})$				$\mathcal{U}(\Omega_{\text{COHENN1}})$				IRPS
	Média	DP	MAX	MIN	Média	DP	MAX	MIN	
1	73,4	25	111,5	38	8,59	3,26	14	4	4,89
2	206,63	51,82	302	145,33	8,63	2,09	12	6	7,65
3	556,5	79,3	663	403	37,2	4,39	43	30	18,55
4	258,5	64,93	343	174	13,9	3,21	18	10	6,15
5	225,4	47,78	299	170	5,1	0,99	7	4	0,94
6	232,8	56,61	279	124	24,8	5,03	29	15	9,7
7	101,35	44,45	197	50	6,45	2,83	13	3	1,8
8	122,18	52,85	253	70	7,75	2,94	13	4	3,13
9	72,73	32,86	129	42	3,43	1,75	6	2	1,28
10	191,05	67,88	330	73,5	8,05	2,48	13	3,5	1,87
11	74,05	20	102	48	8,1	2,38	12	6	4,63
12	98,25	23,49	133,5	65	8,5	2,37	12	5	5,46
13	83,1	25,2	124	38	7,1	2,28	10	3	4,62
14	129,95	30,94	170	70	3,45	0,83	5	2	0,72
15	1317,8	308,21	1788	805	91,3	14,64	101	51	15,69
16	272,9	82,26	382	176	17	6,24	28	10	3,03
17	144,5	28,18	204	108	10,4	1,78	14	8	2,78
18	120,5	32,01	164	63	5,4	1,43	7	3	0,89

penho computacional passível de ser considerado estável em termos de escalabilidade temporal. Ademais, como o peso λ_p relativo à parcela de parcimônia da função de *fitness* das descrições de *ensembles* foi arbitrado em apenas 10%, o número médio de componentes não-nulos das *ensembles* vencedoras finais (*NC*) oscilou na faixa entre dois e quatro componentes. É certo que com uma melhor calibração desse parâmetro de controle (ver sugestão de extensão à Seção 7.4) COHENN poderá se mostrar mais eficiente no que concerne à sua habilidade de realizar a poda evolucionária dos módulos neurais.

Ainda com relação à análise de complexidade dos modelos de combinação gerados por COHENN, a Tab. 7.8 traz medidas estatísticas referentes ao número médio de pesos sinápticos ($\Upsilon(\Omega_{\text{COHENN1}})$) e ao número médio de neurônios escondidos ($\mathcal{U}(\Omega_{\text{COHENN1}})$) presentes em todo o arranjo topológico da *ensemble* vencedora final. (São desconsiderados aqui os eventuais pesos sinápticos pertencentes às redes-combinadoras.) Nessa mesma tabela, um segundo índice, denominado *Índice Relativo de Pesos Sinápticos (IRPS)*, passa a ser reportado. Assim como *IRTE*, a métrica *IRPS* provê ao projetista um caminho mais direto e uniforme para se aferir o desempenho exibido por COHENN; no caso, quando do tratamento do critério de parcimônia. O cálculo de *IRPS* é dado como sendo:

$$IRPS = \frac{\Upsilon(\Omega_{\text{COHENN1}})}{(D+1) \times M} \left(\frac{\text{Peso}}{\text{Dimensão} \times \text{Classe}} \right). \quad (7.4)$$

Ou seja, tal critério dá uma indicação de quantas vezes maior que a rede *feedforward* mínima para o problema tende a ser o modelo de *ensemble* gerado por COHENN. A topologia de rede mínima seria aquela sem nenhuma camada escondida, apresentando somente neurônios de entrada e saída. Pode-se verificar que, excetuando-se *Thyroid* e *Car* (primeira e segunda colocadas em termos de número de amostras), para as demais instâncias de problemas, COHENN exhibe um comportamento de certo modo bem regulado no que tange ao uso dos recursos disponíveis para fins de mapeamento, gerando modelos compactos de *ensembles*.

Sob um outro ponto de vista, as Tabs. 7.9 a 7.11 trazem algumas medidas estatísticas interessantes relativas aos tipos de função de ativação e aos tipos de operadores de combinação fixos mais comumente adotados por COHENN quando da confecção adaptativa das *ensembles* vencedoras finais (COHENN1). Quando da contabilidade dos operadores de combinação, foram desconsiderados os (poucos) casos em que as *ensembles* vencedoras finais continham operadores ajustáveis em vez de fixos — esse número foi pequeno devido, em parte, ao pífio valor atribuído ao parâmetro $NIND\_COMB_a$. No caso de se ter mais de uma *ensemble* vencedora final, essa análise freqüencial foi feita tomando por base somente os arranjos vencedores dissimilares entre si. (O mesmo procedimento foi empregado para os cálculos de $\Upsilon(\Omega_{COHENN1})$ e $\mathcal{U}(\Omega_{COHENN1})$ discutidos acima.) Em existindo mais de uma cópia de uma mesma rede-componente, a freqüência dos tipos de neurônios leva em consideração todas as cópias. Os tipos de função de ativação de componentes nulos, por sua vez, não foram levados em conta para efeito de contabilidade.

No que diz respeito à freqüência relativa dos tipos de função de ativação, pode-se concluir que não houve uma classe específica de neurônio que se destacasse frente às demais em termos de menor ou maior “assiduidade”, analisando-se aqui sob uma perspectiva mais global — quer dizer, considerando todas as instâncias de problemas a um só tempo. Sob uma perspectiva mais local, ou seja, para cada instância em particular, houve casos em que certos tipos de neurônios apresentaram certa prevalência ante os demais. Contudo, antes que se possa tirar ilações mais confiáveis quanto à receita de uso ou não de certos tipos neurônios, novas execuções da arquitetura devem ser realizadas mais vezes (provavelmente, sob outras configurações para os seus parâmetros de controle) de modo a se obter dados freqüenciais mais apurados.

Quanto aos tipos de combinadores fixos, os dados freqüenciais sugerem uma prevalência, até certo ponto significativa, para os operadores de combinação projetados por Rogova [395] (de índices “36” a “41”) com base na teoria da evidência de Dempster & Shafer [9, 175, 499] (vide Seção B.1). Dentre as variações desse algoritmo de combinação que extrai informações probabilísticas (nível de resolução de confiança [497]) dos componentes, aquela de índice “39” foi a mais assídua nos arranjos de *ensembles* vencedoras criados por COHENN. Interessante frisar que Rogova [395] também aponta essa variação como uma das duas que obtiveram melhor desempenho em seus experimentos. Contudo, assim como para o caso anterior, novas rodadas de simulação devem ser conduzidas mais sistematicamente para que se possa tirar conclusões mais seguras acerca do real potencial de um operador de combinação em particular.

Tab. 7.9: Análise frequencial (%) das funções de ativação compondo os arranjos COHENN1. É exposta tanto a frequência específica de cada tipo de função para cada problema como a sua participação relativa no contexto mais geral, ou seja, considerando todos os problemas. A ordem das instâncias de problemas segue aquela da Tab. C.2 enquanto a ordem das funções de ativação é a mesma daquela estabelecida na Tab. 2.1.

Instância	Funções de ativação															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	6,06	18,18	3,03	0	6,06	6,06	0	9,09	15,15	3,03	9,09	9,09	3,03	6,06	3,03	3,03
2	12,12	6,06	9,09	3,03	15,15	12,12	3,03	0	6,06	0	9,09	3,03	9,09	9,09	3,03	0
3	5,88	5,88	2,94	0	26,47	2,94	26,47	0	2,94	17,65	0	0	0	0	0	8,82
4	0	2,56	2,56	5,12	12,82	17,94	12,82	7,69	5,12	0	10,25	2,56	10,25	0	5,12	5,12
5	12,9	9,68	16,13	3,23	9,68	0	0	3,23	6,45	9,68	16,13	0	3,23	9,68	0	0
6	7,14	14,29	7,14	2,38	7,14	4,76	2,38	2,38	11,9	11,9	9,52	2,38	0	2,38	4,76	9,52
7	3,57	14,29	7,14	0	7,14	7,14	7,14	0	10,71	3,57	10,71	14,29	3,57	3,57	7,14	0
8	0	0	31,58	10,53	2,63	2,63	0	0	7,89	5,26	23,68	7,89	0	0	5,26	2,63
9	34,62	3,85	11,54	3,85	7,69	11,54	0	7,69	0	3,85	0	7,69	7,69	0	0	0
10	5,71	8,57	2,86	0	20	2,86	0	0	11,43	11,43	5,71	2,86	5,71	11,43	8,57	2,86
11	4,88	19,51	0	7,32	0	4,88	0	2,44	9,76	17,07	4,88	4,88	0	7,32	14,63	2,44
12	5,41	8,11	2,7	2,7	5,41	16,22	8,11	0	2,7	0	2,7	0	2,7	5,41	16,22	21,62
13	12,12	12,12	3,03	6,06	6,06	9,09	0	6,06	6,06	9,09	12,12	12,12	0	0	3,03	3,03
14	7,14	10,71	3,57	3,57	14,29	3,57	7,14	3,57	14,29	3,57	14,29	3,57	3,57	0	0	7,14
15	2,27	2,27	11,36	2,27	9,09	2,27	11,36	9,09	2,27	4,56	25	0	0	6,82	4,55	6,82
16	28,12	6,25	0	0	12,5	21,87	12,5	6,25	0	3,12	3,12	3,12	0	0	0	3,12
17	5,13	10,26	10,26	5,13	5,13	7,69	0	0	12,82	10,26	10,26	2,56	5,13	0	10,26	5,13
18	6,06	0	15,15	12,12	3,03	9,09	3,03	3,03	12,12	9,09	6,06	9,09	0	0	9,09	3,03
Total	8,15	8,47	7,83	3,83	9,27	7,83	5,28	3,35	7,67	7,03	9,9	4,47	2,88	3,51	5,59	4,95

Tab. 7.10: Operador de combinação fixo mais freqüentemente adotado por COHENN para compor as *ensembles* vencedoras finais relativas a cada instância de problema. A ordem das instâncias de problemas segue aquela da Tab. C.2, ao passo que os índices dos operadores de combinação são dados de acordo com a lista apresentada no Apêndice B.

	Instância de problema								
	1	2	3	4	5	6	7	8	9
Operador	27,39	36	39	15	39	9	39	39	3
Frequência (%)	20	50	40	30	60	30,77	30	50	36,36
	10	11	12	13	14	15	16	17	18
Operador	15	15,38	18,19,39	8,16	19	10,4	39	37	37,38,39,40,41
Frequência (%)	27,27	15,8	16,67	20	46,15	19,35	30	18,75	20

Tab. 7.11: Operadores mais freqüentemente utilizados por COHENN para a geração de suas *ensembles* vencedoras finais. Os índices dos operadores de combinação são dados de acordo com a lista apresentada no Apêndice B.

	Operadores fixos				
	39	37	15	23	12
Frequência (%)	14,34	6,55	5,73	4,92	4,1

7.3.2 Análise de eficácia

Nesta parte, COHENN passa a ser alvo de apreciação sob os quesitos específicos de acuidade e generalização. A Tab. 7.12 fornece elementos essenciais para esse propósito, uma vez que indica as taxas médias percentuais de acerto de duas configurações de *ensembles* produzidas por COHENN. Enquanto COHENN1 alude ao arranjo de *ensemble* vencedor final de cada execução sobre uma dada partição gerada pelo método de validação cruzada, COHENN2 representa, dentre os protótipos de *ensembles* disponíveis à última geração da população de descrições de *ensembles*, aquele que apresentou maior êxito, quer dizer, melhor taxa de acerto, sobre o conjunto de teste⁽¹⁶⁾. Para essa contabilidade, quando o número de *ensembles* vencedoras finais foi maior que um, os resultados catalogados para efeito de análise de generalização se referem ao modelo mais bem sucedido (ou seja, com melhor taxa de acerto sobre o conjunto de teste).

Pode-se observar que, na maioria dos casos, os modelos COHENN1 apresentaram alta capacidade de generalização, mantendo níveis equivalentes da taxas de acerto para treinamento e teste. Interessante notar também que, para as instâncias *Iris* e *Wine*, ambas contendo poucas amostras disponíveis, o melhor modelo produzido pelo ciclo evolutivo para cada uma das 10 partições de dados alcançou 100% de índice de acerto sobre os dados de treinamento. Enquanto para a *Iris* pôde ser evidenciado um certo sobreajuste aos dados, o mesmo não ocorreu para a instância *Wine*. No caso de COHENN2, o que fica válido mencionar é que existem, muitas vezes, na população final outras configurações de *ensembles* disponíveis com capacidade de genera-

⁽¹⁶⁾ Ou seja, COHENN2 e COHENN1 não são necessariamente iguais.

lização distinta daquela apresentada pela *ensemble* vencedora final. Isto é, mesmo que essas configurações de *ensembles* disponíveis não tiveram desempenho equiparável ao do modelo COHENN1 quando da fase de avaliação de acuidade sobre os dados de treinamento, elas puderam exibir desempenho superior na fase de teste. Exemplos notórios são aqueles para as instâncias *Hepatitis* e *Iris*, em que, em média, o ganho de desempenho de COHENN2 sobre COHENN1 ficou em cerca de 8 a 10%.

As Tabs. 7.13 e 7.14 exibem as taxas percentuais de acerto de dois outros modelos neurais (de granulosidade distintas) passíveis de serem extraídos após uma certa execução de COHENN. Enquanto COHENN3 alude à arquitetura de *ensembles* de *ensembles* evolucionárias do tipo (A)⁽¹⁷⁾, com o operador de combinação do segundo nível sendo o voto majoritário simples, COHENN4 diz respeito ao melhor modelo de rede-componente produzido em todas as gerações da espécie de redes-componentes (ou seja, aquele arranjo de rede-componente com melhor *fitness* local). Deve ser salientado que COHENN4 não necessariamente é módulo integrante de COHENN1, já que o seu elevado *fitness* local (acuidade) não implica que ele apresente um elevado *fitness* global (cooperação em grupo).

Assim como ocorreu com COHENN1, na maior parte dos casos, pode-se dizer que tanto COHENN3 como COHENN4 apresentaram uma boa capacidade de generalização, com manutenção de taxas de acerto na fase de teste equivalentes àquelas obtidas no treinamento. Os níveis de acuidade, porém, são geralmente inferiores, em média, àquele exibido por COHENN1. No que tange à configuração COHENN3, a explicação para esse fato decorre de dois aspectos distintos. O primeiro deles diz respeito ao tipo de operador de combinação usado. Embora voto majoritário simples seja um dos operadores mais freqüentemente adotados por projetos relatados na literatura [32, 273] (e até apresentou maior freqüência nos arranjos COHENN1 para a instância de problema *Hepatitis*), ele é um dos menos rebuscados presentes no repertório de operadores utilizado aqui⁽¹⁸⁾. Em segundo lugar, assim como o que ocorreu com os experimentos conduzidos por Yao & Liu [504], certas vezes a última população de descrições de *ensembles* apresentou várias cópias das melhores *ensembles* obtidas, incorrendo em perda de diversidade. No que concerne à configuração COHENN4, pode-se notar que, para algumas instâncias de problemas, o grau de acuidade apresentado pelas melhores redes-componentes, tanto para o treinamento como para o teste, mostrou-se equiparado àquele exibido por COHENN1. Aliás, houve casos em que o nível de precisão exibido no treinamento foi superior. Isso vem a demonstrar a habilidade exibida pelo modelo coevolutivo de se gerar blocos construtivos (i. e., módulos-componentes para o nível superior) de elevada qualidade. Destarte, muitas vezes, pode o projetista decidir recorrer aos modelos COHENN4 no lugar de COHENN1, caso seja de seu interesse o uso de arquiteturas menos complexas.

⁽¹⁷⁾Optou-se por não expor os resultados relativos ao arranjo (B) de *ensembles* de *ensembles* evolutivas pois eles foram muito similares aos de COHENN1. Isso se deveu ao fato de, para muitas das instâncias de problemas, a descrição de *ensemble* vencedora em potencial ter sido localizada já nas primeiras gerações do ciclo coevolutivo; ou seja, grande parte das *ensembles*-componentes (cada uma representando o melhor arranjo de cada geração) passaram a ser cópias de uma mesma descrição.

⁽¹⁸⁾Fica subentendido, portanto, que o desempenho de COHENN3 poderia ter sido incrementado caso outros tipos de operadores da lista tivessem sido usados para assumir o papel do combinador de segundo nível.

Tab. 7.12: Taxas médias percentuais de acerto dos modelos gerados por COHENN sobre os conjuntos de treinamento e teste. COHENN1 alude ao arranjo de *ensemble* vencedor final de cada execução, ao passo que COHENN2 representa, dentre os protótipos de *ensembles* da última geração, aquele que apresentou melhor taxa de acerto sobre o conjunto de teste. A ordem das instâncias de problemas segue aquela da Tab. C.2. Legenda: “MIN”: mínimo valor; “MAX”: máximo valor; “DP”: desvio padrão.

Instância	COHENN1						COHENN2					
	Treinamento			Teste			Treinamento			Teste		
	Média	DP	MIN	MAX	Média	DP	MÍN	MAX	Média	DP	MÍN	MAX
1	91,67	2,62	86,11	94,44	77,78	6,81	75	91,67	81,95	8,19	75	91,67
2	97,35	0,49	96,83	98,41	94,34	1,27	91,3	95,65	94,78	1,01	92,75	95,65
3	88,95	1,72	86,72	91,43	89,77	1,94	87,21	94,19	89,94	1,98	87,21	94,19
4	89,84	0,25	89,3	90,37	87,1	1,59	85,51	91,3	88,12	1,5	86,96	91,3
5	83,64	2,96	79,8	86,87	89,44	6,25	75	97,22	90,83	4,55	83,33	97,22
6	81,35	0,95	79,81	82,69	78,81	2,44	75	82,89	80,66	1,97	77,63	84,21
7	64,49	2,02	62,07	67,24	53,81	7,46	38,1	61,9	56,19	5,4	47,62	61,9
8	94,02	1,07	91,46	95,12	90,34	3,3	86,7	93,33	92,67	2,1	86,7	93,33
9	91,19	1,96	88,1	95,24	74	6,63	60	80	81,33	4,22	73,33	86,67
10	98,42	0,56	97,89	98,95	89,72	4,3	82,86	94,29	91,72	2,84	85,71	94,29
11	100	0	100	100	86,67	0	86,67	86,67	93,33	0	93,33	93,33
12	96,47	2,71	91,18	100	91,67	8,78	75	100	96,67	4,3	91,67	100
13	100	0	100	100	95	5,83	83,33	100	99,17	2,63	91,67	100
14	91,22	3,1	87,72	96,5	80	6,24	70	90	83,5	7,09	75	95
15	92,83	0,05	92,8	92,9	92,78	0	92,78	92,78	92,78	0	92,78	92,78
16	71,97	1,21	69,87	73,36	59,16	2,51	54,76	63,1	61,67	2,91	58,33	66,67
17	100	0	100	100	97,65	4,11	88,24	100	99,41	1,86	94,12	100
18	94,29	1,84	92,86	96,43	96	6,99	80	100	97	4,83	90	100

Tab. 7.13: Taxas médias percentuais de acerto dos modelos de *ensembles* de *ensembles* evolucionárias com combinação por voto majoritário (COHENN3) gerados por COHENN sobre os conjuntos de treinamento e teste. A ordem das instâncias de problemas segue aquela da Tab. C.2. Legenda: “MIN”: mínimo valor; “MAX”: máximo valor; “DP”: desvio padrão.

Instância	Treinamento				Generalização			
	Média	DP	MIN	MAX	Média	DP	MIN	MAX
1	88,89	3,21	83,33	94,44	75	5,27	66,67	83,33
2	96,93	0,6	95,77	97,88	93,62	1,01	91,3	94,2
3	88,39	2,22	83,94	91,43	89,48	1,97	86,63	93,6
4	89,04	0,63	87,7	89,84	86,96	0	86,96	86,96
5	81,52	4,71	73,74	87,88	89,44	4,86	83,33	97,22
6	79,52	1,4	75,96	80,77	79,21	1,73	76,32	81,58
7	59,65	8,18	44,83	67,24	53,87	6,36	42,86	61,9
8	92,8	1,57	90,24	95,12	90,68	3,05	86,7	93,33
9	89,29	2,02	85,71	92,86	76,67	7,19	66,7	86,67
10	97,89	1,22	95,79	98,95	89,43	3,58	82,86	94,29
11	98,78	1,29	97,56	100	87,5	5,6	80	93,33
12	91,18	5,55	82,35	97,06	94,17	5,62	83,33	100
13	94,55	2,78	87,88	96,97	97,5	5,63	83,33	100
14	87,89	4,41	78,95	92,98	80	8,16	65	95
15	92,83	0,05	92,8	92,9	92,78	0	92,78	92,78
16	70,74	1,62	68,12	72,93	59,64	2,78	55,95	64,29
17	98,37	3,57	89,8	100	95,88	6,23	82,35	100
18	93,22	2,03	89,29	96,43	95	7,07	80	100

Tab. 7.14: Taxas médias percentuais de acerto dos melhores modelos de redes-componentes (COHENN4) gerados por COHENN sobre os conjuntos de treinamento e teste. A ordem das instâncias de problemas segue aquela da Tab. C.2. Legenda: “MIN”: mínimo valor; “MAX”: máximo valor; “DP”: desvio padrão.

Instância	Treinamento				Teste			
	Média	DP	MIN	MAX	Média	DP	MIN	MAX
1	91,39	0,88	88,89	91,67	77,5	8,83	58,33	83,33
2	97,04	0,51	96,3	97,88	93,62	2,07	91,3	97,1
3	79,57	5,07	72,59	86,51	81,22	6,89	72,09	89,53
4	89,52	0,28	89,3	89,84	87,25	0,61	86,96	88,41
5	52,07	7,94	45,45	72,22	63,28	7,96	44,44	75
6	80,77	0,99	78,85	81,73	78,03	2,33	73,68	81,58
7	36,9	8,95	22,41	56,9	35,71	6,83	28,57	52,38
8	94,27	1,02	92,68	96,4	86,02	4,38	76,7	90
9	90,96	1,88	88,1	95,24	71,34	8,91	53,33	86,67
10	97,58	1	95,79	98,95	86,57	4,68	80	91,43
11	98,54	1,71	95,12	100	82,67	4,66	73,33	86,67
12	91,47	5,09	82,35	97,06	91,67	10,39	75	100
13	99,09	1,46	96,97	100	96,67	4,3	91,67	100
14	88,95	2,35	85,96	92,98	74	9,94	50	85
15	88,29	2,39	86,68	92,8	87,13	3,07	85	92,78
16	42,71	6,16	34,93	54,15	38,69	5,59	28,57	46,43
17	99,18	1,43	95,92	100	94,71	4,34	88,24	100
18	72,86	3,45	67,86	78,57	76	9,66	50	80

Tab. 7.15: Comparação dos modelos produzidos por COHENN com aqueles oriundos das técnicas de *Boosting*, *Bagging* e *Stacking*, em termos da taxa média percentual de acerto sobre os dados de teste. A ordem das instâncias de problemas segue aquela da Tab. C.2. Legenda: “DP”: desvio padrão.

Instância	COHENN1		COHENN2		COHENN3		COHENN4		Boosting		Bagging		Stacking	
	Média	DP	Média	DP	Média	DP	Média	DP	Média	DP	Média	DP	Média	DP
1	77,78	6,81	81,95	8,19	75	5,27	77,5	8,83	90,71	6,05	90,77	7,94	87,69	13,17
2	94,34	1,27	94,78	1,01	93,62	1,01	93,62	2,07	95,57	3,53	96,28	2,26	96,42	1,94
3	89,77	1,94	89,94	1,98	89,48	1,97	81,22	6,89	99,07	0,68	99,19	0,62	98,9	1,23
4	87,1	1,59	88,12	1,5	86,96	0	87,25	0,61	85,92	6,56	87,35	5,84	86,53	6,18
5	89,44	6,25	90,83	4,55	89,44	4,86	63,28	7,96	96,73	2,8	96,46	2,57	96,19	2,91
6	78,81	2,44	80,66	1,97	79,21	1,73	78,03	2,33	76,44	4,73	76,3	4,88	77,08	2,97
7	53,81	7,46	56,19	5,4	53,87	6,36	35,71	6,83	62,06	6,22	61,6	6,61	57,84	10,72
8	90,34	3,3	92,67	2,1	90,68	3,05	86,02	4,38	81,51	6,89	82,85	6,19	84,15	6,43
9	74	6,63	81,33	4,22	76,67	7,19	71,34	8,91	81,42	14,82	85,21	10,21	80,75	9,85
10	89,72	4,3	91,72	2,84	89,43	3,58	86,57	4,68	92,6	2,71	88,89	3,67	91,74	3,67
11	85,34	2,81	92	4,22	87,5	5,6	82,67	4,66	96	5,62	96	4,66	95,33	4,5
12	91,67	8,78	96,67	4,3	94,17	5,62	91,67	10,39	92,5	9,98	91,86	6,81	95	8,96
13	95	5,83	99,17	2,63	97,5	5,63	96,67	4,3	89,42	10,9	91,03	9,07	88,46	10,59
14	80	6,24	83,5	7,09	80	8,16	74	9,94	84,17	9,22	83,67	4,54	80,31	10,6
15	92,78	0	92,78	0	92,78	0	87,13	3,07	94,38	0,43	94,58	0,29	94,13	0,49
16	59,16	2,51	61,67	2,91	59,64	2,78	38,69	5,59	79,92	4,51	79,56	4,43	80,14	3,59
17	97,65	4,11	99,41	1,86	95,88	6,23	94,71	4,34	97,75	2,91	97,19	2,96	97,75	2,91
18	96	6,99	97	4,83	95	7,07	76	9,66	96,09	5,05	96,09	5,05	91,18	8,52

Por sua vez, a Tab. 7.15 contrasta as taxas médias percentuais de acerto sobre os dados de teste exibidas pelos modelos produzidos por COHENN com aquelas produzidas por modelos oriundos da aplicação das técnicas de *Boosting*, *Bagging* e *Stacking*, considerando os mesmos *folds* gerados pelo método de validação cruzada estratificada e assumindo RNAs *feedforward* de topologias prefixadas⁽¹⁹⁾ como estimadores fracos (instáveis). No caso de *Boosting*, adotou-se a versão M1 (com reponderação das amostras) do algoritmo AdaBoost [160] (ver Fig. 4.4 e Alg. 2). Já para *Stacking*, o treinamento do combinador é feito via validação cruzada, sendo esse combinador uma outra RNA *feedforward*, cujos parâmetros topológicos e de aprendizado foram configurados como sendo iguais aos das redes-componentes.

Os resultados sugerem que, na maioria dos casos, o desempenho exibido por COHENN1 e COHENN2 se equipara àquele obtido com as técnicas mencionadas. Enquanto para as instâncias de problemas de índices “2”, “4”, “6”, “9”, “10”, “12”, “13”, “14”, “17” e “18” a aplicação par-a-par do teste-T emparelhado (ver Seção A.3) veio a confirmar a hipótese nula de equivalência de desempenho (pois o grau de significância foi maior que o *threshold* de confiança de 95%), para as instâncias de índices “1”, “3”, “5”, “11” e “15”, a mesma constatação não pôde ser comprovada, já que os modelos COHENN1 e COHENN2 apresentaram desempenho inferior. Para a instância de índice “7”, COHENN1 e COHENN2 foram equivalentes a *Stacking*, mas não a *Bagging* e *Boosting*. Por outro lado, para a instância 8, o teste-T emparelhado indicou desempenho superior para COHENN1 e COHENN2 frente aos demais. É válido lembrar que novas calibrações para os parâmetros de controle de COHENN (p. ex., via aumento do número de gerações e/ou do tamanho das populações) podem vir a implicar ganhos significativos de desempenho, colocando-a mais próxima das outras técnicas para as instâncias de problemas mais complicadas, como as de índices “7” (*Glass*) e “16” (*Vehicle*) — para essas instâncias, os níveis de acuidade apresentados por todos os modelos (i. e., produzidos ou não por COHENN) ficam bem aquém daqueles exibidos para as demais instâncias, o que atesta o elevado grau de dificuldade associado à tarefa de estimação de tais problemas.

Finalmente, é pertinente que se tente posicionar o nível de acuidade demonstrado por COHENN num contexto mais abrangente, promovendo comparações com outras abordagens proeminentes de aprendizado de máquina. As Tabs. 7.16 a 7.18 trazem informações a esse respeito, comparando as taxas médias de erro sobre dados de teste apresentadas por COHENN1 com aquelas exibidas por outras técnicas para as instâncias de problemas *Credit*, *Diabetes* e *Heart*⁽²⁰⁾. Duas classes de algoritmos podem ser distinguidas de antemão: uma de cunho evolucionário (englobando COHENN, COVNET [168], ELMI [299], EENCL [298] e SpecEns [281]) e outra de cunho não-evolucionário [1, 321] (envolvendo métodos puramente estatísticos, métodos baseados em árvores de decisão, métodos baseados em regras e métodos conexionistas). O que os resultados sugerem é que as abordagens evolutivas tendem a se mostrar mais eficazes que as demais, com destaque ainda maior

⁽¹⁹⁾Para *Boosting* e *Bagging*, o número de RNAs *feedforward* usadas como componentes corresponde ao número máximo de iterações ou reamostragens conduzidas pela técnica. Para os experimentos, esse valor foi configurado como sendo igual ao de *NCOMP* (idem para *Stacking*). Já o número de neurônios escondidos, a taxa de aprendizado e o número de épocas de treinamento dessas redes foram arbitrados como sendo *NMAX_HID_COMP*, 0,3 e 50, respectivamente, para todas as simulações.

⁽²⁰⁾É pertinente mencionar que nem todas as taxas de erro apresentadas nessas tabelas foram obtidas via método de validação cruzada.

Tab. 7.16: Comparação das taxas médias de erro exibidas por COHENN1 com aquelas apresentadas por outras abordagens de AM para a instância de problema *Credit*. (Fontes: Michie *et al.* [321], Liu *et al.* [298, 299], Lee *et al.* [281] e García-Pedrajas *et al.* [168].)

Modelo	Tx. de erro	Modelo	Tx. de erro	Modelo	Tx. de erro
COHENN1	0,129	COVNET	0,116	EPNET	0,115
ENNCL	0,135	AC ²	0,181	EPNET-ens	0,093
ELMI	0,13	BayTree	0,171	acasper	0,137
Discrim	0,141	NaiveBay	0,151	acascor	0,136
Quadisc	0,207	CN2	0,204	DIPOL92	0,141
Logdisc	0,141	C4.5	0,155	BP	0,154
SMART	0,158	ITrule	0,137	RBF	0,145
ALLOC80	0,201	Cal5	0,131	CART	0,145
k-NN	0,181	IndCART	0,152	LVQ	0,197
CASTLE	0,148	NewID	0,181	SpecEvol	0,089

Tab. 7.17: Comparação das taxas médias de erro exibidas por COHENN1 com aquelas apresentadas por outras abordagens de AM para a instância de problema *Diabetes*. (Fontes: Michie *et al.* [321], Liu *et al.* [298, 299], Lee *et al.* [281] e García-Pedrajas *et al.* [168].)

Modelo	Tx. de erro	Modelo	Tx. de erro	Modelo	Tx. de erro
COHENN1	0,212	COVNET	0,199	EPNET	0,224
ENNCL	0,221	AC ²	0,276	EPNET-ens	0,222
ELMI	0,223	BayTree	0,271	acasper	0,231
Discrim	0,225	NaiveBay	0,262	acascor	0,245
Quadisc	0,262	CN2	0,289	MPyramid	0,232
Logdisc	0,223	C4.5	0,27	MTilling	0,229
SMART	0,301	ITrule	0,245	Kohonen	0,273
ALLOC80	0,301	Cal5	0,25	DIPOL92	0,224
k-NN	0,324	IndCART	0,271	BP	0,248
CASTLE	0,258	NewID	0,289	RBF	0,243
CART	0,255	LVQ	0,272	SpecEvol	0,202

Tab. 7.18: Comparação das taxas médias de erro exibidas por COHENN1 com aquelas apresentadas por outras abordagens de AM para a instância de problema *Heart*. (Fontes: Michie *et al.* [1, 321] e García-Pedrajas *et al.* [168].)

Modelo	Tx. de erro	Modelo	Tx. de erro	Modelo	Tx. de erro
COHENN1	0,097	COVNET	0,143	EPNET	0,168
acasper	0,192	acascor	0,199	EPNET-ens	0,151
MSM1	0,165	RBF	0,182	IncNet	0,1
Discrim	0,158	NaiveBay	0,166	LDA	0,155
QDA	0,246	BP	0,187	SNB	0,169
SVM	0,185	C4.5	0,462	CART	0,192
k-NN	0,158	LVQ	0,171	FSM	0,16

para as abordagens coevolutivas COVNET e COHENN. Tratando-se particularmente da instância de problema *Heart*, COHENN parece ter atingido o nível de acuidade mais elevado até o presente, tomando por base a vasta lista comparativa de indutores veiculada na página virtual do projeto Statlog [1] — a abordagem IncNet figura como a que obteve o maior êxito para essa instância de problema.

7.4 Recapitulação e outras perspectivas

O objetivo deste capítulo foi o de expor detalhadamente um novo modelo arquitetural de gênero coevolutivo cooperativo, de alcunha COHENN, destinado à configuração automática de *ensembles* heterogêneas de RNAs *feedforward*. Como encaminhado na Seção 7.1, a principal motivação por trás da concepção dessa nova abordagem girou em torno da perspectiva de se propiciar uma ferramenta alternativa dirigida ao tratamento uniforme e integrado das etapas comumente envolvidas no projeto de *ensembles* efetivas. Mediante o ajuste simultâneo e coadaptativo dos vários *hotspots* apresentados à Fig. 7.1, COHENN passa a lidar, de maneira transparente, com as possíveis influências mútuas exercidas por esses pontos. Trata-se de uma propriedade de grande apelo, uma vez que a identificação clara, *a priori*, dessas interdependências é algo difícil de se realizar na prática.

Ademais, dada a natureza hierárquica e simbiótica inerente a COHENN, torna-se possível o desenho não-manual e concorrente de sistemas modulares connexionistas exibindo diferentes níveis de granulosidade, encerrando desde arquiteturas de redes neurais simples (i. e., redes-componentes) como arranjos topológicos mais elaborados (i. e., *ensembles* e *ensembles* de *ensembles*). Enquanto a Subseção 7.2.1 discorreu especificamente sobre as funcionalidades promovidas por COHENN, a Subseção 7.2.2 municiou o leitor com uma visão clara e objetiva acerca da dinâmica e dos elementos estruturais que compõem tal modelo arquitetural. Os resultados experimentais relativos às várias simulações conduzidas pelo autor são relatados e discutidos de maneira detalhada à Seção 7.3, em que ficam evidenciadas as potencialidades exibidas por COHENN, tanto em termos de eficiência como de eficácia. Com base em tal experimentação sistemática, pôde-se confirmar a hipótese levantada anteriormente de que as propriedades intrínsecas ao maquinário (co)evolutivo são, de fato, bem apropriadas para arcar com a tarefa não-trivial de projeto automático de comitês de máquinas.

Algumas extensões à versão atual do modelo arquitetural podem ser vislumbradas com o intuito de se sanar algumas de suas deficiências e/ou de incrementar ainda mais as suas potencialidades. Uma primeira extensão seria a de possibilitar o treinamento local das redes-combinadoras, empregando algum esquema de validação cruzada, assim como estabelecido para *Stacking*. Isso ainda não foi feito pelo seguinte motivo: se uma dada rede-combinadora participar de mais de uma descrição de *ensemble* simultaneamente, os ajustes feitos em seus pesos sinápticos, tomando por base as saídas das redes-componentes de uma das *ensembles*, poderão entrar em conflito com aqueles ajustes a serem feitos num segundo momento, tomando por base as saídas dos componentes de uma outra *ensemble*. Uma alternativa seria a de se treinar somente uma rede-combinadora caso ela pertencesse à melhor descrição de *ensemble* atual. Outro meio de se contornar tal problema seria a de possibilitar a criação de diferentes clones da mesma rede-combinadora, substituindo

as outras redes-combinadoras da população com menor *fitness* local.

Uma segunda extensão interessante seria a de se proporcionar um esquema de escalonamento temporal para lidar, em diferentes épocas, com os objetivos conflitantes impostos à otimização das descrições de *ensembles*. Assim, num primeiro momento, o sistema estaria focando em localizar, por exemplo, arranjos de combinação com maior nível de acuidade, deixando de lado os critérios de parcimônia e complexidade. Ao longo do ciclo coevolutivo, os pesos associados aos critérios de parcimônia e diversidade (λ_p e λ_d) passariam a ser incrementados paulatinamente, incorrendo na eliminação de redundâncias e aumento de diversidade nos arquétipos com maiores níveis de precisão já encontrados. Uma outra maneira possível de lidar com essa questão seria a de se codificar nos próprios cromossomos das descrições de *ensembles* os valores de λ_p e λ_d , deixando o próprio engenho evolutivo revelar os melhores valores para esses parâmetros de controle. Uma terceira alternativa para essa questão seria a de se servir dos insumos providos pelos AEs estendidos para lidar com problemas de otimização multiobjetivo [95, 116], assim como realizado pela abordagem MOBNET [167].

Um terceiro ponto de investigação diz respeito ao problema de atribuição de crédito às redes-componentes nulas, como forma de se evitar que, às vezes, tais módulos dominem a população, como notificado anteriormente. Uma possível saída para isso seria a de se controlar a taxa de crescimento desses módulos por geração; ao se passar de um certo limiar preestabelecido, alguns dos componentes nulos passariam a ter seus *bits* reguladores alterados aleatoriamente. Além disso, a exemplo do que foi feito para as redes-componentes, os cromossomos relativos às descrições de *ensembles* poderiam também passar a apresentar campos seletores indicando quais dos *N_COMP* módulos possíveis irão ser, de fato, evocados para fins de combinação. Esse recurso foi adotado, de um modo similar, pela abordagem HGT-GAME, como meio eficaz de se definir automaticamente o número de especialistas em arranjos de ME. Com isso, poder-se-ia eliminar os problemas relativos às redes-componentes nulas, que passariam a não mais existir. Uma terceira alternativa seria a de se atribuir à parcela de *fitness* local dessas redes a média dos valores de *fitness* local das redes-componentes não-nulas, em vez de ter que se arbitrar um valor (no caso, “1,0”) qualquer.

Outros pontos que já estão sob investigação compreendem (i) a criação e avaliação de desempenho de outros tipos de arranjos de *ensembles* de *ensembles* evolutivas (com outros tipos de operadores de combinação); (ii) o emprego de outras métricas para medição do grau de diversidade intrínseco a cada protótipo de *ensemble* (ver Tab. 4.1); e (iii) a avaliação de desempenho do aprendizado híbrido baldwiniano, no lugar de lamareckiano, nos módulos componentes produzidos por COHENN.

Capítulo 8

Outros projetos em engenharia de sistemas inteligentes modulares

“Estamos todos num mesmo barco, em mar tempestuoso, e devemos uns aos outros uma terrível lealdade.”

G. K. Chesheston

Este capítulo é reservado à apresentação das demais contribuições geradas pelo autor na área de sistemas inteligentes modulares, dando destaque àquelas empregando técnicas evolutivas, conforme as metas estabelecidas no Cap. 1. Optou-se pela reprodução, na íntegra⁽¹⁾, de artigos relativos aos projetos desenvolvidos como meio de se manter coerência na exposição das idéias. Comentários subseqüentes, almejando explanar alguns pontos complementares sobre os trabalhos, são também fornecidos ao longo da exposição. Ao final, é feita uma revisão abreviada de outros empreendimentos conduzidos pelo autor que são correlatos ao tema.

8.1 Redes neurais coevolutivas

O primeiro artigo, intitulado “*Evolving heterogeneous neural networks for classification problems*” e publicado como texto completo nos anais da terceira edição da *Genetic and Evolutionary Computation Conference* [94], se destina, num primeiro momento, a caracterizar as propriedades de redes neurais heterogêneas (i. e., RNAs *feedforward* com diferentes tipos de neurônios). Em seguida, descreve-se, mais detalhadamente, uma nova abordagem coevolucionária hierárquica, chamada de HCGA, que foi concebida tendo em vista o desenho automático de instâncias de redes neurais heterogêneas para problemas de classificação.

⁽¹⁾Algumas alterações foram feitas nos textos como forma de corrigir pequenos erros ortográficos ou prover informações adicionais aos resultados dos experimentos. De outro modo, para evitar conflitos com a seqüência numérica (arábica) das seções da tese, decidiu-se que as seções dos artigos fossem ordenadas por numerais romanos.

I. Introduction

Multi-layer Perceptron (MLP) and Radial Basis Function (RBF) networks are the most common feedforward artificial neural networks (ANNs) used on classification, recognition and prediction problems. Typically, ANNs are composed of a predefined number of similar neurons in their hidden layer. Despite the existence of some rules of thumb to indicate this parameter in accordance with the available training data, such empirical considerations are failure-prone, preventing their applicability in all cases.

In this context, designing ANNs through simulated evolution has been showing its effectiveness as an automatic alternative to manual configuration. The designer is no more required to provide expertise knowledge about the problem to be handled and its representation into the neural architecture [480]. Moreover, such technique is very adaptable to dynamic changes in the environment and its employment has brought improvements on a range of aspects, such as better network performance, increased robustness, faster training, and better utilization of the computational resources (parsimony). As a multitude of different net architectural solutions exists for each problem domain, finding the best configuration (or just a high-quality one) can be regarded as well as a searching problem, where the aforementioned aspects could be considered as guide parameters for assessing and leveraging an evolving population of neural prototypes.

By other means, although the node activation function has been shown to be an important parameter for the ANN configuration—it is typically the source of non-linearities responsible for the approximation capabilities—, fewer works have explored such fact while devising new neural net topologies. Currently, there are no extensive results reported in the literature contemplating the actual effectiveness of employing Gaussian or logistic transfer functions in all neurons at the hidden layer. This tendency is followed, generally, for simplification purposes.

However, if the mapping function of each neuron could be automatically defined, better results regarding the net performance and the training convergence rate could be achieved. In this work, we attempt to investigate such conjecture by using GAs for the design of *heterogeneous neural networks*. The idea is to view each hidden unity as a peer with distinguished capabilities (different transfer node) to be explored while integrating with others in a same aggregate (neural net). Such abstraction aims at the *cooperative* combination of evolving populations of experts, mixing their qualifications towards the problem solution [375, 512].

For such endeavor, we have made use of a hierarchical approach, known as HCGA, composed of two evolutionary levels [336]. In an upper level, a GA is used to build the net topology, that is, to choose neurons (possibly with distinct features) to take part into the net hidden layer. In a bottom level, a coevolutionary approach was selected in order to train the networks by adjusting both the activation function parameters of the hidden neurons and the incoming input weights.

In the remainder of the paper, we introduce remarks concerning the employment of evolutionary techniques for neural nets configuration, present our framework towards the automatic design of heteroge-

neous networks, show results from our experiments, identify future plans of work, and address some final considerations.

II. Genetic Algorithms and Coevolution

In the conventional genetic algorithm model, a population of strings (chromosomes) codifying the possible solutions for the problem at hand passes through a cyclic process in which new candidates are constantly created and evaluated according to some adequacy measure known as *fitness*. Ancestors are charged by computational operators, very much resembling natural evolutive phenomena (reproduction, selection and mutation), being progressively replaced by more adapted newcomers. The population fitness tends to converge on the course of the process and (sub-) optimal solutions are obtained at final stages.

Some problems with this model have been reported. First, it is very prone to the “local minima/maxima problem”, as it depends on the configuration (search space distribution) of the initial population. Likewise, some fast convergence problems may occur if the population size is not properly set. Finally, it is not very suitable for the representation/generation of complex structures such as those composed by many entities (e.g. neural net architectures).

There is now such a trend to apply coevolution as a more effective technique towards complexity overcoming [377]. In this realm, two or more populations of the same or different species are optimized together, one influencing the other by some means. The coevolutionary process has a range of features [375]: (i) Each species represents a subcomponent of a potential solution; (ii) Complete solutions are obtained through the composition of members representing the different types of species (A given species could have more than one representative in the group.); (iii) The credit assignment for each species is a direct function of the values of fitness associated with complete solutions in which their members take part; (iv) There is flexibility on defining the number of available species; and (v) A separated evolutionary algorithm (GA, ES), that may have a totally different configuration from the others, does the evolution of each species.

III. Evolutionary Algorithms Applied to Neural Networks

GAs have been used in conjunction with neural networks in three major undertakings [480]: data pre-processing, weight optimization algorithm, and a procedure to search for a suitable neural net topology. The second synergy has been hampered mainly because of the *Competing Conventions Problem*. As the weight adjustment with GAs relies heavily on recombination, there may be many equivalent symmetric solutions to the same optimization problem, which may delay the convergence process. This can be alleviated through more appropriate crossover operators, which would try to avoid individuals showing the same cyclic genetic order of their chromosomes.

Another strategy centers upon the integration between evolutionary programming (EP) and ANNs. Liu and Yao [295] have presented an EP-based algorithm for the tuning of ANNs with different activation function nodes. The weights are adjusted through a combination of the Backpropagation (BP) algorithm with a random search algorithm. For simplicity, the authors chose to use only the logistic and the Gaussian neurons, as they represent two broad classes of activation functions with complementary features. The resulting *generalized neural net* (GNN), as they call, resembles very much what we name here as heterogeneous neural networks, albeit our approach gears towards the employment of a hierarchical coevolutionary-genetic algorithm (HCGA) and is open to other types of neuronal behaviors.

Moreover, there is currently a tendency to merge constructive and evolutionary approaches. The former follow the general strategy of starting up with a minimal architectural template and then going on annexing new units and adjusting them in accordance with the others already embodied, until a desired solution could have been found [361]. In such realm, Ensley and Nelson [139] implemented another version of the Cascade-Correlation (CasCor) in which it is possible to design neural nets through the competitive fusion of neurons with different types of mapping functions. At each step, the new CasCor selects from a predefined finite set of transfer functions the one which best complements the work performed by the others.

Iyoda and Von Zuben [221] have also attempted to analyze the impact of configuring ANNs with different activation functions at the hidden layer by proposing an evolutionary hybrid architecture inspired by another constructive method (Projection Pursuit Learning—PPL). Such approach also incorporates distinct composition functions in the output layer (additive and/or multiplicative), guiding to a higher efficiency on the combination of the mapping efforts realized by the hidden neurons. Their algorithm relies only on a classical GA, not directly promoting for the cooperation among the units.

By other means, Potter [373] conceived the idea of replacing the gradient descent procedure typically used to adjust the unit placement by a competitive genetic algorithm. Therefore, successive runs of the GA can optimize successive hidden units. The competing entities, thus, are individual units rather than whole networks. The work depicted here follows another direction: To promote the cooperation of these building blocks by means of coevolution.

The roll of aspects indicated in the previous section for the coevolutionary approach seems very suited for simulating cooperation and/or competition behavior among neural entities. Following this premise, Whitehead and Choate [479] have devised a cooperative-competitive genetic scheme for the automatic specification of two RBF parameters, centers and variances, when applying such kind of feedforward neural net to time series prediction. In such attempt, a GA operates on a population of competing basis functions $\phi_i(\|\mathbf{x} - \mathbf{c}_i\|)$, where $\|\cdot\|$ is typically the Euclidian norm and $\mathbf{x}, \mathbf{c}_i, d_i$ are the input patterns vector, the radial basis center and a scaling factor, respectively. The whole population corresponds to a single RBF net. Hence, individual RBFs, although competing with each other, are forced to cooperatively model the desired function over the domain of interest, sharing the mapping liabilities.

Through another line of reasoning, Zhao and others [510, 512] have applied cooperative coevolution

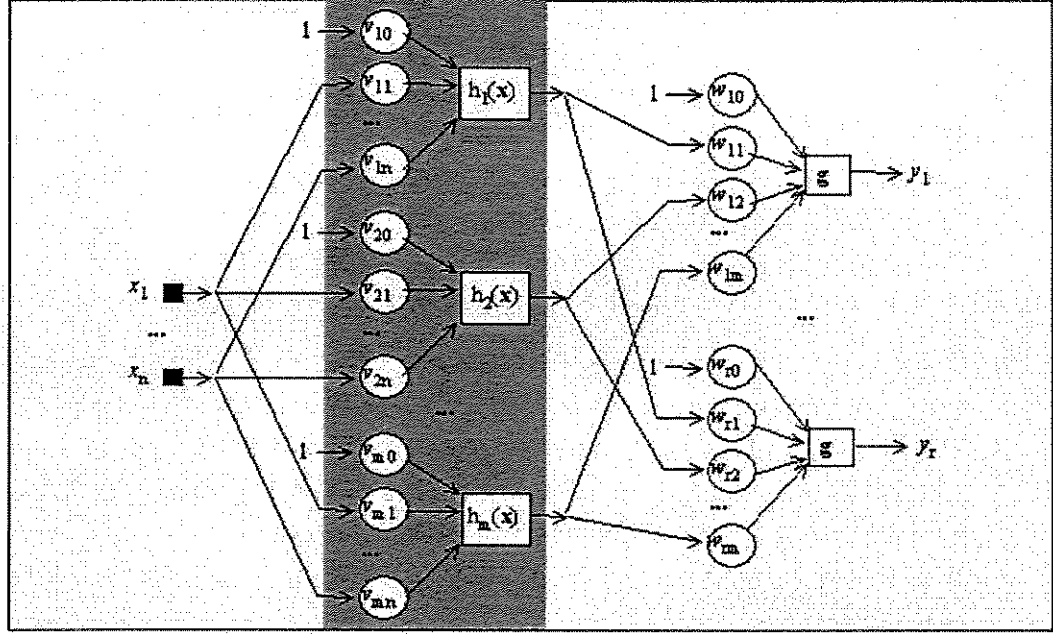


Fig. 8.1: Architecture of the heterogeneous neural network to be evolved by the hierarchical approach. The shadowed box indicates the components to be optimized by the coevolutionary process: neurons with different activation functions and the associated weights.

to modular neural networks, devising an algorithm in which each module is individually optimized by a separate GA. In this model, the blend of neural building blocks comes to be a coarser alternative when compared with the previous one.

Finally, it is worth to mention the prominent evolutionary methodology developed by Moriarty and Miikkulainen [334, 336]. SANE relates to a “symbiotic adaptive neuro-evolution system”, in which a population of homogeneous neurons is coevolved to compose a neural net devoted to be deployed on dynamic environments. This hierarchical solution attempts to optimize the neural topology by two means. First, since neurons are recognized as functional building blocks, their ensemble can be more accurately performed. Second, since no neuron is evaluated only by its own capabilities, but rather by the qualities of the groups in which it takes part, evolutionary pressure exists to evolve several complementary neuron types. The architecture shown in the next section was inspired by such guidelines.

IV. HCGA Approach

Here, we first present the main design deliberations taken during the conception of the HCGA approach, when applied to heterogeneous neural nets. Then we assess its adequacy and performance through a series of benchmarking results over well-known pattern classification problems.

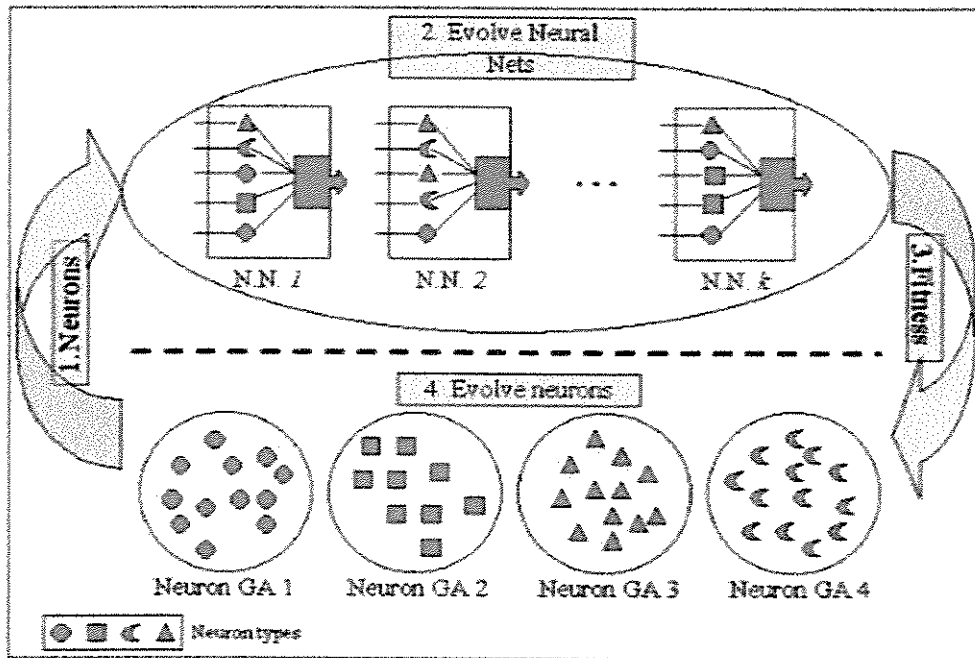


Fig. 8.2: Process of hierarchical evolution of the neural networks.

IV.1 HCGA Architecture

Fig. 8.1 shows the heterogeneous neural network model behind the HCGA proposal. It resembles very much a typical feedforward neural net; the main difference lies on the hidden layer as its neurons may have distinct activation functions ($h_j(\cdot)$). These functions are chosen from the delimited candidate set (which is extensible) presented in Table 8.1. The reason behind selecting such candidates is that they comprehend a broad parcel of those usually employed in the construction of neural nets, ranging from simple behaviors to various degrees of non-linearities. Besides the logistic (hyperbolic tangent) and Gaussian functions (as employed by Liu and Yao [295]), we opted to make use of steps, polynomials and other Gaussian-alike shapes.

As illustrated, $\mathbf{x}$ and $\mathbf{y}$ represent, respectively, the input vector of the training/test patterns and the resulting net output. The output neurons have all the same aggregation function $g(\cdot)$, implemented here as an additive linear combination ($\sum$), and the number of hidden neurons is predetermined. There are two sets of weighted connections, namely, $\mathbf{V}$ and $\mathbf{W}$. The latter is optimized during a supervised configuration process, through stimulus-response pairs, by means of the Least Mean Square (LMS) method. The adjustment of the weights in $\mathbf{W}$ is defined in order to solve the following optimization problem:

$$\min_{\mathbf{W}} \|\mathbf{y} - \mathbf{s}\|^2,$$

Tab. 8.1: Set of activation function candidates.

Candidate	Activation Function
Linear	$f_1 = x$
Step	$f_2 = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases}$
Hyperbolic Tangent	$f_3 = \frac{e^{sx} - e^{-sx}}{e^{sx} + e^{-sx}}$
Gaussian	$f_4 = e^{-\frac{x^2}{2\sigma^2}}$
Cauchy	$f_5 = 1 + \frac{x^2}{2\sigma^2}$
Multiquadric	$f_6 = \sqrt{f_5}$
Inverse Multiquadric	$f_7 = \frac{1}{f_6}$
Polynomial	$f_8 = x^n$

where $\|\cdot\|$ is the Euclidian norm and

$$\mathbf{y} = \mathbf{H}\mathbf{W} = \begin{bmatrix} \mathbf{h}_1(x_1) & \mathbf{h}_2(x_1) & \dots & \mathbf{h}_m(x_1) \\ \mathbf{h}_1(x_2) & \mathbf{h}_2(x_2) & \dots & \mathbf{h}_m(x_2) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{h}_1(x_p) & \mathbf{h}_2(x_p) & \dots & \mathbf{h}_m(x_p) \end{bmatrix} \mathbf{W}$$

Given the training set $\{\mathbf{x}_i, \mathbf{y}_i\}_1^p$, the LMS will then try to minimize the sum of the squared errors produced by each of the p input-output patterns. The $\mathbf{H}$ matrix is obtained, in this case, after the definition of m transfer functions chosen among the candidates given by Table 8.1 and applied to the p patterns thereupon. The optimal solution for the output weights is given by

$$\mathbf{H}^T \mathbf{H} \mathbf{W} = \mathbf{H}^T \mathbf{s} \rightarrow \mathbf{W} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{s},$$

where $(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$ is the *pseudo-inverse* of $\mathbf{H}$, which shall only exist if $\mathbf{H}$ has a non-deficient rank.

By other means, the shadowed box in Fig. 8.1 indicates the components of the heterogeneous net to be optimized directly by the HCGA approach. As depicted in Fig. 8.2, we conceived a hierarchical evolutionary architecture composed of two levels. In a bottom level, a coevolutionary approach was selected in order to train the networks by adjusting both the activation function parameters of the hidden neurons and the incoming input weights. In order to promote the cooperative behavior among the neurons that participate in a given net (aiming at the optimized sharing and division of responsibilities) and, at the same time, detect the most promising entities across the various nets, each unity receives its fitness according to the average fitness of the networks wherein it engages. Each kind of neuron can be viewed as a distinct evolving species and is allotted to a separate GA customized to represent its attributes.

In an upper level, a conventional GA is used to build the net topology, that is, to choose neurons (possibly with distinct features) to compose the hidden layer. The same neuron, thus, is allowed both to integrate various neural architectures and to appear more than once in the same structure. Neural nets

compete against each other across generations and are refined through the iterated LMS optimization process applied to their output weights.

Appendix A is dedicated to the HCGA pseudo-algorithm. The number of generations considered for benchmarking purposes relates to the upper GA cycles. The initial populations of both kinds of GA (upper and bottom) are set in a random manner (zero-average and uniform distribution). At the upper level, however, we decided to allow for the creation of new individuals in an intermediary population as a means to increase the diversity through generations. This intermediary population is also randomly generated.

At each cycle of the upper level GA, the evaluation of the neural net individuals is made over a different pattern test set, in such a way to incorporate their degree of generalization. Their fitness is inversely proportional to the mean squared error produced over the test patterns and the selective pressure is elitist generation-based, that is, the k better individuals from the last generation are not replaced by less adapted offspring [190]. To allow the net refinement phase, we decided to employ two selection operators, one for the progressive fine-tuning and the other for actual best net maintenance, whose parameter rates are given in Table 8.2 (NNET_ELITIST and NNET_SURVIVAL_RATE).

The chromosomes associated with neural nets are codified as follows: Each gene represents a logical link to a given neuron pertaining to a certain bottom GA. In this way, it is possible to have the same neuron appearing more than once in a given net. As the order of the neurons in the hidden layer is irrelevant, the uniform mutation operator was employed as it allows for the simultaneous swapping of n neurons at a time (for us, n was limited to 2). The recombination operator used was the *one-cut* crossover.

At the bottom level of evolution, each of the possible activation functions has an associated GA. By this means, it is possible to codify the peculiarities of each transfer function in a proper chromosome template. The fitness of the neural nets (set at the upper level) is propagated to all neurons that integrate them; in this way, the fitness of a neuron is given by the average fitness of the nets they participate. (Neurons that do not take part in any neural topology for NEURON_NON_FITNESS_HIST GA cycles are replaced.) Typically, the codification of the neuron chromosomes comprehends two slots: one for the input weights and other for the neuron's function parameters. The mutation and crossover operators actuate on one of those two slots each time, with equal probability.

Tab. 8.2: System configuration parameters: NNET refers to the upper level GA and NEURON to the bottom level GAs.

Parameter	Meaning	Value
NNET_POPSIZE	Number of networks in GA	50
NNET_MAX_GEN	Maximum number of generations for the neural nets	20
NNET_EXPANSION_RATE	Percentage of new elements to be inserted in the intermediary population during the neural nets refinement phase	50%
NNET_ELITIST	Number of best nets copied to next generation (elitist selection)	40%
NNET_SURVIVAL_RATE	Number of best nets whose neurons are not changed on next generation	5%
NNET_CROSSOVER_CHANCE	Chance of crossover occurrence	50%
NNET_MUTATION_CHANCE	Chance of mutation occurrence	25%
NNET_MAX_MUTATION_SIZE	Maximum number of exchanged neurons	2
NNET_NUM_HID_NEURONS	Number of neurons at the hidden layer	15
NEURON_POPSIZE	Number of neurons in each GA	20
NEURON_MAX_GEN	Number of generations for the neurons (and the whole algorithm)	50
NEURON_EXCHANGE	When to switch between selection operators (in percent of generation)	35%
NEURON_ELITIST	Number of best neurons copied to next generation (elitist selection)	40%
NEURON_BICLASSIST1	Number of best neurons copied to next generation (biclassist selection)	30%
NEURON_BICLASSIST2	Number of worst neurons copied to next generation (biclassist selection)	20%
NEURON_CROSSOVER_CHANCE	Chance of crossover occurrence	25%
NEURON_MUTATION_CHANCE	Chance of mutation occurrence	1%
NEURON_NON_FITNESS_HIST	Maximum number of generations until the replacement of a particular neuron that has not been chosen for the composition of any net	10

Tab. 8.3: Number of neurons of each type on the best network for each data set. **C#** stands for the *Card* data set and **H#** for the *Heart* data set.

	C1	C2	C3	H1	H2	H3
Linear	3	1	2	2	5	4
Step	1	6	3	3	3	7
Hyperbolic Tangent	1	2	3	2	—	1
Gaussian	3	2	4	2	2	1
Cauchy	2	—	1	3	1	—
Multiquadric	2	2	1	2	2	—
Inverse Multiquadric	3	2	1	1	1	2
Polynomial	—	—	—	—	1	—

IV.2 Results

The two pattern classification problems considered here were obtained from the PROBEN1 benchmarking repository [379], allowing the comparison of our proposal with others. The first problem, *Card*, refers to the task of approving or not the delivery of credit cards to particular customers, taking into account their profiles. The database has 690 samples of possible customers (patterns), with 51 input parameters (net input attributes) and two possible output responses (yes or no). 44% of the customers have good profiles and there are some absent data referring to some attributes, something that hinders the classification process. We used 518 samples to train the networks and 172 to test them. Those samples were used in 3 different orders (predefined in PROBEN1).

The *Heart* problem lies in the area of predicting cardiac diseases by observing some clinical cases (patients' health conditions). In this repository, there is a bunch of 920 patient samples (patterns) each composed of 35 inputs (personal data) and two outputs (prone to cardiac problems or not). Some attributes are also missing, making the decision even harder. We used 690 samples for training and 230 to test the networks. Here again we used the samples in 3 different, predefined orders.

Table 8.3 shows the configuration of the best network evolved for each problem. As can be noted, some kinds of neurons appeared more frequently than others. This is in line with what is expected: Since we are facing a classification problem, a Step neuron is typically more suitable for such a task than a Polynomial would be.

Figure 8.3 shows graphically the fitness of the best network and the average fitness of all networks for each data set along generations. The best network converges very fast due to the elitist selection used by the GA [190]. The average fitness oscillates continuously along generations because every time the neurons are evolved the networks owning them have to be readjusted.

Table 8.4 compares the results achieved by many neural network architectures [391] and our HCGA approach for the same data sets. HCGA performs better than all other approaches in most of the cases,

Tab. 8.4: Compared results (percentage of correctly classified patterns) between best instances of HCGA and other ANNs. Standard deviation in parentheses.

	C1	C2	C3	H1	H2	H3
MLP	86.00(1.03)	81.00(0.86)	81.00(1.19)	80.00 (0.96)	82.00 (1.14)	76.00(1.12)
RBF	88.00(0.67)	82.00(0.70)	83.00(0.70)	82.00(1.20)	82.00(1.72)	79.00(0.48)
CasCor	84.0(2.48)	79.00(1.87)	82.00(1.97)	82.00(1.96)	80.00(1.66)	74.00(1.97)
Tower	84.88(2.41)	78.49(2.91)	79.65(2.26)	78.70(2.73)	77.80(3.11)	70.87(3.36)
Pyramid	86.05(1.87)	79.07(2.97)	79.07(3.45)	79.57(2.93)	78.26(2.69)	76.09(2.38)
DistAI	84.35(2.32)	84.06(4.10)	82.90(3.54)	79.13(4.03)	80.87(2.92)	82.61(2.71)
PerCasc	87.58(2.12)	86.80(2.97)	84.07(2.11)	82.72(2.64)	82.56(3.03)	80.96(2.02)
Upstart	90.70(2.03)	86.05(1.96)	83.72(2.53)	84.05(2.48)	81.45(2.06)	80.78(2.65)
Tiling	80.23(2.74)	79.05(2.48)	79.07(2.08)	76.52(1.98)	71.64(2.02)	77.39(2.18)
HCGA	90.12(1.52)	87.79(2.03)	88.95(2.29)	88.26 (1.84)	85.65(1.75)	81.30(2.89)

showing its great generalization capabilities when applied to classification problems. For instance, the HCGA improvement over the second best ANN for the Card-3 problem reaches nearly 30%, taking into account the classification error rate.

Liu and Xiao [295] also achieved good results when comparing their GNN approach with common MLPs and Gaussian MLPs in classification problems. Through a series of experiments, the evolved GNN performed better, exhibiting lower means of the average test set error against the ones produced by the MLP and Gaussian MLP. Nevertheless, the benchmarking repository used for their analysis was a modified version of the *Heart* database, composed only of 303 data samples, with only 270 patterns effectively exploited.

Iyoda and Von Zuben [221], as well, attained good outcomes when comparing the performance of their hybrid architecture against the typical MLP and PPL approaches in approximation problems. Nonetheless, the computational costs related to their proposal constantly reached huge amounts, something that may become a bottleneck when applying the technique to hard problems.

IV.3 Future Work

Some improvements could be undertaken as future work. To combine other constructive techniques with ours in different manners seems suitable. For instance, to employ a CasCor implementation to produce a net that would serve as an initial start-up template to generate the initial population of neural nets at the upper HCGA level. This would comprise a typical sequential arrangement of strategies.

Likewise, it would also be possible to incorporate the idea established by Iyoda and Von Zuben [221] of evolving distinct composition functions in the output layer (additive and/or multiplicative). Other enterprises include: (i) The possibility of also augmenting or pruning [385] the number of hidden neurons at each generation, given space to the arising of “extend-shrink” behaviors; (ii) The employment of other EAs to

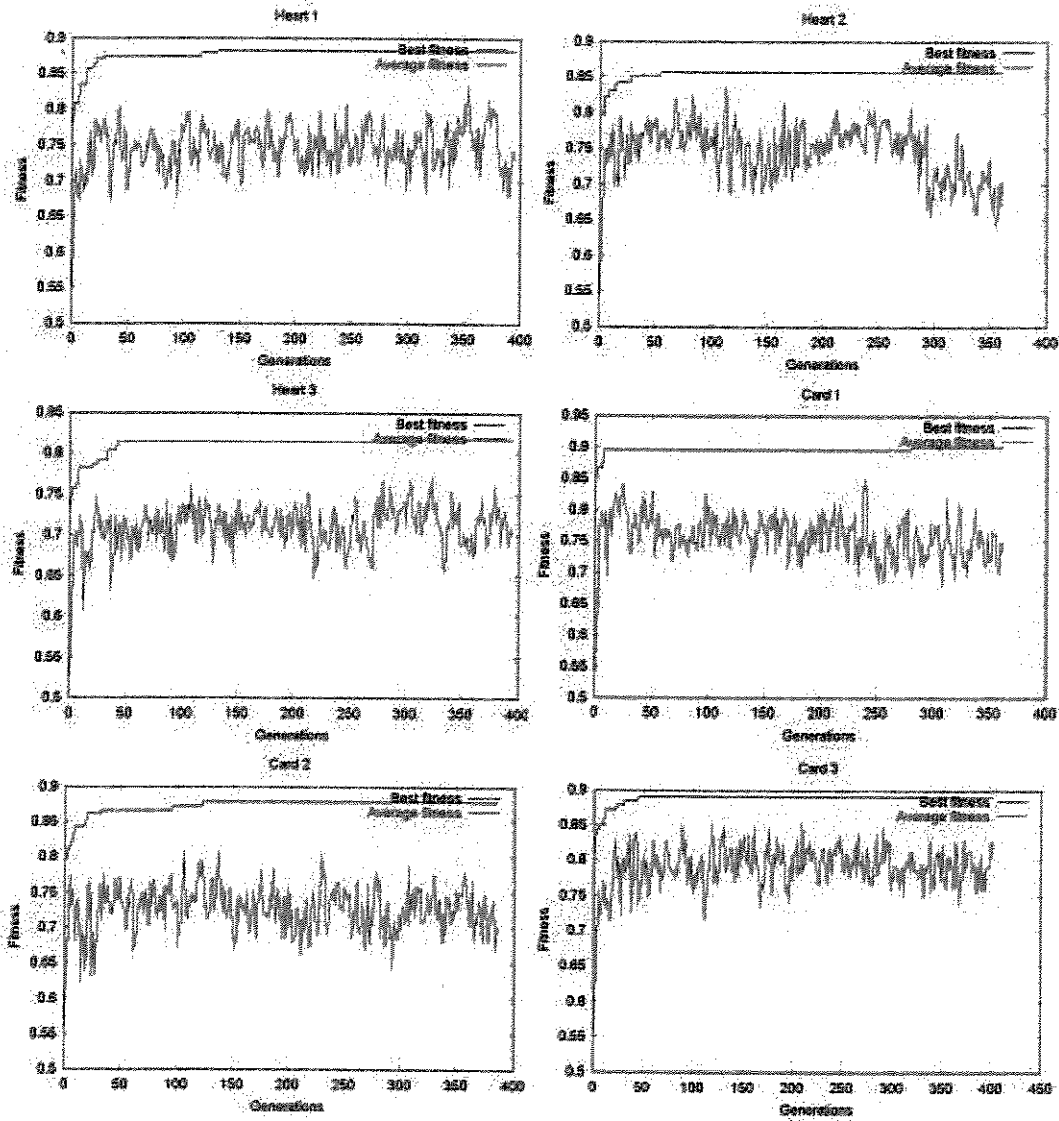


Fig. 8.3: Results showing the evolution of the best neural network and the average fitness of the networks along generations.

evolve the neuronal subpopulations; (iii) The expansion in the number of levels in the hierarchy; and (iv) The application of cooperative coevolution to more coarser building blocks.

V. Conclusion

This work has examined the suitability of merging together into the same framework various promising approaches proposed recently for the configuration of neural net architectures, particularly involving coevolution. The resulting model (HCGA) comprehends a new hierarchical-based evolutionary scheme devoted to the progressive assembling of heterogeneous neural structures. This blending strategy was assessed through a series of benchmarking tests over classification problems. The findings obtained so far corroborate other results already presented in the literature, showing that HCGA constitutes a promising design strategy in the direction of fully automatic adjustment of an extended set of neural networks parameters.

Appendix A. HCGA Pseudo-algorithm

```

1. Create initial population;
   Create neuron populations;
   Create neural net population;
2. Evolve neural nets refinement;
   do
     Create intermediary population (expansion);
     Evaluate neural networks
       Train (adjust weights via LMS)
       Test (compute NN_error);
     Compute fitness;
       fitness ← - NN_error;
     Elitist Selection;
     Mutation;
     Adjust the expansion rate;
   while( subGenCount < NNET_MAX_GEN &&
         (bestFit > lastBestFit || avgFit > lastAvgFit) );
   Set survival flag on neurons of the NUM_SURVIVE best networks;
   Cutoff the exceeding neural nets;
   Propagate fitness to neurons;
     Neuron fitness ← avg. of NNets it participated;
3. Evolve neurons;
   if( subGenCount < NEURON_EXCHANGE )

```

```

        Biclassist Selection;
    else
        Elitist Selection;
        Mutation;
    end
4. Repeat steps 2. and 3. for NEURON_MAX_GEN times

```

8.1.1 Considerações adicionais

Posteriormente à realização deste trabalho, foram conduzidos alguns experimentos, ainda bem incipientes, explorando a idéia alternativa de se encapsular subredes heterogêneas de menor porte (com, no máximo, quatro ou cinco neurônios escondidos) para atuarem como funções de ativação dos neurônios intermediários de uma RNA de mais alto nível. Essa idéia, denominada *aninhamento* (*nesting*) por este autor [93], teve como inspiração a existência de grupos (*clusters*) neuronais no cérebro humano exibindo processamento em diferentes níveis de resolução e especialização [69, 224, 342, 394]. Aplicou-se uma variação da abordagem HCGA em que, em vez de populações de espécies diferentes de neurônios, utilizou-se populações de espécies distintas de subredes heterogêneas, cada uma com um *template* (i. e., ordem dos tipos de neurônios) preestipulado. Resultados estatisticamente validados ainda carecem de ser realizados como forma de se avaliar o real potencial dessa extensão⁽²⁾. Contudo, considerando o notável desempenho exibido pelos modelos correlatos COVNET [168] e MOBNET [167], estes discutidos à página 157, passa-se a crer que a idéia de HCGA aninhado deva também trazer bons frutos em termos de desempenho para modelos neurais.

Outra extensão da arquitetura HCGA (de nome HCGA_2) foi proposta e validada mais recentemente por Weingaertner *et al.* [473], onde um terceiro patamar evolutivo passou a ser introduzido na hierarquia original (ver Fig. 8.4), ficando este responsável pela definição automática dos *templates* de redes heterogêneas completas. A introdução desse terceiro nível trouxe algumas vantagens adicionais ante à primeira versão, como detectado pelos autores. Dentre as quais, destacam-se duas: (i) há agora a possibilidade de se realizar, dinamicamente, tanto o aumento (*growing*) como a eliminação (poda) do número de neurônios da camada intermediária de redes heterogêneas — na versão original, só poda evolucionária era permissível; (ii) com a separação dos diferentes tipos de *templates* em diferentes instâncias de AGs, possibilitou-se uma maior especialização dos papéis realizados pelos neurônios, já que ficaram estimuladas, de antemão, interações mais localizadas entre as suas populações. Os resultados apresentados comprovaram a eficácia dessa extensão, confirmando as potencialidades da abordagem coevolutiva HCGA como um todo. Um protótipo combinando as extensões HCGA_2 e *nesting* coevolutivo ainda não foi implementado, mas é pensado para o futuro.

Cabe ainda salientar que, no artigo apresentado, o desempenho mensurado para a abordagem HCGA foi resultado de $N_s = 5$ execuções. Esse baixo número se deveu ao fato de o tempo computacional exigido, para

⁽²⁾Em verdade, pode-se considerar a idéia de *nesting* coevolutivo como um meio-termo entre as abordagens HCGA e COHENN (Cap. 7).

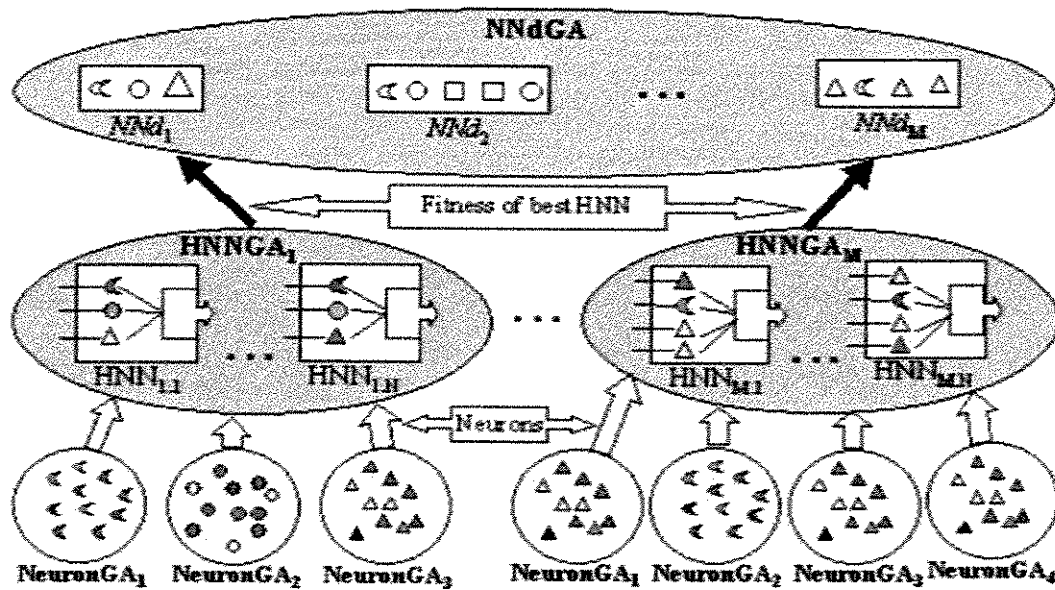


Fig. 8.4: Arquétipo da abordagem hierárquica HCGA_2 proposta por Weingaertner *et al.* [473]. O terceiro nível fica responsável pela evolução de diferentes descrições (*templates*) de redes heterogêneas (isto é, número de neurônios e seus tipos associados). As diferentes instâncias de AGs da primeira camada provêm neurônios para as diferentes instâncias de redes heterogêneas compondo o segundo patamar, ao passo que o *fitness* de cada *template* da última camada é dado pelo *fitness* do melhor modelo de rede heterogênea pertencente à sua população associada.

fins de simulação, ter sido bem elevado, chegando à ordem de um ou dois dias para uma única execução⁽³⁾. Para a abordagem HCGA_2, como esperado, os custos computacionais foram ainda maiores, havendo necessidade, na época, de recorrer-se a máquinas com alta capacidade de processamento, estas disponibilizadas pelo CENAPAD [473].

8.2 Seleção automática de modelos de SVM

O seguinte trabalho, cujo título é “*GA-based selection of components for heterogeneous ensembles of support vector machines*”, foi publicado como artigo completo nos anais de 2003 da *IEEE International Conference on Evolutionary Computation* [87]. Ele versa sobre a caracterização de um algoritmo evolutivo customizado (GASel) atuando como mecanismo auxiliar de seleção e combinação de diferentes instâncias de SVMs (cada uma com um par $\langle \text{função-kernel}, \text{função de perda} \rangle$ distinto) para fins de configuração automática de *ensembles heterogêneas de SVMs* (HE-SVMs) [284, 287, 289]. O desempenho por trás de GASel é avaliado comparativamente com mecanismos de seleção alternativos em problemas de classificação.

I. Introduction

Among the many possible machine learning (ML) techniques one could employ for the construction of high-performance classifier systems, Support Vector Machines (SVMs) have shown to be a great candidate mainly because of their ability to generalize well while untangling complex input data relationships. The appeal of SVMs lies in their strong connection with the underlying *statistical learning theory*: From this perspective, they constitute an approximate implementation of the *structural risk minimization* method [463]. Basically, SVMs follow the simple idea of nonlinearly mapping input data into high-dimensional feature spaces wherein a linear decision surface discriminating two different regions is designed. This decision surface is produced as the optimal hyperplane that separates the two classes with maximal margin (amongst the functions the machine can implement, such surface relates to the one with minimal *capacity* [97] in accordance with the amount of available training data). The optimal hyperplane can be uniquely shaped by resolving a constrained quadratic programming (QP) problem whose solution is given in terms of a subset of the training patterns that lie within the maximal margin of separation (*support vectors*).

No computation in the feature space is actually performed since SVMs rely upon the direct application of nonlinear *kernel functions* over the input data. The basic task of a kernel, defined as the dot product of two mapped patterns, is to realize feature extraction in a computationally-efficient manner. The flexibility behind the kernel's format allows an SVM to concurrently search across a wide variety of hypothesis spaces; thus, the choice of the best kernel function in line with the problem at hand seems to be crucial in order to leverage

⁽³⁾Para os experimentos, um protótipo da arquitetura foi implementado em Java (JDK 1.1.4), sendo as simulações realizadas em estações Sun-UltraSparc em rede (ou seja, não-dedicadas) configuradas com processadores de 167MHz e 256 MB de memória RAM.

SVM's performance. In such regard, we have formerly conceived *heterogeneous ensembles of SVMs* (HE-SVMs) [284, 287]. In contrast to other ideas that center around data resampling for SVMs [251], our approach stipulates that different component SVMs with distinct kernel functions, such as polynomials or radial basis functions (RBFs), may be created and properly combined into the same learning machine structure. HE-SVMs, thus, promote the automatic configuration of SVMs, and have shown better generalization capability when compared with the single SVM (S-SVM) approach [284]. Typically, the HE-SVM methodology implies (1) the separate training of each single SVM; (2) the selection of more appropriate components to integrate the ensemble; and (3) the combination of the outputs of the selected SVM models to generate the ensemble's outcome. Through this methodology, it is possible to experimentally ascertain which kernel type(s) is (are) more suitable to tackle the given classification/regression task.

In this work, we have focused mostly on how to find better SVM components for the automatic assembling of HE-SVMs, since the model selection process has direct impact on the level of generalization achievable by any learning machine [246]. Formerly [287], the component selection procedure was designed as two complementary steps, namely (i) the ordering of the components in consonance with a pre-established performance criterion; and (ii) the choice of the optimal subset of components to be combined. In such context, we have applied both the selection method proposed by Perrone and Cooper (P&C) [369] (which employs the mean squared error (MSE) over a detached selection dataset as the performance criterion) and another selection method based on the Vapnik-Chervonenkis (VC) dimension [100, 463]. Although showing considerable results, the approach behind steps (i)-(ii) is biased to recruiting the best-ranked components and its performance is directly dependent on the quality of the chosen selection dataset. Even though the VC dimension has beaten MSE as selection criterion (step (i)) and is more compliant with the structural risk minimization philosophy, the sequential analysis of the ordered list in the step (ii) may be too greedy for granting the diversity underlying the component-error distributions and, therefore, should incur performance loss.

With this in mind, in this paper, we have applied an evolutionary mechanism (named GASel) for optimizing the HE-SVM's component selection task. Genetic algorithms (GAs) have shown good behavior as automatic component designers [281] as well as model selectors [403, 515] for multiclassifier systems. In the HE-SVM scope, GASel has the advantage of merging steps (i)-(ii) into the same global search process and being robust to the largeness/roughness of the space of possible component selections. By now, a subset of S-SVMs (codified as a GASel chromosome) has its fitness given by the convex sum between the ensemble's error produced after applying the associated combination operator (majority voting) and a term that measures the diversity underlying the component's error distributions.

For performance assessment, genetically-configured HE-SVMs are compared with those produced with VC-dimension-based model selection and P&C's model selection by means of some complicated pattern classification experiments. These comprehend both UCI benchmark datasets [54, 151] and the bioinformatics task of cancer recognition based on gene expression data generated by DNA microarrays [13, 188]. Simulation results confirm our expectation that GASel comes to be a high-quality automatic mechanism for producing

more accurate HE-SVM settings.

The paper is organized as follows. Section II brings an overview of HE-SVMs, presenting the main steps behind the formalization underlying SVMs and comparing our approach with others that also make use of ensemble methods for SVMs. In Section III, we focus on the problem of automatically selecting components for integrating HE-SVMs, wherein we introduce the nuances underlying GASel alongside other three alternative methods. In Section IV, we assess the performance of GASel in three non-trivial pattern classification experiments. Section V concludes the paper.

II. Heterogeneous Ensembles of Support Vector Machines

Given a training set of N data points $\{\mathbf{x}_k, y_k\}_{k=1}^N$, where $\mathbf{x}_k \in \mathbb{R}^n$ is the k -th input pattern and $y_k \in \mathbb{R}$ is the k -th output pattern, the SVM approach aims at constructing a classifier of the form

$$y(x) = \text{sign} \left(\sum_{i=1}^N \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \right) \quad (8.1)$$

where each α_i , $i = 1, \dots, N$, is a positive real constant and b is a real constant. Towards this aim, the SVM designer assumes that

$$\begin{cases} \mathbf{w}^T \phi(\mathbf{x}) + b \geq +1 & \text{if } y_k = +1, \\ \mathbf{w}^T \phi(\mathbf{x}) + b \leq -1 & \text{if } y_k = -1 \end{cases}$$

where $\phi(\cdot)$ is a (not explicitly constructed) nonlinear function vector that maps the input space into a higher dimensional space. For the non-separable case, variables ξ_k are introduced such that

$$\begin{cases} y[\mathbf{w}^T \phi(\mathbf{x}) + b] \geq 1 - \xi_k & k = 1, \dots, N \\ \xi_k \geq 0 & k = 1, \dots, N \end{cases} \quad (8.2)$$

According to the structural risk minimization principle, satisfying the following optimization problem (a.k.a. the *primal*) induces the minimization of the risk bound:

$$\min L(w, \xi_k) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \mu \sum_{k=1}^N \xi_k$$

subject to (8.2). The second term of the right-hand side of the equation above is commonly written as a function of the employed *loss function* (typically, the linear ϵ -insensitive). To solve this optimization problem, one should construct the Lagrangian

$$\begin{aligned} L_1(\mathbf{w}, b, \xi_k, \alpha_k, \gamma_k) &= \frac{1}{2} \mathbf{w}^T \mathbf{w} + \mu \sum_{k=1}^N \xi_k - \\ &\sum_{k=1}^N \alpha_k \{ [\mathbf{w}^T \phi(\mathbf{x}_k) + b] y_k - 1 + \xi_k \} - \sum_{k=1}^N \gamma_k \xi_k \end{aligned}$$

by including Lagrange multipliers $\alpha_k \geq 0$, $\gamma_k \geq 0$ ($k = 1, \dots, N$). The solution comes as the saddle point of the Lagrangian and is given by solving the following constrained QP problem:

$$L_2(\alpha_k) = -\frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) + \sum_{k=1}^N \alpha_k \quad (8.3)$$

such that $0 \leq \alpha_k \leq \mu$, $k = 1, \dots, N$, $\sum_{k=1}^N \alpha_k y_k = 0$.

The nonlinear mapping $\phi(\mathbf{x}_k)$ in (8.3) is related to the kernel function $K(\mathbf{x}, \mathbf{x}_k)$ by imposing $\phi(\mathbf{x})^T \phi(\mathbf{x}_k) = K(\mathbf{x}, \mathbf{x}_k)$, which is motivated by Mercer's Theorem [100, 463]. The classifier (8.1) is, finally, designed by solving the *dual*

$$\min L_2(\alpha_k) = -\frac{1}{2} \sum_{i,j}^N y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{k=1}^N \alpha_k \quad (8.4)$$

subject to the restrictions in (8.3). The weights can be set as $\mathbf{w} = \sum_{k=1}^N \alpha_k y_k \phi(\mathbf{x}_k)$. Because the matrix associated with this QP problem is nonsingular, the solution to (8.4) will be global.

Furthermore, for a fixed pre-defined basis, one can show that the optimal hyperplane generated by (8.4) that satisfies the constraint $\|\mathbf{w}\| \leq a$ has a VC dimension h that is bounded by $h \leq \min([R^2 a^2], n) + 1$, where $[\cdot]$ denotes the “integer part” operator and R is the radius of the smallest ball containing the points $\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_N)$. For the generalization error using hyperplanes in the feature space, VC theory can be used to obtain the upper bound $\xi \leq O(R^2/NC^2)$ and $C^2 = 1/\|\mathbf{w}\|^2$, where C is the class separation margin.

As an extension to this conventional view, where a typical SVM archetype employs only one learning machine setting as the classifier hyperplane in a high-dimensional space, we have proposed the combination of M classifiers (estimators) f_i as an ensemble of SVMs [284, 287]. An ensemble is a collection of learning machine components whose individual decisions to the same task are combined in some manner to produce a better outcome [191]. This usually implies that the classifiers are produced by different means and show decorrelated error patterns over a training dataset, in such a way as to improve the overall group's generalization capability.

Kim *et al.* have also explored the idea of “SVM ensembles” [251]. However, they have specified the independent training of each individual SVM through a replicated dataset produced via a bootstrap method, viz. Bagging. By other means, Valentini and Dietterich [460] have investigated how bias-variance (BV) decomposition could offer further guidance for developing ensemble methods using SVMs as base learners. In their study, the authors have “analyzed the learning properties of SVMs using BV decomposition of the error as a tool to understand the relationships between kernels, kernel parameters, and learning processes in SVM.” Yet, they have conducted BV analysis only over different variants of the same kernel type (i.e. by employing different kernel parameters) in order to produce bagged ensembles of selected low-bias SVMs. By recognizing the important role of the kernel functions as similarity measures for the construction of the high-dimensional feature space, we have considered the case where each classifier f_i in the ensemble is an

S-SVM instance with a disparate kernel type (whose parameters are set via cross-validation). Hence, distinct similarity measures can work concurrently and cooperatively in the feature extraction process in order to enhance the HE-SVM classification decision.

The simultaneous training of all component SVMs has the advantage of employing only one QP problem involving all kernel types. However, for huge amounts of input data, this might turn to be an inefficient process since an HE-SVM requires resources that are proportional to the square of the product between the number of single SVMs and the number of training samples. To circumvent such drawback, we have applied in this work a separate QP problem for each kernel type. Moreover, previously, all component SVMs were associated with the same loss function (in charge of modeling the data noise level) [463], namely the linear ϵ -insensitive loss function. Here, following the premise that the performance of the loss function is implicitly dependent upon the characteristics of the application data at hand, we have considered, in the experiments reported in Section IV, other loss function types as well, specifically the quadratic ϵ -insensitive loss function and the Huber loss function.

III. Automatic Component Selection for HE-SVMs

Model selection entails any procedure applied over a learning algorithm in order to minimize the generalization error of the resulting learning machine by balancing its complexity with its performance [246]. In the HE-SVM perspective, the model selection process is viewed as choosing ensemble components among the various S-SVM models produced separately. Such an idea is in compliance with the arguments given recently by Zhou *et. al* [514, 515], who have formally stated that, in some cases, ensembling an appropriate subset of individual components (in their case, neural networks) might incur superior performance when compared to common schemes of aggregating all components at hand.

From the HE-SVM standpoint, the model selection problem may be formulated as follows: Given the population $P = \{1, 2, \dots, M\}$ of previously- and independently-trained heterogeneous S-SVMs, one should choose the subset $P^* \in 2^P$ (with cardinality $Card$ in the range $[1, \dots, M]$) whose elements, when properly combined, produce the best (optimal) ensemble outcomes for the problem in sight— 2^P stands for the power set of P . In this sense, the optimal component selection to be searched is very dependent on the chosen optimality criterion, which justifies our idea of investigating different criteria (see subsections below).

In the experiments reported in Section IV, we have applied simple majority voting (MV) [191] over the selected components as the combination method. Voting is the most common method used to aggregate classifiers and has also been employed towards multiclassifier combining through GAs [403]. In MV, the final output is the label that receives more than half of the votes of the ensemble components.

III.1 Perrone and Cooper's Model Selection (PCSel)

In an ensemble of classifiers, as we increase the size of the component population P , the assumption that all deviations from the true solution are mutually independent does not hold anymore. When this assumption fails, adding more components to the group incurs loss of computational resources, since this will not improve the ensemble performance. Moreover, this can be harmful in the sense that we may include components with very bad performance, jeopardizing the execution of the whole resulting classifier. Thus, it would be very appealing to find the subset P^* indicating the optimal group of components. One could try all $2^M - 1$ possible non-empty subsets of 2^P but, for large M , this search becomes unmanageable. As an alternative, Perrone and Cooper [369] advocated to order the elements of the population according to increasing MSE values and then to generate a set of $m \leq M$ classifiers by successively adding the ordered elements of P . In each component addition trial, it is verified whether the new ensemble's MSE is diminished or not. In positive case, the classifier is kept; otherwise, it is discarded.

III.2 VC dimension based Model Selection (VCSe1)

The VC dimension (h) provides an extensive measure (according to the statistical learning theory) to assess the generalization capability of an SVM. In a previous work [287], we have investigated its applicability in component selection as well, serving as ordering criterion for ranking the S-SVM components. Such approach has the advantage of not being prone to the problem of choosing the best selection dataset and has not shown the PCSel tendency of indicating only the local performance of each component. Consequently, VCSe1 has outperformed PCSel in prior experiments [287]. In this paper, we have also considered as a subcase of VCSe1 the application of the maximal margin (C) separation as ordering criterion, when the sphere radius R (as defined in Section II) is limited—we call this variation as MMSel.

III.3 Genetic Algorithm based Model Selection (GASel)

Genetic algorithms [30] constitute a global optimization technique that has been shown to be successful in highly complex domains, and therefore seems to be appropriate for dealing with the complexity issues underlying the space of possible S-SVM combinations (2^P). Such issues include discreteness, non-differentiability, deceptiveness (small variations in the combinations may imply great changes in fitness value) and multi-modality (different S-SVM ensembles showing equal performance). In such context, Zhou *et. al* [514, 515] have also applied a GA-based methodology to search for optimal selections of neural network components, and the resulting combinations achieved generalization performance superior to those produced through well-known ensemble methods, such as Boosting, Arcing, and Bagging (for classification) and weighted-averaging (for regression).

In GASel, the chromosomes are encoded as binary strings with length M wherein each bit (gene) indicates the presence (1) or absence (0) of an S-SVM in the ensemble. The initial population is randomly

crafted (via uniform distribution) and successive generations undergo a set of transformations over selected individuals by means of standard genetic operators (one-point crossover and simple mutation). Elitist selection (with a generation gap of 10% of the population) is employed jointly with the stochastic universal sampling (SUS) selection operator [30]. The individuals are selected to breeding in accordance with a quality-assessment rule, known as fitness function, given by:

$$Fitness_k = \beta \cdot MSE_k + (1 - \beta) \cdot \frac{1}{N} \sum_{i=1}^N y_i(\mathbf{e}_k) \quad (8.5)$$

where the first term on the right-hand side of the expression above refers to the accuracy of the ensemble corresponding to the k -th chromosome (represented by vector $\mathbf{e}_k$) over a detached selection dataset (the “degrees of activation” of the outputs of an ensemble, one for each class, are taken as the average of the activations of its components), and the second term takes into account how diverse are the components with regard to their error patterns, to be measured according to the following rule taken from [403]:

$$y_i(\mathbf{e}_k) = \begin{cases} 0 & \text{if } \sum_{j=1}^M y_{ji} e_{kj} \leq \left\lfloor \frac{\sum_{j=1}^M e_{kj}}{2} \right\rfloor, \\ 1 & \text{if } \sum_{j=1}^M y_{ji} e_{kj} > \left\lfloor \frac{\sum_{j=1}^M e_{kj}}{2} \right\rfloor \end{cases} \quad (8.6)$$

where y_{ji} is the transformed binary output (i.e. “0”, for misclassification, and “1”, otherwise) of the j -th S-SVM for the i -th data sample, whereas $e_{kj} \in \{0, 1\}$ indicates exclusion or inclusion of this component. The parameter β , which should be fine-tuned according to the problem at hand, provides a tradeoff measure between the two aforementioned fitness parcels. Moreover, to promote parsimony in the genetic design of the final ensemble, we have followed the minimum description length (MDL) principle of model selection [246] and, only in case of fitness ties between two or more distinct HE-SVM arrangements, we have introduced a penalization charge (proportional to the number of selected components) to discriminate lengthy combinations from those with more moderate sizes.

IV. Experiments

Four kinds of ensembles were considered in the succeeding experiments, one for each component selection method. For PCSe1, VCSe1, and MMSe1, we have followed the steps of the algorithm in Fig. 8.5, while, for GASE1, the algorithm in Fig. 8.7 was used. The list of kernels adopted includes *Linear*, *Polynomial*, *RBF*, *E-RBF*, *Sigmoid*, *Fourier*, *Linear Spline*, *Bn-spline*, and *Anova-spline* [287]. As for each of these nine kernel types it is possible to apply one of three loss functions (see Section II), there are $M = 27$ S-SVM models with chance to be selected/combined to produce P^* ; that is, the size of the search space S is $O(2^{27})$. In the experiments with GASE1, we have set $MAX\_GEN = 100$, $size(\pi) = 100$, $P_{cross} = 0.7$, and $P_{mut} = 0.007$ (Fig. 8.7). Regarding all methods, the simulation results were achieved through a series of 10 trials for each method.

1. Repeat the Steps 2–6 M times;
2. Generate training, selection and test datasets;
3. Normalize input data according to each kernel;
4. Find the best kernel parameters (using 10-fold cross-validation);
5. Train each S-SVM with its associated loss function and selected parameters;
6. Calculate the number of misclassified patterns (or bound on the error generalization) for each kernel over the selection dataset;
7. Rank the component SVMs from best to worst according to the given ordering criterion and pick the best;
8. For each remaining kernel in the ordered list (using the selection dataset):
 - (a) Add it temporarily to the current ensemble and calculate the output with MV;
 - (b) If the number of misclassified patterns decreases, keep with it; otherwise, discard it.
9. Calculate the output of the resulting ensemble over the test dataset;
10. Obtain the mean and standard deviation over the set of correctly-classified patterns.

Fig. 8.5: Algorithm for PCSe1, VCSe1, and MMSe1 experiments.

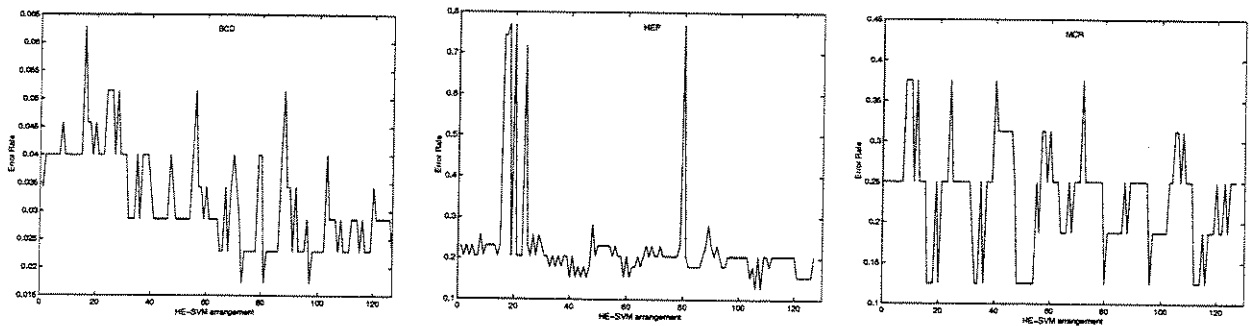


Fig. 8.6: Features of the search spaces in BCD699, HEP, and MCR experiments: “Neighbor” ensemble configurations (in Hamming space) showing different error rates while quite dissimilar arrangements may present similar performance.

1. Repeat the Steps 2–5 M times;
2. Generate training, selection and test datasets;
3. Normalize input data according to each kernel;
4. Find the best kernel parameters (using 10-fold cross-validation);
5. Train each S-SVM with its associated loss function and selected parameters;
6. Randomly creates an initial population π of chromosomes (component selections);
7. Calculate the population fitness according to Eq. (8.5);
8. While ($num\_gen \leq MAX\_GEN$) do
 - (a) Stochastically select mates for reproduction according to their fitness;
 - (b) Apply recombination operator (crossover) for generating childhood according to crossover rate P_{cross} ;
 - (c) Apply simple mutation over each bit of the population, with mutation rate P_{mut} ;
 - (d) Calculate the population fitness according to Eq. (8.5).
9. For the set of fittest ensembles in the last generation (possibly many):
 - (a) Calculate their output over the test dataset;
 - (b) Rank them according to their performance and pick the final best.
10. Obtain the mean and standard deviation values of the final best ensemble over the set of correctly-classified patterns.

Fig. 8.7: Algorithm for GASel experiments.

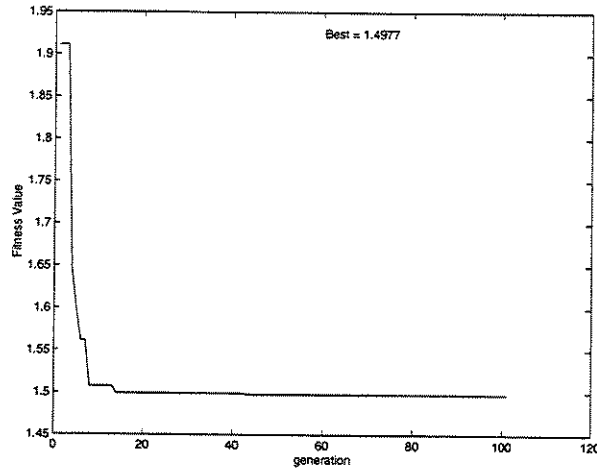


Fig. 8.8: A typical running of GAsel for BCD699.

Before delving into details about each experiment alone, it is worth to analyze graphically the “physical” aspects underlying the space 2^P of possible component combinations. Fig. 8.6 brings the error rates of 127 ($= \sum_{i=0}^6 2^i$) contiguous HE-SVM arrangements located in the range [$< 000...0000001 > - < 000...1111111 >$], which were produced by varying only one S-SVM component at a time (Hamming distance of one). Although this is not the actual space searched through the fitness function of GAsel (Eq. 8.5), it provides a good dimension of the hurdles intrinsic in the HE-SVM component selection problem. It is quite noticeable that the search space in all three problems show multi-modality (several arrangements with equal performance), local minima, as well as deceptiveness (great error rate variations by changing only few bits) issues.

IV.1 UCI Breast Cancer Diagnosis

The “breast cancer diagnosis” (BCD699) problem, whose dataset is made available in the Internet, relates to the task of distinguishing between patients that are more likely to develop breast cancer from those that are not, taking into perspective histological data produced by biopsies taken from members of both groups. The dataset consists of 699 samples each one with nine parameters. Each feature has been rated on an integer 10-scale. Of the 699 patterns, 458 are indicated as benign (65.5%) and 241 are indicated as malignant (34.5%). In the experiments, we have used the same data partition of the work of Lee *et al.* [281], and the training, selection and test datasets have 349, 175, and 175 examples, respectively.

Figure 8.8 shows the convergence process underlying a typical running of GAsel while optimizing HE-SVMs in a BCD699 experiment. Several times the best individual is discovered in a generation much

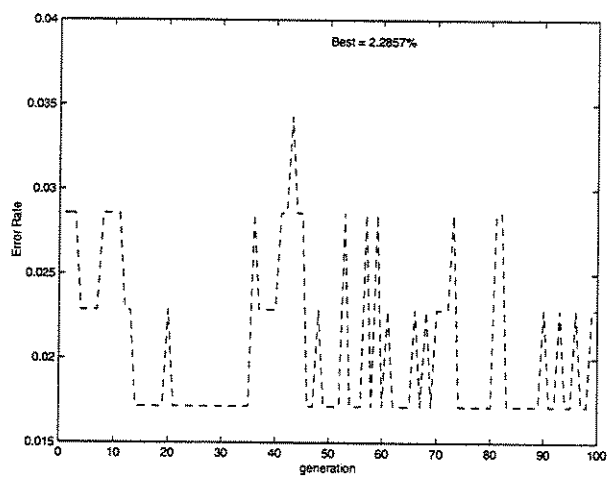


Fig. 8.9: Variation of error rate through a running of GASel—BCD699.

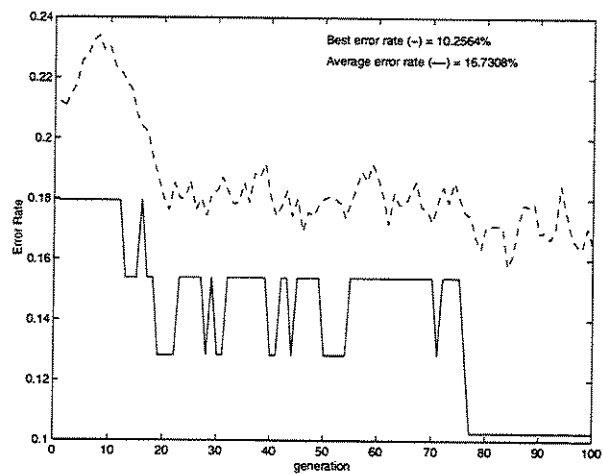


Fig. 8.10: Variation of error rate through a running of GASel—HEP.

Tab. 8.5: Results for different HE-SVM component selection methods for BCD699 (test dataset)

	Mean	Std. Dev.	Best
Random Search	95.81	1.51	97.15
PCSel	96.38	1.19	97.71
VCSel	96.57	0.99	97.71
MMSel	96.57	0.99	97.71
GASel ($\beta = 0.5$)	97.14	0.57	97.71

Tab. 8.6: Results for different HE-SVM component selection methods for HEP (test dataset).

	Mean	Std. Dev.	Best
PCSel	82.11	5.06	89.47
VCSel	80.00	4.40	86.48
MMSel	81.58	4.56	89.47
GASel ($\beta = 0.5$)	82.63	3.00	86.48

earlier than the last one. The aim behind Fig. 8.9 is to depict the variation in the ensembles' error rates through time in the same GASel trial as in Fig. 8.8; since the fitness function (given as in Eq. (8.5)) does incorporate a parcel that tries to maximize diversity within the ensembles, the average error rate in each generation may not show a decaying behavior as in Fig. 8.8. Oftentimes a less diverse but very accurate ensemble may be replaced by others showing better diversity levels, although with inferior recognition rates. Such issue should be controlled by a proper setting of the tradeoff parameter (β).

Table 8.5 brings a comparative analysis between the HE-SVM settings produced by means of the different component selection methods. Also included (to serve as a baseline of comparison) are the results obtained through extensive random search of the optimal HE-SVM configuration (1,000 iterations per running).

IV.2 UCI Hepatitis Dataset

The hepatitis (HEP) dataset relates to a 2-class problem with 155 samples and the task is to predict, according to a patient's profile, whether he/she is prone to die. There are 19 input attributes and 167 missing values, something that increases the problem difficulty. The class distribution of 32/123 is kept through the data partition sets for training (78), selection (39), and test (38).

Figure 8.10 shows the variation in time of the HE-SVMs' error rates (best and average rates) of a particular GASel trial, whereas Table 8.6 brings the results accomplished with the HE-SVM structures produced through the different selection methods.

Tab. 8.7: Results for different HE-SVM component selection methods for MCR (test dataset).

	Mean	Std. Dev.	Best
PCSel	82.67	3.65	86.67
VCSel	82.67	3.65	86.67
MMSel	82.67	5.96	86.67
GASel ($\beta = 0.5$)	84.00	3.65	86.67

IV.3 Cancer Recognition via Microarray Gene Expression Data

This experiment concerns the cancer identification task based on gene expression data (MCR) investigated by Alon *et al.* [13]. This problem is an extremely difficult one for many methods, since the feature dimension is usually very high, and the training samples are usually very scarce. Gene expression information was extracted from DNA microarray data, resulting, after pre-processing, in a table of 62 tissues X 2,000 genes expression values. The matrix contains the expression of the 2,000 genes with highest minimal intensity across the 62 tissues. Since there were no defined training and test sets, we split randomly the data into 31 samples for training, 16 samples for component selection, and 15 samples for test. Table 8.7 depicts the results attained with HE-SVMs produced with the different component selection methods.

IV.4 Further Discussion

The results shown in Tables 8.5–8.7 confirm our expectation that the HE-SVM approach with GASel as model selector could produce performance results quite comparable to those achieved by means of the PCSel and VCSel (MMSel) methods. As we have evaluated all $M = 27$ S-SVM models in the selection dataset prior to the running of GASel, the computational time spent in a GASel trial was not too higher than that produced by any of the other methods. Moreover, GASel has the advantage of reporting different component configurations with equal performance over the selection dataset in the final generation, something that may be used to indicate experimentally which types of kernel functions should be employed in the construction of SVM-based learning machines for coping specifically with the problem at hand. Actually, the classification performance achieved with all HE-SVM configurations (irrespective of the chosen component selection method) is very significant and is comparable to the best available in the literature [151, 188, 281].

V. Concluding Remarks

In this paper, we have focused on the model selection process underlying the HE-SVM approach. In this sense, we have verified through complicated pattern classification problems that employing an evolutionary mechanism for the simultaneous selection and combination of components comes to be a promising strategy for configuring more accurate ensembles of SVMs. The classification performance achieved by all

variants of HE-SVM (albeit the extra computational requirements underlying all steps in HE-SVM simulations) was impressive, mainly if one concerns the bioinformatics task. As future work, we shall investigate the employment of resampling methods (like Bagging and Boosting [251]) as a means to improve the HE-SVM performance. Likewise, automatic fine-tuning of the GASE1 tradeoff parameter (β) as well as the application of different types of combination operators (besides MV) [281] shall also be investigated.

8.2.1 Considerações adicionais

Vale aqui um comentário sucinto sobre o porquê de se ter usado diferentes funções de perda quando da configuração das 27 funções-*kernel* adotadas nas experimentações. Mesmo considerando somente problemas de classificação, o papel da função de perda (responsável por mensurar o risco empírico) parece ser bastante relevante na “calibragem” da capacidade de aproximação da função-*kernel*, sendo que diferentes funções de perda podem implicar distintos desempenhos para uma mesma função-*kernel*, conforme salientado por Smola [436]. Com uma função de perda apropriada, pode-se diminuir o número de SVs “detectados” pelo *kernel*, acarretando, daí, a diminuição da complexidade do modelo final. Dessa forma, desconsiderou-se, na fase de *treinamento* das 27 instâncias de SVM, a função de decisão (*hard*) $\text{sign}(\cdot)$, passando-se a trabalhar como se para um problema de regressão — ou seja, levando-se em conta a saída de “ativação” (*soft*) de cada máquina.

Embora não se tenha mostrado nesse artigo, o desempenho final das melhores instâncias de HE-SVMs (i. e., suas taxas de estimação correta), com ou sem mecanismo evolutivo de seleção/cominação, tem sido constantemente superior àquele apresentado por seus melhores componentes. Isso considerando tanto problemas de regressão como de classificação, o que vem a corroborar os resultados apresentados no Cap. 7. As Refs. [284, 287, 289, 291] trazem mais informações a esse respeito. Entretanto, uma observação é pertinente: por se tratarem de estimadores mais elaborados (i. e., menos instáveis — ver Cap. 4) que redes neurais, máquinas simples de vetores-suporte *já* tendem, de antemão, a apresentar desempenho bem satisfatório, geralmente superior àquele apresentado por modelos neurais sofisticados (inclusive, redes heterogêneas). Desse modo, foi costumeiro notar, nas várias experimentações, que as configurações finais de HE-SVMs envolveram a combinação de poucos componentes (no máximo, duas ou três instâncias de SVMs com *kernels* distintos) [284, 289], haja vista que cada um dos modelos já apresentavam, por si só, desempenho de estimação bem substancial, equiparados, muitas vezes, uns aos outros.

8.3 Misturas de especialistas evolutivas

No próximo trabalho, rotulado de “*Hybrid genetic training of gated mixtures of experts for nonlinear time series forecasting*” e publicado como artigo completo nos anais da edição de 2003 da *IEEE International Conference on Systems, Man and Cybernetics* [88], passa-se a descrever um instrumento evolutivo dirigido à configuração automática de instâncias completas de misturas de especialistas engatilhadas (ou seja, definição

do número, das topologias e dos pesos das RNAs-especialistas, bem como das saídas da rede mediadora), sendo estas aplicadas a problemas de previsão de séries temporais não-lineares compostas por observações de variáveis físicas ou econômicas. Mediante o emprego de uma codificação especial, baseada em genes reguladores (vide AGs estruturados, introduzidos na Seção 6.5), a abordagem proposta, HGT-GAME, viabiliza o crescimento e a poda dinâmicos de especialistas, de tal modo a casar a complexidade da composição final àquela do problema em vista. Ademais, para alavancar o desempenho do método evolutivo, lançou-se mão de uma estratégia híbrida lamarckiana (ver Seção 6.6), por meio da qual um refinamento local é feito em cada cromossomo, pai ou filho (*gating* + especialistas), a cada geração. Para tanto, foi embutido o algoritmo de retropropagação do erro (descida do gradiente de primeira ordem — ver Subseção 2.4.2) dentro do passo M do algoritmo EM (vide Seção 4.2). Resultados de experimentos são reportados, cotejando o desempenho de instâncias de MEs evolutivas, produzidas e treinadas via HGT-GAME, com aquele obtido por instâncias convencionais de MLPs e MEs engatilhadas, cujas topologias são fixadas *a priori*.

I. Introduction

Mixture of Experts (ME) models [239] account for modular neural network architectures devised to decompose complex problems into simple subtasks. From the statistical viewpoint, an ME should be considered as a mixture model for estimating conditional probability distributions. In this learning framework, a gating network (GN) is liable to (softly) partition the input space into regions to be each assigned to a given expert network (EN). In terms of a mixture model, EN outputs correspond to conditional component densities whereas the GN generates input-biased mixture coefficients. With such assumption, it is possible to train MEs by means of the *Expectation-Maximization* (EM) algorithm.

The standard ME model comprises a single-layer of perceptrons with soft-max activation functions as GN and ENs with linear activation functions. Besides, there exist two variants, namely *Localized MEs* and *Gated MEs*, that have gained much recent attention. The former, introduced by Xu *et al.* [496] and later deployed by Ramamurti and Ghosh [381], accomplishes problem decomposition by employing soft hyper-ellipsoids (in place of soft hyperplanes) produced via normalized Gaussian kernels. The second variant, formulated by Weigend and others [472], uses a single nonlinear GN to coordinate competing nonlinear ENs (typically, multilayer perceptrons—MLPs) for dealing with different regimes of a dynamic process. The advantage of using a nonlinear gating lies in the possibility of generating more complex time/space decompositions through nonlinear decision boundaries, something very attractive to cope with the nonstationarity and overfitting issues that are common in nonlinear time series prediction.

Moerland [328] described and assessed within a uniform formalization different training methods, such as the gradient descent, for minimizing aspects of the ME error function. In this work, we investigate another training mechanism, based on *Genetic Algorithms* (GAs) [30], toward the automatic structural design and parameter configuration of Gated MEs. In HGT-GAME, we represent a whole ME instance (GN + ENs) into a given chromosome. For tackling nonlinear prediction problems, we have followed the guidelines of

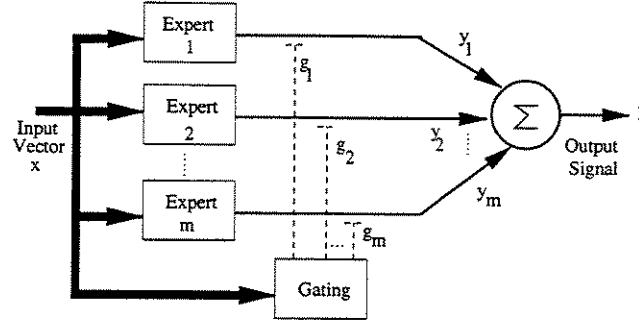


Fig. 8.11: A typical ME architecture.

Weigend *et al.* [472] and, besides the codification of all GN and EN weights, we also put the ENs' associated variances (one for each output) into optimization. Also, by making use of *regulatory genes* [108], which allow the exclusion/inclusion of full blocks of genetic code relative to ENs, our approach makes possible the automatic pruning and growing of experts [381]. In order to improve HGT-GAME's exploitation/exploration capabilities, as the space of feasible ME architectures may be too large to be dealt with, a local search refinement (a.k.a *hybrid training* [481]) is performed over each HGT-GAME ME chromosome via the gradient descent algorithm (*Backpropagation*—BP), in such a manner as to further fine-tune each ME's configuration parameters (values for weights and variances).

In what follows, we (i) overview some formalization issues behind ME models, commenting on their application to time series prediction; (ii) introduce the main design aspects underlying HGT-GAME; and (iii) evaluate the behavior of HGT-GAME-trained Gated MEs when tackling some non-trivial forecasting problems. The predictive performance of the resulting models are contrasted with that attained with BP-trained MLPs, as well as with Gated MEs produced solely via gradient descent [328].

II. Mixtures of Experts for Time-Series Prediction

ME models are particularly useful for coping with mapping problems in which the form of the mapping is different for distinct regions of the input space [239, 381]. Even though applying a single network could provide a reasonable global solution for these problems, the strategy of assigning different expert subnetworks to distinct regions seems to be more effective in such undertaking. This is accomplished in MEs (Fig. 8.11) by employing a separate gating module in charge of deciding, based on the input data and its degrees of confidence on the experts' capabilities, which of the competing experts should be used to determine the overall system output.

Formally stating, given an input sample $\mathbf{x}$, each expert k ($k = 1, \dots, K$) produces an output $\mathbf{y}_k$ in accordance with its local conditional probability function $\phi_k(\mathbf{t} \mid \mathbf{x})$, and then the final system output $\mathbf{y}$ is

given as the weighted sum of the estimated probabilities from all experts, namely,

$$\mathbf{y}(\mathbf{x}) = \sum_{k=1}^K g_k(\mathbf{x}) \mathbf{y}_j(\mathbf{x}), \quad (8.7)$$

where $g_k(\mathbf{x}) (g_k \geq 0, \sum_k g_k = 1)$ is the gating's confidence level on assigning the given input sample to the k -th expert.

Essentially, each expert is adaptively trained to extract relevant features of its associated space whereas the GN is trained to produce fair global weights for combining the experts' outputs and to softly partition the input space. The decoupled training between gating and expert modules becomes possible by resorting to the Expectation-Maximization algorithm [239], whose formulation to estimate the gating and experts' parameters comprehends two well-delimited steps defined over a log-based error function, given by

$$E = - \sum_{n=1}^N \ln \sum_{k=1}^K g_k \phi_k. \quad (8.8)$$

The idea is that the minimization of this error function can be simplified if each pattern could be allocated to exactly one EN (this is possible by utilizing some *missing variables* that equal to one for one component and zero for the others). The steps comprise:

- **E-Step:** Calculates the expected values for missing variables $h_k^{(n)}$ (assuming that all the other parameters are known):

$$h_k^{(n)}(\mathbf{t}^{(n)} | \mathbf{x}^{(n)}) = \frac{g_k(\mathbf{x}^{(n)}) \phi_k(\mathbf{t}^{(n)} | \mathbf{x}^{(n)})}{\sum_{k'=1}^K g_{k'}(\mathbf{x}^{(n)}) \phi_{k'}(\mathbf{t}^{(n)} | \mathbf{x}^{(n)})}. \quad (8.9)$$

- **M-Step:** The term to be minimized for GN is

$$E_{GN} = - \sum_{n=1}^N \sum_{k=1}^K h_k^{(n)}(\mathbf{t}^{(n)} | \mathbf{x}^{(n)}) \ln(g_k(\mathbf{x}^{(n)})) \quad (8.10)$$

and for the expert j is

$$E_{EN} = - \sum_{n=1}^N \sum_{k=1}^K h_k^{(n)}(\mathbf{t}^{(n)} | \mathbf{x}^{(n)}) \ln(\phi_k(\mathbf{t}^{(n)} | \mathbf{x}^{(n)})), \quad (8.11)$$

where ϕ_k is the conditional probability density of the k -th expert, given by:

$$\phi_k(\mathbf{t}^{(n)} | \mathbf{x}^{(n)}) = \frac{\exp\left\{-\frac{0.5(\mathbf{t}^{(n)} - \mathbf{y}_k^{(n)})^T (\mathbf{t}^{(n)} - \mathbf{y}_k^{(n)})}{\sigma_k^2}\right\}}{(2\pi)^{d_t/2} (\sigma_k^2)^{1/2}},$$

where d_t is the dimension of $\mathbf{t}$, σ_k^2 denotes the variance of the k -th expert, and $\mathbf{y}_k$ is a function of some of its parameters (usually, its connection weights).

For nonlinear time series prediction, Hong *et al.* [208] have already investigated the behavior of Localized MEs [496] composed of linear experts and trained via an evolutionary algorithm, namely, the *Evolutionary Structured Optimization* (ESO) method. The achieved ME models outperformed (in terms of the mean square error—MSE) Gaussian RBF networks produced using the same evolutionary method. Nevertheless, amongst the various ME parameters that may be susceptible to automatic adaptation via evolutionary computation, only those related to the gating module (centers and widths of probability density functions) were actually optimized by this means. Moreover, Hong *et al.*'s approach does not provide direct support to automatic pruning/growing of the constituent experts, relying upon an information theoretic measure (viz., Bhansali-Downham information criterion) to carry out parsimony in the ME architecture construction. Such issues were taken into account when designing the HGT-GAME mechanism (see Section III).

Weigend *et al.* [472] extended the standard ME framework for coping with forecasting problems by employing nonlinear ENs (MLP predictors) to be coordinated by a nonlinear gating network. By this means, the input space can be split nonlinearly through the hidden units of the GN (which is now in charge to discover the different temporal regimes and the switching between them), and the emergent, decomposed sub-processes can be nonlinear in nature through the hidden units of the ENs. Moreover, in Gated MEs, the noise-level parameters associated with each individual EN are also made adaptable. This is justifiable since it enhances the whole temporal segmentation process (areas of dissimilar predictability are grouped together) and also prevents overfitting (the experts learn to match their variances to their local noise levels).

The EM-based training of Gated MEs follow the same steps as in (8.9), (8.10), and (8.11), wherein the gating and expert parameters are typically obtained by employing a nonlinear optimization algorithm, such as gradient descent [328]. In the next section, we present an alternative, hybrid-genetic-based mechanism for the automatic structural and parametric configuration of Gated MEs.

III. Adaptive Gated MEs with HGT-GAME

Fig. 8.12 brings the pseudo-algorithm with the main steps behind the HGT-GAME approach for training Gated MEs. As an extended GA-based adaptive framework [30], the key conceptual issues underlying HGT-GAME comprise: (i) the (quasi-)hierarchical encoding of a whole Gated ME instance into a chromosome with the possibility to undertake automatic pruning/growing of ENs; (ii) the definition of the fitness function to discern between appropriate and non-appropriate individuals; (iii) the definition of the genetic operators (and their rates); and (iv) the possibility to perform local training refinement over the gating and expert modules.

III.1 Codification and Fitness Function

The HGT-GAME codification scheme is highly influenced by the ideas brought forth by Dasgupta and McGregor in the conceptualization of *Structured GAs* (sGAs) [108]. The main aspects of the latter approach are the use of genetic redundancy and the employment of a hierarchical structure for representing problem solutions as linear chromosomes. Regulatory genes, acting as switching operators to turn on or off whole blocks of genes, constitute the primary sGA mechanism for eliminating redundancy while interpreting and mapping genotypes into phenotypes. In sGA, binary genes at any level may be active or passive, and high-level genes activate or deactivate sets of lower level genes.

Such nonlinear interpretation of linear strings has two distinctive characteristics that seem particularly useful for the evolution of Gated ME architectures: (i) a single change at a higher level of the hierarchy enables multiple changes at lower levels, bringing about the possibility of large variations in the phenotypes from one generation to the other (hence, the ME adaptation process can exhibit patterns of evolutionary “jumps” across the fitness landscape); and (ii) although not active in one generation, important building blocks (a.k.a. “junk genes”) may still persist in the chromosome structure and be carried in a neutral and apparently redundant form to subsequent generations, something very interesting to ensure the continuous coadaptation of the ME chromosome’s parts and to maintain high levels of genetic variability (thus, preventing from premature convergence).

Fig. 8.13 shows how a Gated ME instance is represented as an HGT-GAME chromosome. As it is noticeable, there are only two levels in our hierarchy. At the higher level, there are K regulatory bits, where the k -th bit indicates whether the k -th expert (and its associated parameters, represented via real-coded genes at the lower level) will be considered (‘1’) or not (‘0’) in the phenotypic interpretation of an ME individual. (As there is only one gating for each ME, its parameters—connection weights—will always be “expressed.”) Through a series of dynamic prunings and insertions of expert modules alongside the adaptive process, such sGA-based codification scheme enables the automatic determination of the optimal ME structure in consonance with the problem at hand. Therefore, HGT-GAME presents itself as an alternative (evolutionary-based) ME constructive approach to the one proposed by Ramamurti and Ghosh (in which the removal and/or creation of ENs are directly supervised by the ME designer) [381].

As indicated in Fig. 8.12, in the minimization process behind HGT-GAME, we make use of the negative of the loglikelihood error function defined in (8.8) as the metric to assign fitness values among competing HGT-GAME chromosomes. This agrees with the ME philosophy, since (8.8) serves as an adequate (entropy-related) measure to evaluate the quality of the interactions between GN and ENs in the problem decomposition and solution.

1. Randomly generate the initial population (*Chrom*);
2. Decode *Chrom*, apply the EM algorithm, and then train each ME with gradient descent (BP)
 - (a) $New\_Chrom \leftarrow BPtrain(Chrom, \tau_1)$;
3. If (*Lamarckian Search*)
 - (a) $Chrom \leftarrow New\_Chrom$;
4. Evaluate *Chrom*
 - (a) $fit(Chrom) \leftarrow -Likelihood\ function$;
5. While ($gen \leq MAX\_GEN$)
 - (a) $fit(Chrom) \leftarrow Ranking(fit(Chrom))$;
 - (b) Offspring selection (GGAP);
 - (c) Offspring recombination (P_{cross});
 - (d) Offspring mutation (P_{mut});
 - (e) Train parents (*Chrom*) τ_1 times;
 - i. $\tau_1(gen) \leftarrow \tau_1(0) - \left\lfloor \tau_1(0) \times \frac{(gen)}{MAX\_GEN} \right\rfloor$;
 - (f) Train children (*Offspring*) τ_2 times;
 - (g) Evaluate *Chrom* and *Offspring*;
 - (h) If (*Lamarckian Search*)
 - i. $Chrom \leftarrow New\_Chrom$;
 - ii. $Offspring \leftarrow New\_Offspring$;
 - (i) Join *Chrom* and *Offspring* in a common pool and cutoff the worst individuals;
 - (j) $gen \leftarrow gen + 1$

Fig. 8.12: HGT-GAME's pseudo-code.

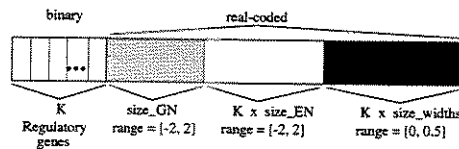


Fig. 8.13: HGT-GAME chromosome representation.

III.2 Genetic Operators and Rates

In HGT-GAME, the *Stochastic Universal Sampling* (SUS) algorithm [30] is applied to select individuals for mating. The idea behind SUS is to make a single draw from a uniform distribution and use this to determine how many offspring to assign to all parents, a strategy that minimizes chance fluctuations. Moreover, we have also made use of a linear ranking operator to better discriminate between the chromosomes' fitness values. Contrasted to conventional, steady-state GAs, in HGT-GAME not all individuals in the current population are replaced by their offspring, which is controlled by a generation gap (*GGAP*) constant, and the best among parents and children are then holded.

Novel ME chromosomes are produced via recombination and mutation, with rates given by P_{cross} and P_{mut} . As HGT-GAME follows a mixed "binary-real" codification scheme, dissimilar operators were applied to the different ME chromosome parts. For the binary (regulatory genes) portion, one-point crossover and simple bit mutation operators were used. For the real-coded (gating and experts' parameters) part, *arithmetic crossover* and an EP-based mutation operator were employed [30]. The latter stipulates a new value for a given variable by adding to its current amount a small random perturbation (with normal distribution) that fits within the variable's range. In contrast to recombination operators that exchange information between mates, arithmetic crossover attempts to blend, via convex sum, the component variables of two given parents, say x_{1i} and x_{2i} , resulting in a new individual x' given by: $x'_i = \alpha \times x_{1i} + (1 - \alpha) \times x_{2i}$, where $\alpha \in [0, \dots, 1]$.

III.3 Hybrid Training

The combination of *Lamarckism* and GAs (i.e. hybrid genetic search) has been explored as a promising way to enhance the efficiency achieved with common or customized genetic operators [481]. While GAs are most proficient at exploring (via mutation) and reconciling (through recombination) widely separated portions of the search space caused by scattered populations, Lamarckian search is more useful at exploring localized areas (big valleys or peaks) of the fitness landscape.

In HGT-GAME, we have exploited such potentialities in the evolution of Gated ME instances by applying, in each generation of the GA main cycle, the gradient descent algorithm at each module (gating or expert) of each chromosome separately. This is done in an iterative basis, where the several configuration parameters (weights and variances) of parents and offspring are locally optimized through BP for τ_1 and τ_2 iterations, respectively. In order to promote fairness when comparing parents with their offspring and also to avoid overtraining and loss of computational resources, the former parameter is made adaptive and shows a linear decaying behavior with rate proportional to the number of generations already executed (see Fig. 8.12).

IV. Results

We conducted three experiments with different nonlinear time series to compare the performance of Gated MEs trained via HGT-GAME to those trained only with BP, and also to MLPs. In the first, we assessed HGT-GAME in the prediction of the Mackey-Glass chaotic time series [208, 381], defined as $x_{MG}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t)$. The aim is to anticipate the value of a future point in $k = t + P$ having as background the points generated up to $k = t$. From $x_{MG}(t)$, we extracted 1,000 input-output data samples of the form $[x(t-18), x(t-12), x(t-6), x(t); x(t+6)]$, where $t = 118$ to 1,117. The first 500 samples were used for training and the rest for test. In the second experiment, we considered the Laser dataset, where the data partition was 1,000 samples for training and 1,000 samples for test [508]. Three contiguous values (lags) $[x(t-3), x(t-2), x(t-1)]$ were used as input for predicting the target value $x(t)$. The last experiment (Forex) involved the one step ahead prediction of the daily exchange rate between US Dollar and British Pound (USD/GBP) [16], in a total of 784 samples covering a period of three years (Jan/1991–Dec/1993), from which the first 600 observations were used as training data. As the actual rates series shows nonstationarity problems, we followed the guidelines of Anastasakis and Mort [16] and worked upon a modified time series (with a smoother behavior), whose form is given by $x_{FX}(t) = p_t - \frac{1}{5} \sum_{i=1}^5 p_{t-i}$, where p_t is the actual price of the USD/GBP exchange rate.

Aiming at uniformity in the comparison, we applied the parameters summarized in Table 8.8, where h_g and h_e are the number of hidden neurons of the GN and ENs, respectively, t_{EM} is the number of EM iterations, t_M denotes the number of training epochs used in the M-step, and K_M stands for the maximum number of ENs per ME instance. In HGT-GAME, t_{EM} refers to each generation while t_M is replaced by τ_1 and τ_2 . Conversely, simple MLPs comprise five hidden neurons and are trained until reaching 2,000 epochs. Moreover, we normalized all training/test datasets in the range $[-1, 1]$, and the forecasting performance measures⁽⁴⁾ were calculated in the resulting normalized series. For all produced models, the average performance in 5 trials was recorded.

From Table 8.9 and Figs. 8.14–8.17, it is noticeable that HGT-GAME was very effective in automatically assembling Gated ME structures for tackling the aforementioned forecasting problems. The obtained performance results were quite analogous to those shown by models produced exclusively with BP, despite the fact that the latter were trained exhaustively for more than 100,000 BP epochs and the number of BP iterations for each HGT-GAME chromosome was only 75 ($\tau \times t_{EM}$). Moreover, most of the times, the achieved performance by HGT-GAME in the test datasets were equivalent to that acquired in the training phase, reflecting the good generalization behavior of its produced models. Overall, the Gated ME results are quite comparable (or even better) to those shown in the literature [16, 208, 508].

⁽⁴⁾SSE stands for “Sum of Squared Errors,” MSE is the same as “Mean Square Error,” MAE refers to “Mean Absolute Error,” MAPE denotes “Mean Absolute Percentage Error,” RMSE means “Root Mean Square Error,” NRMSE stands for “Normalized Root Mean Square Error,” TheilCoef denotes “Theil’s Coefficient,” and MeanRev symbolizes “Mean Reversion.”

Tab. 8.8: Parameter settings in Gated ME experiments.

Parameter	Gated-BP	HGT-GAME
h_g, h_e	5 neurons	5 neurons
t_{EM}	200 steps	3 steps
$t_M (\tau_2, \tau_1(0))$	500 iterations	25 iterations
K_M	{3,5} experts (fixed)	5 experts
$Chrom$	—	25 strings
P_{mut}	—	30%
P_{cross}	—	75%
$GGAP$	—	0.9
MAX_GEN	—	50

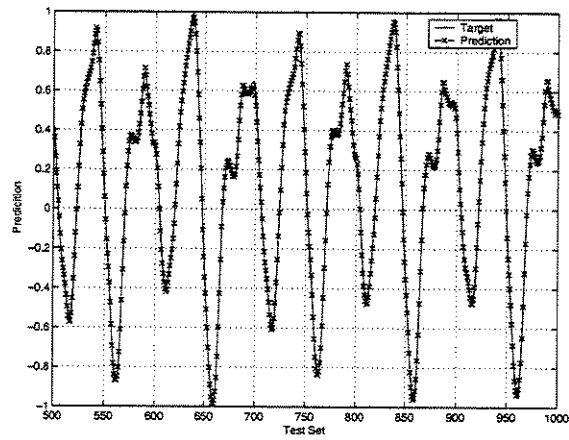


Fig. 8.14: Mackey-Glass Series Prediction (HGT-GAME).

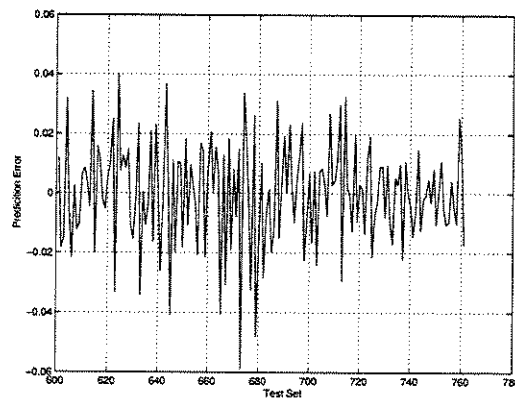


Fig. 8.15: Prediction error for Forex series (HGT-GAME).

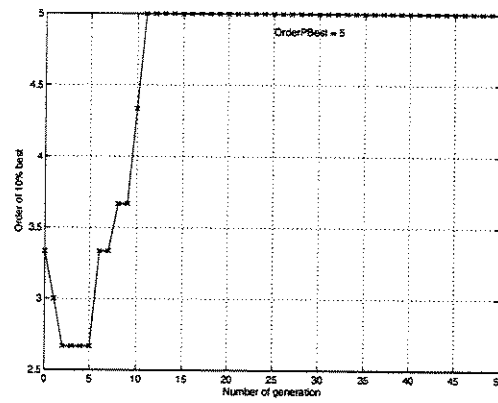


Fig. 8.16: Dynamic growing and pruning—Average number of experts in the 10% best MEs (Forex).

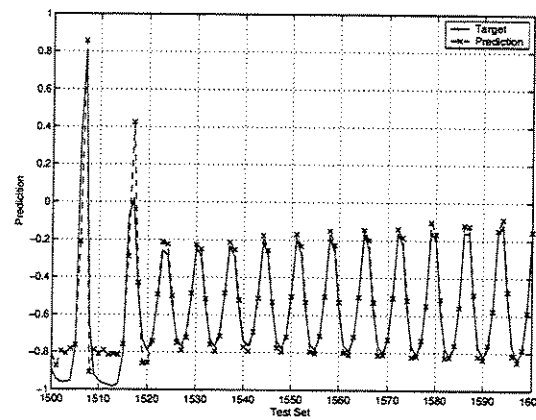


Fig. 8.17: Window of prediction for Laser time series (HGT-GAME).

Tab. 8.9: Average Results for Training/Test Datasets. In parentheses, the final number of ENs. *SD* means "Standard Deviation" calculated in relation to MSE.

	Training								
	SSE	MSE	MAE	MAPE	RMSE	NRMSE	TheilCoef	MeanRev	SD
	Mackey-Glass								
Gated-HGT-GAME(4) Gated-BP(3) Gated-BP(5) MLP	0.19237	0.00038	0.01523	15.33544	0.00088	0.03845	0.2648	0.0384	9.52%
	0.03074	0.00006	0.00607	8.31403	0.00035	0.01537	0.1059	0.0154	5.21%
	0.03141	0.00006	0.00568	4.76386	0.00035	0.01555	0.1070	0.0155	3.59%
	0.42851	0.00086	0.02384	14.10222	0.00131	0.05751	0.3952	0.0574	1.12%
Gated-HGT-GAME(5) Gated-BP(3) Gated-BP(5) MLP	0.22447	0.00037	0.01473	1.59792	0.00079	0.78256	1.0059	0.6610	13.33%
	0.18522	0.00031	0.01300	1.41109	0.00072	0.75574	0.9132	0.6001	4.80%
	0.18589	0.00031	0.01301	1.41209	0.00072	0.76076	0.9148	0.6011	3.94%
	0.32304	0.00054	0.01717	1.86215	0.00095	2.50417	1.2060	0.7925	1.07%
Gated-HGT-GAME(4) Gated-BP(3) Gated-BP(5) MLP	11.21716	0.01125	0.06865	66.69139	0.31452	0.2960	0.2865	0.00336	15.23%
	2.71745	0.00273	0.02180	19.97744	0.00165	0.14284	0.1457	0.1410	3.21%
	1.99276	0.00200	0.01868	18.43002	0.00142	0.12104	0.1248	0.1208	5.03%
	11.01982	0.01105	0.06979	69.43926	0.00333	0.30169	0.2934	0.2840	2.81%
	Test								
	SSE	MSE	MAE	MAPE	RMSE	NRMSE	TheilCoef	MeanRev	SD
Gated-HGT-GAME(4) Gated-BP(3) Gated-BP(5) MLP	Mackey-Glass								
	0.17965	0.00036	0.01489	10.78616	0.00085	0.03732	0.2558	0.0373	14.66%
	0.02921	0.00006	0.00593	6.94126	0.00034	0.01504	0.1032	0.0150	5.92%
	0.03093	0.00006	0.00546	4.25430	0.00035	0.01549	0.1062	0.0155	7.11%
0.42131	0.00084	0.02347	18.45716	0.00130	0.05721	0.3918	0.0571	2.3%	
Gated-HGT-GAME(5) Gated-BP(3) Gated-BP(5) MLP	Forex								
	0.04834	0.00030	0.01372	1.48286	0.00137	0.83542	0.9768	0.7641	11.28%
	0.04194	0.00026	0.01260	1.36130	0.00127	0.89032	0.9099	0.7118	4.88%
	0.04171	0.00026	0.01261	1.36239	0.00127	0.90925	0.9074	0.7098	4.14%
0.05670	0.00035	0.01514	1.63759	0.00148	2.64111	1.0579	0.8276	1.97%	
Gated-HGT-GAME(4) Gated-BP(3) Gated-BP(5) MLP	Laser								
	13.84519	0.01385	0.07534	64.11503	0.00372	0.33822	0.3157	0.3071	17.88%
	2.66299	0.00266	0.02312	26.44303	0.00163	0.13769	0.1384	0.1347	5.78%
	1.82348	0.00182	0.02033	14.57271	0.00135	0.11247	0.1146	0.1114	9.12%
13.93351	0.01393	0.07549	54.71098	0.00373	0.33622	0.3167	0.3081	3.26%	

V. Conclusion Remarks

In this work, we have presented a hybrid evolutionary method for the constructive design and training of Gated ME models. The results obtained so far, considering non-trivial forecasting problems (Section IV), give support to HGT-GAME's applicability as they validate its effectiveness. As future work, we shall extend HGT-GAME's framework to also encompass hierarchical [239] and Localized ME models [496]. Moreover, we shall also investigate the potentiality of applying "Baldwinian hybrid search" [481] for constructing MEs.

8.3.1 Considerações adicionais

Com o intuito de se avaliar ainda mais as potencialidades da abordagem HGT-GAME, novas simulações para as séries Forex e Laser foram conduzidas, considerando tanto treinamento híbrido lamarckiano como baldwiniano (ver Seção 6.6), bem como uma nova calibragem de alguns parâmetros importantes. A Tab. 8.10 apresenta um resumo dos resultados obtidos com esses novos experimentos (média sobre 5 simulações para cada configuração), sendo essa tabela complementada pelos gráficos das Figs. 8.18 e 8.19. Com base nesses dados, algumas observações importantes podem ser encaminhadas:

- É notório que o casamento entre o aumento no número de gerações do GA (MAX_GEN) e o aumento no número de iterações do algoritmo *Backprop* ($\tau_2, \tau_1(0)$), este dirigido ao ajuste dos parâmetros livres da *gating* e dos especialistas no passo-M, incorreu, em média, em maior ganho de desempenho para a abordagem HGT-GAME. Deve-se notar que agora o número máximo de iterações do *Backpropagation* para um certo cromossomo chega a ser de 5.000 ($= \tau_2 \times t_{EM}^f$), ou seja, ainda bem abaixo das 100.000 iterações realizadas para as arquiteturas MEs fixas. Todavia, essa nova configuração de tais parâmetros de controle acarretou um aumento substancial em termos de custo computacional, fazendo com que, em média, o tempo exigido para uma única execução da série Laser chegasse à ordem de dias⁽⁵⁾. A calibragem automática desses parâmetros de acordo com a instância de problema em mãos parece ser, pois, uma estratégia bem propícia para essa investigada no futuro, o que implicará novos ciclos de experimentações.
- No caso da série Forex, o aumento de desempenho foi notado tanto para a versão lamarckiana como para a baldwiniana, aproximando os resultados obtidos por HGT-GAME àqueles obtidos via MEs engatilhadas com números de especialistas preestabelecidos (Tab. 8.9) — para algumas métricas, houve até ganho substancial de desempenho.
- No caso da série Laser, os resultados obtidos revelam um ganho ainda mais notável de desempenho para a versão lamarckiana, algo não correspondido à altura pela variante baldwiniana. Ao se analisar, por exemplo, a Fig. 8.18(f), fica aparente que a rede mediadora da HGT-GAME baldwiniana parece não

⁽⁵⁾ É certo também que esse alto custo se deve, em muito, ao fato de o código ter sido implementado na linguagem interpretada da ferramenta Matlab (Versão 6.0).

ter conseguido arcar muito bem com as várias regiões de mudança de regime temporal exibidas por essa difícil série caótica, a qual foi empregada na competição realizada pelo *Santa Fe Institute* no começo da década passada [470]. De todo modo, o desempenho mostrado por essa versão baldwiniana é equiparável àquele exibido pelas MEs engatilhadas de tamanho fixo treinadas exaustivamente com *Backpropagation* (Tab. 8.9), superando a versão lamarckiana anterior. Ademais, verifica-se para essa série que houve, em média, um incremento no número de especialistas do modelo ME final para a versão lamarckiana (era quatro e passou a ser cinco), algo corroborado tanto pela versão baldwiniana como pelos experimentos conduzidos por Weigend *et al.* [472] — nesse último caso, os autores estabeleceram um número arbitrário de especialistas (de oito a 15) e verificaram que, ao final das simulações, somente cinco ou seis deles tinham suas portas da *gating* ativadas para algum padrão de treinamento.

- Embora os resultados devam ser considerados como de caráter preliminar, as colocações feitas nos parágrafos anteriores parecem endossar a conjectura elaborada por Ku *et al.* [257, 258] (ver à página 153) de que abordagens lamarckianas sejam mais apropriadas para lidarem com o treinamento de arquiteturas neurais contendo um número elevado de parâmetros livres (como é o caso das MEs investigadas), haja vista a dificuldade de se realizar simultaneamente a assimilação genética de tantas variáveis, em se tratando das versões baldwinianas. Por outro lado, os resultados parecem também confirmar a tese exposta na página 135 de que o desempenho relativo entre métodos baldwinianos e lamarckianos depende diretamente da instância de problema sob análise: para o caso da série Forex, pode-se dizer que, em princípio, ambas as variantes são equípotentes, algo não constatado para a série Laser.
- Após a publicação desse artigo, detectou-se, mediante o cálculo da métrica *TheilCoef* (a qual não constava na versão original do trabalho), que os comportamentos de *todos* os preditores implementados à época sobre a série Forex estavam sendo bastante similares àquele exibido pelo preditor trivial baseado na hipótese do caminho aleatório, tanto no que concerne ao treinamento como ao teste. (Para verificar isso, basta notar os valores próximos da unidade para esse critério de desempenho — vide Tab. 8.9 — e atentar para a discussão sintética à Subseção 2.4.2.) Os novos resultados para a abordagem HGT-GAME, tanto para a versão lamarckiana como para a baldwiniana, denunciam um certo ganho de desempenho no que tange a esse critério, principalmente na fase de treinamento. Ainda assim, em tese, parece de fato haver sentido em adotar a hipótese do caminho aleatório como um preditor de referência a ser superado quando da análise de séries econométricas [387]. No caso da série Laser, por sua vez, lançar mão de preditores triviais como esse e a Reversão média deverá surtir pouco ou nenhum efeito em termos de desempenho.
- As saídas de ativação das portas da rede mediadora, tanto para a versão lamarckiana (sob configurações antiga e nova) como para a baldwiniana, revelam um intenso chaveamento entre os especialistas quando da previsão da série Laser, indicando um modo interessante de decomposição do problema. Em vez de alocar especialistas distintos a regiões separadas do mapeamento temporal⁽⁶⁾, a aborda-

⁽⁶⁾Nas experimentações conduzidas anteriormente por Lima *et al.* [286], observou-se uma decomposição do mapea-

gem evolutiva HGT-GAME deu origem a *gatings* com comportamentos temporais quase que binários (“ON/OFF”) em suas portas. Tal comportamento foi observado também nas experimentações conduzidas por Weigend *et al.* [472], os quais provêem uma interpretação mais aprofundada desse tipo de segmentação baseada na dinâmica caótica da série⁽⁷⁾. De todo modo, tanto a versão lamarckiana como a baldwiniana parecem ter alocado dinamicamente um dos especialistas para ajudar especificamente no tratamento das regiões de transição entre os regimes da série — tais regiões de interrupção de crescimento contínuo do envelope do sinal são referidas como *colapsos* por Weigend *et al.* [472]. O fato de ser justamente nessas áreas que as MEs resultantes apresentem os seus maiores picos de erro (ver Figs. 8.18(a) e 8.18(b)) leva este autor a pensar que um aumento ainda maior nos valores dos parâmetros *MAX_GEN*, τ_2 e $\tau_1(0)$ pode vir a carrear melhores índices de desempenho para a abordagem HGT-GAME, tendo em vista uma configuração mais otimizada desse especialista “tapa-buraco”. Tal suposição carece de ser constatada na prática, mas é suportada também pelos resultados obtidos por Weigend *et al.* [472], segundo os quais dois ou três especialistas desse gênero tiveram que ser convocados pela *gating* para a mesma subtarefa.

mento inverso (i. e., de identificação) em blocos de pontos contíguos separados no tempo, tanto para o modelo ME linear (original) como para as suas variantes engatilhada e localizada, todos treinados somente via *Backpropagation*.

⁽⁷⁾Nos experimentos de Weigend *et al.* [472], foram utilizados 10.000 pontos de treinamento e 2.500 pontos de teste, estes divididos em dois conjuntos de avaliação. O número de *lags* adotado pelos autores foi 10. A investigação conduzida aqui considerou uma parcela de dados menor da mesma série, além de um número menor de atrasos, ambos em conformidade com o trabalho de Zhang & Joung [508].

Tab. 8.10: Novos resultados para HGT-GAME, incluindo aqueles para treinamento híbrido baldwiniano. Alguns parâmetros de controle foram alterados (comparar com a Tab. 8.8): $\tau_2, \tau_1(0) = 500$ iterações *Backprop*, $t_{EM}^p = 3$ (número de passos EM para cromossomos-pais), $t_{EM}^f = 10$ (número de passos EM para cromossomos-filhos) e *MAX_GEN* = 100 gerações. Aqui, *L* indica “lamarckiano” e *B* “baldwiniano”. Os rótulos dos critérios seguem aqueles estabelecidos na Tab. 2.2. *DP* significa “Desvio Padrão”, medido com relação à métrica EQM.

	SEQ	EQM	EAM	EPAM	RQ_EQM	RQN_EQM	T_r	T_μ	DP
Forex									
Treinamento									
Gated-HGT-GAME(5) (L)	0,18830	0,00031	0,01307	1,41884	0,00072	0,79367	0,79494	0,69610	4,13%
Gated-HGT-GAME(5) (B)	0,27085	0,00045	0,01387	1,52755	0,00087	0,67046	0,83105	0,72772	7,93%
Teste									
Gated-HGT-GAME(5) (L)	0,04111	0,00026	0,01250	1,35078	0,00126	0,95020	0,90217	0,70283	6,56%
Gated-HGT-GAME(5) (B)	0,04330	0,00026	0,01274	1,37768	0,00125	0,90729	0,91351	0,71167	10,77%
Laser									
Treinamento									
Gated-HGT-GAME(5) (L)	0,97096	0,00097	0,01602	13,82342	0,00099	0,08443	0,08708	0,08430	7,44%
Gated-HGT-GAME(5) (B)	2,53601	0,00254	0,01916	20,50788	0,00160	0,13661	0,14074	0,13624	12,88%
Teste									
Gated-HGT-GAME(5) (L)	1,43045	0,00143	0,01809	14,44745	0,00120	0,09872	0,10146	0,09871	6,89%
Gated-HGT-GAME(5) (B)	2,41858	0,00242	0,02112	12,78808	0,00156	0,12911	0,13193	0,12835	13,55%

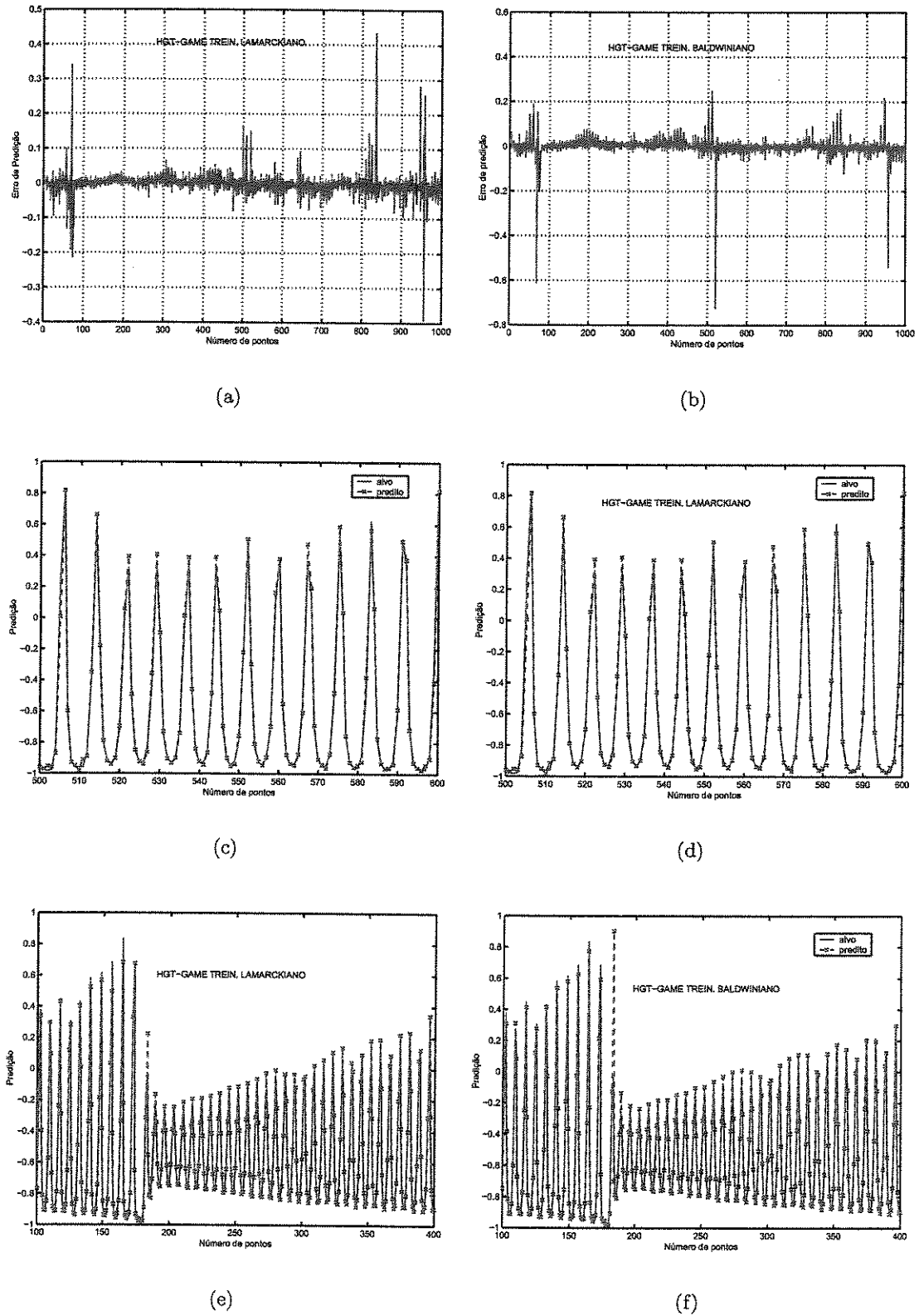


Fig. 8.18: Comparações, sobre o conjunto de teste da série Laser, entre instâncias HGT-GAME treinadas com aprendizado híbrido lamarckiano e baldwiniano. Configuração dos parâmetros dada na Tab. 8.10. (a)-(b) Erro de previsão. (c)-(d) Primeira janela de previsão. (e)-(f) Segunda janela de previsão.

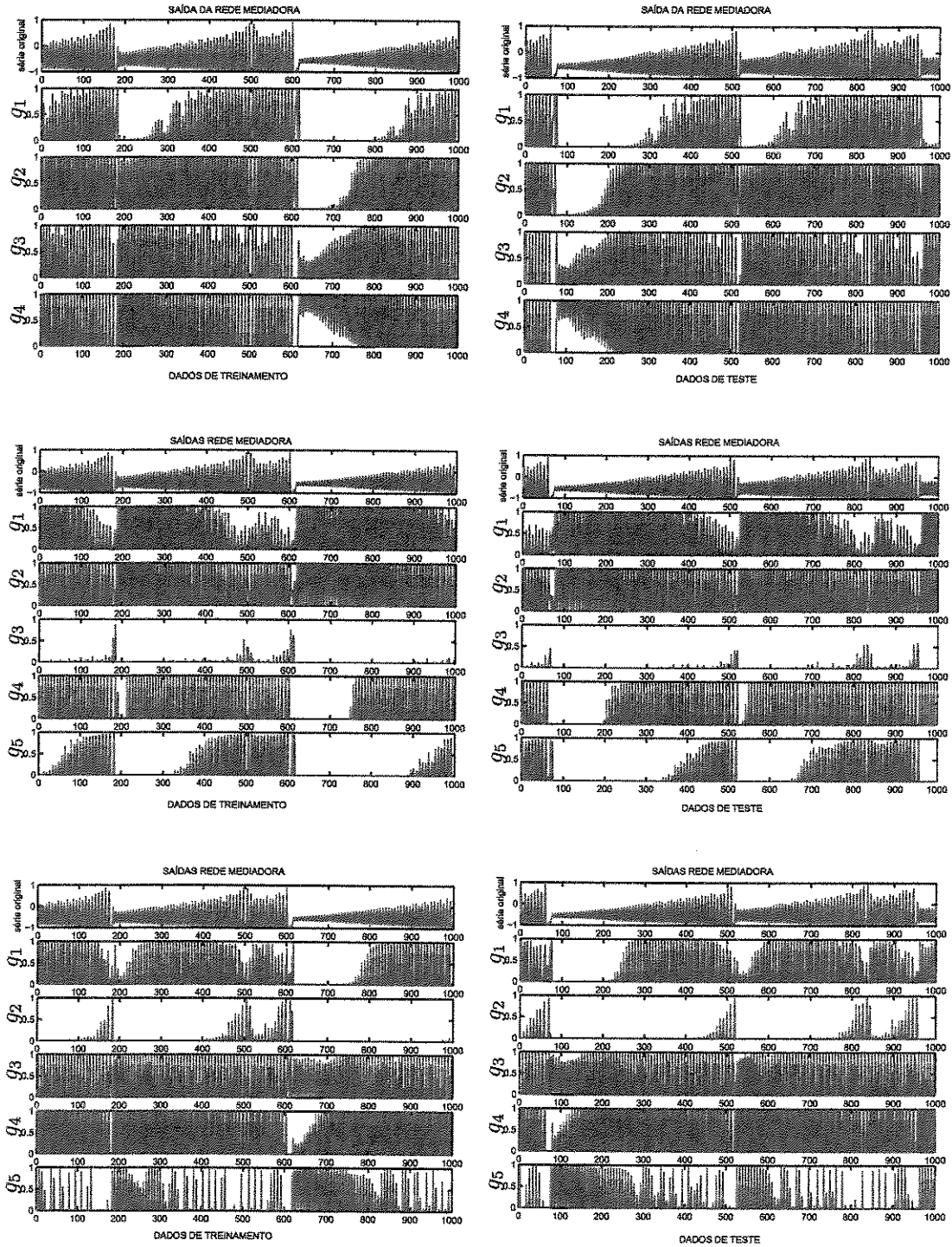


Fig. 8.19: Comparações, sobre os conjuntos de treinamento e teste da série Laser, entre as saídas das redes *gating* das melhores instâncias HGT-GAME treinadas com aprendizado híbrido lamarckiano e baldwiniano. (a)-(b) Lamarckiano com $\tau_2, \tau_1(0) = 25$ iterações *Backpropagation* (configuração antiga). (c)-(d) Baldwiniano com $\tau_2, \tau_1(0) = 500$ iterações *Backprop.* (e)-(f) Lamarckiano com $\tau_2, \tau_1(0) = 500$ iterações *Backpropagation* (configuração nova).

8.4 Organização emergente de times de agentes artificiais

O último artigo, denominado *Emergence of multiagent spatial coordination strategies through artificial coevolution* e publicado num fascículo especial do jornal *Computers & Graphics* dedicado ao tema de Vida Artificial e síntese de comportamento [92], descreve um procedimento alternativo destinado à definição automática de estratégias de coordenação espacial para times de futebol de robô simulado. A cada jogador (agente) é propiciado um conjunto comum de habilidades e é outorgada uma área delimitada do campo onde pode atuar. A idéia por trás dessa abordagem é a de otimizar o comportamento do time como um todo via um processo de coadaptação evolucionário (coevolução cooperativa) mediante o qual novas espécies de jogadores passam a emergir progressivamente no ecossistema (i. e., campo de futebol) desempenhando papéis (*roles*) complementares aos daquelas já existentes. Resultados de simulações experimentais são analisados de modo a avaliar, qualitativamente, os méritos da abordagem proposta.

I. Introduction

One of the most compelling and challenging tasks inside the Distributed Artificial Intelligence (DAI) field is that of suitably devising coordination protocols customized to the problems in mind. Coordination can be summarized as the property of a system of agents performing some activities in a shared environment, concerning with how to effectively orchestrate the group (inter-)actions, in time and space, for achieving coherence [232, 475]. It usually incurs complexity, as there are no predefined general recipes indicating how to establish, *a priori*, the rules of group behavior in view of all possible situations/scenarios. Moreover, there is a range of aspects, such as the homogeneity/heterogeneity of the agents' skills or the environmental characteristics (static versus dynamic), that should also be regarded when one chooses the coordination mechanisms to be employed.

Soccer seems to be a rich testbed domain for the study of multiagent coordination issues. In such context, a set of players must work together in order to put the ball into the opposing goal (augmenting its score) while at the same time defending its own. This is a typical domain where cooperation [125] should take part in—the individuals have the same global objective. One important issue for a soccer team win a game is the strategy it uses, during the game period, to place each of its components in a given region of the field (such as backfield, leftwing, attack etc). That is, how to delimit the zone at which a certain player can perform better in order to improve the capabilities of the whole group. This sort of coordination effort is referred to here as *spatial strategy*. Our primary aim is to evolve this process of team formation through a coevolutionary approach [216, 358, 375], so that a spatial strategy can emerge without human interference. In this regard, the idea is to qualitatively analyze how much the arrangement of a team may influence its overall performance. Furthermore, a secondary purpose underlying this initiative is to reveal the potentials of applying artificial life (coevolutionary-based) techniques towards complex behavior modeling in societies of artificial agents. The problem we are trying to tackle (emergence of team positioning strategies) is straightly

related to the task of automatic synthesis of multiagent behavior since the organization policies adopted by a team directly constrain the possible dynamic comportment it might assume while facing (non-predicted) future situations presented by the environment.

In the sequence, we introduce the robot soccer problem and the artificial coevolutionary approach applied to multiagent spatial coordination, present our framework and solution, show the results from our experiments, and finally identify future plan of work.

II. Background and related work

Many approaches to tackle coordination problems are currently available in the literature [125, 232, 272]. Most of them center around the specification and implementation of high-level protocols (many times based on human social interactions) containing the actions to be taken at particular cases, either by a single agent or by the whole group. Well-known examples following such idea are Laird and others' knowledge-based coordination model and Jennings's formalism of *commitments* and *conventions*. In the latter, for instance, rules to undertake a specific course of action are conceived before the actual deployment of the team in the environment (joint commitments). In order to monitor whether these rules have been fulfilled or whether they are still valid in changing circumstances, there are also additional emergency instructions towards the dynamic adjustment of the group activities through time (social conventions).

Such kind of endeavor aiming at the conception of an *explicit* scheme of coordination seems to be only suited for a constrained class of problems, showing both performance and scalability bottlenecks when applied at more complex domains. Alternative mechanisms have been conceived in order to surpass those deficiencies, such as distributed planning and real-time (re-)planning [444, 475].

A new-fashioned line of research (followed in this work) involves the pervasive use of evolutionary techniques as a means to improve both individual as well as group abilities in a concurrent manner. This methodology stipulates for group organization in a seamless and *implicit* manner; that is, there is no need for explicit pre-codified protocols. The coordination activities can now be viewed as an optimization problem whose solution(s) is (are) searched via a computational procedure that mimics the steps of the natural evolutionary process. Applying such strategy for the automatic configuration of robot soccer spatial coordination strategies constitutes the primary contribution of this work.

The CMUnited [442, 443], developed at Carnegie Mellon University, has been one of the most successful physical robot soccer teams in the contests of the RoboCup world championship [2]. It encompasses a layered learning technique to first train the players' basic skills (dribbling, shooting) for then building more complex capabilities (passing, positioning) upon the basic ones. The formation of the team can change in the course of the game, but the set of possible formations is determined empirically and one of them is chosen in accordance with the current situation of the match [444].

By other means, Balch [34] has used his robot soccer simulator [3] to investigate behavioral specialization in learning robot teams. In his work, all agents have a common set of skills from which they build their tasks achieving coordination through a Q-learning (reinforcement learning) algorithm. After playing for some time against a fixed strategy control team, the learning agents specialize into complementary roles because their reward depends on the score of the game, not on individual actions.

Some papers already report on work concerning the application of artificial learning and evolution to some soccer-related problems. For instance, the approach proposed by Agah *et al.* deals with the production of evolutionary cooperative strategies by means of a devised cognitive architecture based on Tropism [8], whereas Matsubara and colleagues employed a neural network approach towards players' on-line learning on how to take correct decisions (pass a ball to a peer or shoot towards the opposite goal) according to some pre-established field positioning situations [309]. Andou has already assessed the employment of reinforcement learning schemes to update players' positions on the field based on where the ball has previously been located [18]. By other means, Luke *et al.* set out to create a completely learned team of agents using genetic programming [300]. Their approach already employed an artificial coevolutionary methodology which was conceived primarily towards behavior-based team coordination, not coping with spatial organization problems.

The approach underlying this work differs from others in several aspects. First, we do not have any predefined formation for the players, but want that the formation emerges by means of an evolutionary scheme. We do not use reinforcement learning, but also apply the result of the game as a reward function for the employed evolutionary technique (in order to calculate the level of group adaptability), so that the performance of a single player depends on the performance of the whole team. Finally, as a more adequate strategy for the soccer players progressive spatial co-adaptation, a novel memory-based, cooperative coevolutionary architecture [375, 380] towards the dynamic popup (emergence) of instances of evolutionary algorithms has been designed.

III. Simulated robot soccer

Research on robot soccer has received an increasing attention through the last years. Soccer is an attractive domain for multiagent study as the success of a team depends very much on some form of coordination [309, 405]. It is also very appealing because the game is played in a dynamic, real-time, competitive and cooperative environment, from which the agents (players) perceive only a small part (limited visual perspective), what typically incurs the need of world modeling, distributed learning and planning. The control of the agents is decentralized and the changes in the environment neither are fully predictable nor happen in discrete time steps. For our purposes, the game is simplified and is ruled according to the following aspects:

- Teams are composed of five players.

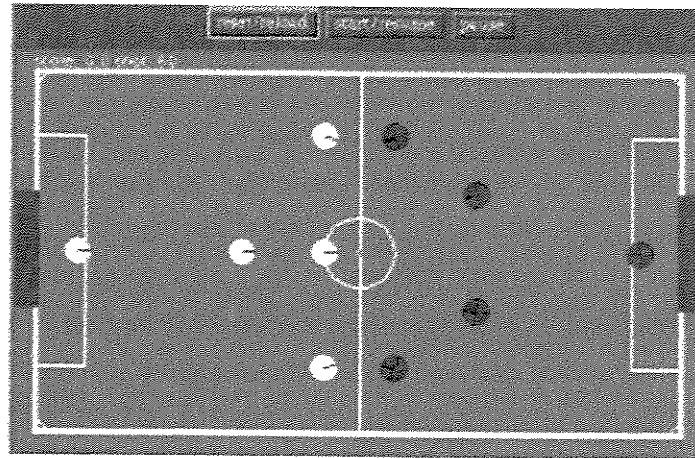


Fig. 8.20: Game start position on the Java-based robot soccer simulator.

- The sidelines are walls—the ball bounces back instead of going out-of-bounds.
- After a scoring event, the ball is immediately placed back in the center of the field.
- Each player has accurate information about the position of the other peers and adversaries as well as of the ball.

According to Huhns and Stephens [475], there are two commonly used methods for apportioning tasks among cooperative agents. One is the *functional* distribution, in which cooperation comes as the union of the individual capacities of the players (one player is a good shooter, other is a passer, and so on). The second method, called *spatial* distribution, is a form of cooperation where the agents divide the search or performing space (in soccer, this is the field) into well-defined areas, in such a way as to quicken the team performance through the sharing of goal responsibilities. In this work, the latter method was chosen for experimenting with the coevolutionary coordination of robot soccer agents.

The Java-based soccer simulator employed in our experiments [3] (Fig. 8.20) implements each player on a separate OS *thread* and runs the simulation in discrete steps. At each step, the robots process their sensor data before ascertaining their appropriate effector commands.

IV. Coordination via coevolution

Evolutionary Computation (EC) has come out as the branch of computational intelligence research employing metaphors from natural evolutionary phenomena as a means to achieve efficient problem solving (search) techniques [31]. Its applicability has constantly increased in recent years, and many are the engineering fields that have some of their processes improved through the application of evolutionary-based approaches. The majority of the implementations of such approaches descend from four independent lines

of research, namely *genetic algorithms* (GA), *genetic programming* (GP), *evolutionary programming* (EP), and *evolution strategies* (ES).

In the conventional GA model, a population of strings (*chromosomes*) codifying the possible solutions for the problem in hand passes through a cyclic (generation-based) process in which new candidates are constantly created and evaluated in accordance with some measure of environment adequacy known as *fitness*. Ancestors are charged by computational operators very much resembling natural evolutionary phenomena, such as reproduction, selection and mutation, being progressively replaced by more adapted newcomers. The population fitness tends to converge in the course of the process and (sub-) optimal solutions are obtained at final stages.

Some problems with this model have already been reported. First, it is very prone to the “local minima/maxima problem”, as it depends very much on the configuration (search space distribution) of the initial population. Likewise, some fast convergence problems may occur if the population size is not properly set. This model has, as well, scalability problems, like how to incorporate all the knowledge about the problem and to discriminate and prioritize (possibly several) distinct factors in a unique evaluation function. In the same manner, the representation of some heterogeneity issues behind the problem may be constrained, as the phenotypic interpretation of parameters is the same for all the individuals (single species). Moreover, the model is also not adequate for the evolution of sets of interacting rules with variable sizes whose individual fitness are determined by their interactions via a simulated micro-economy. (*Classifier systems* and other related works, such as SAMUEL [376], have been devised to surpass such drawback.) Finally, it is not very suited for the representation/generation of complex structures such as those composed by many sub-entities (as it is the case of multi-agent coordination systems).

In order to tackle such deficiencies, distributed genetic algorithms [216] have been introduced. The idea is to bring about a set of genetic algorithm instances working together in a parallel/distributed environment in order to find out the best solution for a common problem. Each GA runs independently from the others. Other, more recently investigated, concepts are those of *niches* and *speciation* [303]. The first brings the idea of dynamically mounting small groups of correlated individuals that act upon a close region of a large search space. Individual niches compete for the allocation of trials. The second refers to new forms of “on-the-fly” species generation.

Extending the boundaries, there is now such a trend to apply *artificial coevolution* [358] as a more suitable technique towards complexity overcoming. Artificial coevolution has its roots in its biological counterpart. Simply put, coevolution means “any reciprocal evolutionary change in interacting species [449].” Although vague, such definition is powerful enough to comprehend any natural process in which two or more species, typically coexisting in a same environment, have their evolutionary trajectories somehow affected by the stable ecological *interactions* and *interrelationships* their members jointly promote and take part in. In the artificial realm, two or more populations of different species are optimized together, one influencing the other by some means.

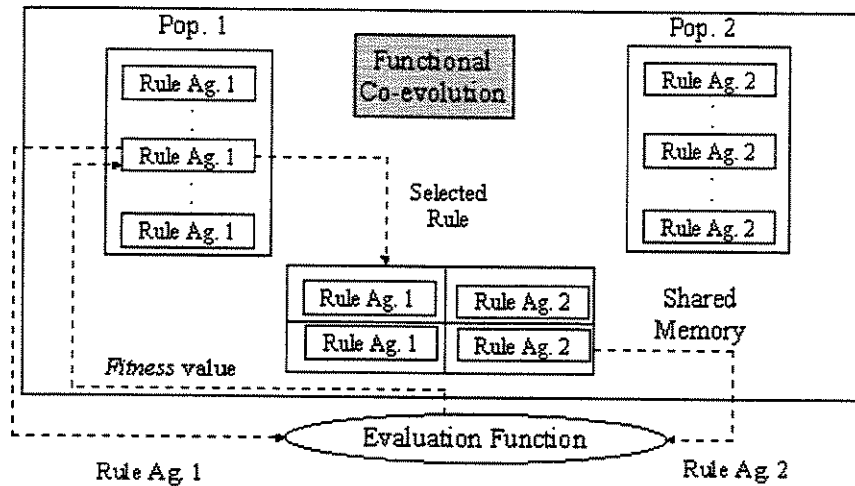


Fig. 8.21: Shared-memory based approach for multiagent coordination.

Artificial coevolution seems very suited for simulating cooperation and/or competition behavior among multiagent entities. Following such premise, Puppala *et al.* have devised a shared-memory based approach [380] to evolve cooperating individuals of two different species—painters and whitewashers—for solving a room painting problem (see Fig. 8.21). In this case, each of the agents has unique skills necessary to complete a job; that is, they are interdependent and the group behavior depends on the joint behavior of both components. The idea is to find pairs whose members are best adapted to each other, so the overall performance can be improved. This should be regarded as a kind of functional decomposability of a huge, high-level problem. In their scheme, an individual from the first population (codified by a rule of behavior) is assessed by mating it with other individuals of the second population (the reverse is also true). Its highest performance evaluation on all pairs that it participates is assigned as its fitness. Instead of randomly picking the individuals, the authors conceived a buffer for grabbing and remembering the most successful pairs achieved so far: In this case, the mating is done by selecting the N best partners from the other population which prevailed at the last generation. The memory is updated if a fitness value of a new assembled pair is higher than any of those currently stored, promoting the replacement of the stored pair with the minimum value (tail of the list) by the new one.

V. Soccer team spatial coordination

In this section, the features underlying our proposal for soccer team spatial coordination are gradually presented.

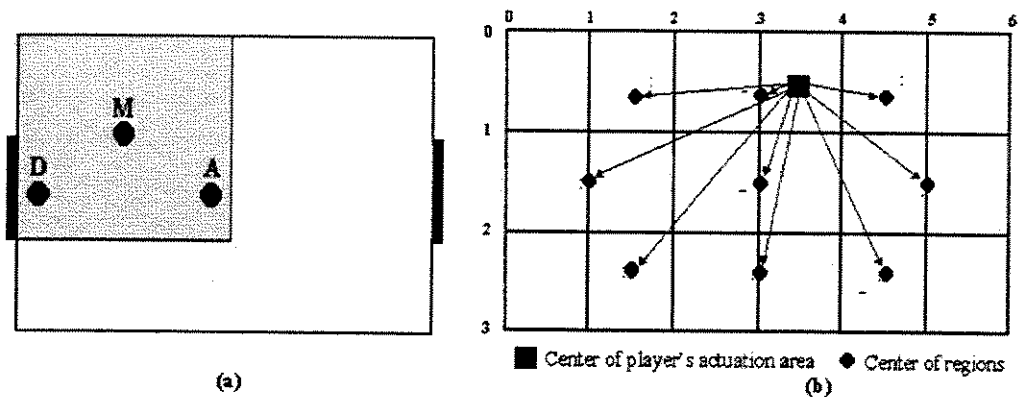


Fig. 8.22: (a) Actuation area of a player with its defense (D), middle (M) and attack (A) position. (b) Discrete field coordinates (6×3 rectangles). Each player is associated to the closest region, from nine defined on the field, according to the distance measured between the center of the actuation area to the center of the region.

Tab. 8.11: Coordinates of the region points used to classify the players according to their acting area.

Region	Coordinates	Region	Coordinates
r_1	(1.5, 0.75)	r_6	(5.0, 1.5)
r_2	(3.0, 0.75)	r_7	(1.5, 2.25)
r_3	(4.5, 0.75)	r_8	(3.0, 2.25)
r_4	(1.0, 1.5)	r_9	(4.5, 2.25)
r_4	(3.0, 1.5)		

V.1 Players and regions

Since our main interest was the formation of the team and its influence on the result of the game, all players, from both opposing teams, were modeled with the same basic skills and control algorithm (see Appendix A). Each player was allowed to perform only inside a particular actuation area, which was characterized by three mark points: *defense* (D), *middle* (M), and *attack* (A) (Fig. 8.22a). In order to avoid a player choosing a too small area to play, the field was separated into 18 squares, as shown in Fig. 8.22b. For classification purposes, we considered nine delimited regions covering the whole field (Fig. 8.22b). Each player of an experimental team was bestowed with a label indicating the region to which it belongs to, in accordance with the minimum Euclidean distance between the center of its actuation area and the center of all nine regions. Table 8.11 shows the coordinates of these regions.

V.2 Teams

The investigation was conducted by engaging experimental teams against a fixed opponent *control* team that uses a 1:3:1 formation (Fig. 8.26a). The purpose was to evolve (or create) new teams that were able to defeat the control team in soccer contests, owing only to a different spatial distribution of the players. The motivation is to certify whether the task of choosing one from a range of different formation strategies has direct influence on the relative performance of an evolved team versus the control team.

V.3 Architecture

Based on the concepts of niches, speciation and cooperative coevolution, we have designed a new architecture for multiagent spatial coordination. The most innovative idea is that of *progressively* assembling the evolving teams (ecosystems) by allocating for each of the five possible players (positions) a promising acting region to be represented by a dynamically created species. In the beginning, all players are randomly selected from the same GA instance (named GA_0)—we could employ any other evolutionary algorithm as well. Then, in the course of generations, some players, competing against all, will prevail, spawning new offspring very akin to them. Those most adapted players and their offspring certainly will perform over similar field regions, characterizing a promising searching area for another GA instance, giving birth to a speciation process. (This is why we partitioned the field into nine logical regions.) As times passes by, new GAs are popped up and GA_0 is restarted with another initial population if the number of players of new species (may be more than one at the same time) fires up a certain threshold. Each new spawned GA is assigned to a place (player) in all future teams formation. That is, one of its members will be selected to take place in each experimental team thereafter. The coadaptation process is granted as the new GAs (species) are formed by in accordance with the already existing ones. At a final step, the spawned GAs have their populations evolved synchronously during some other few cycles for fine-tuning purposes. Fig. 8.23 shows the details of such coevolutionary framework.

Some other considerations are worth to be mentioned:

- This approach also guides the spring of new individuals in the GA_0 population that have complementary roles from those of the already selected species, promoting for an automatic means of problem decomposition.
- The new created GA instances will not allow its individuals to evolve until each of the five field positions has an associated species. This avoids the possibility of badly influencing the formation of the new GAs with corrupted (biased) initial populations of GA_0 . If a GA_0 instance does not produce any novel species for a delimited number of generations it must be replaced by another instance in such a way as to accelerate the search process.
- In order to give to its individuals a better chance to survive and to be selected for a new team, each

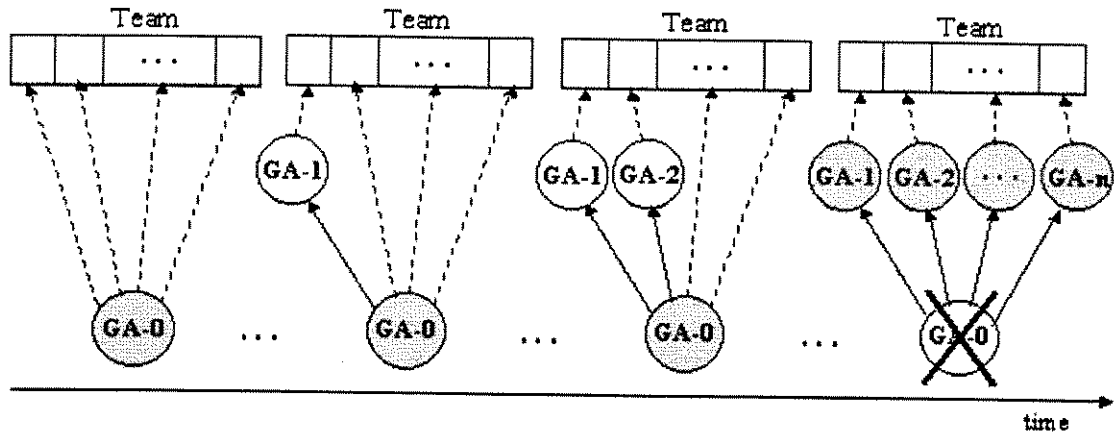


Fig. 8.23: Creation of the teams: in the beginning, all players are taken from GA_0 . As soon as a region of the field has enough players to form a new population, this population is copied into a new GA_1 . After that, each team will have one player coming from that GA. GA_0 is restarted and the procedure is repeated until we have one GA for each position of the team. Then GA_0 is destroyed and the others are activated to coevolve.

new created GA instance can not have more than the half of the number of individuals of GA_0 . Only the most adapted are picked.

- The GA_0 population decreases as the number of species increases. This is because there will be less slots in a team the GA_0 individuals will struggle for.
- The architecture is also memory-based. However, in order to avoid combinatorial explosion problems (as we should have populations in the range of hundreds of elements), we did not adopt Puppala's individual fitness evaluation based on cross-mating. Instead, we opted to apply random teams assemblage (limiting the number of possible teams) for those positions that do not already have an associated species.

For implementation purposes, six classes codified in Java mainly compose this architecture; they are *Player*, *Team*, *Team_pool*, *Memory*, *Simulator*, and *Genetic_Algorithm*, whose interrelationship model is presented in Fig. 8.24. Fig. 8.25 brings a high-level execution flowchart for our coevolutionary approach.

V.4 Genetic algorithm

The creation and evolution of the players are controlled by a genetic algorithm that uses elitist selection, one-point crossover and mutation [31] to generate new players from a previous population. The initial population is randomly generated, in a uniform distribution. The fitness function used to reward the players is not based on their single performance, but on the score of the team where the player actuates.

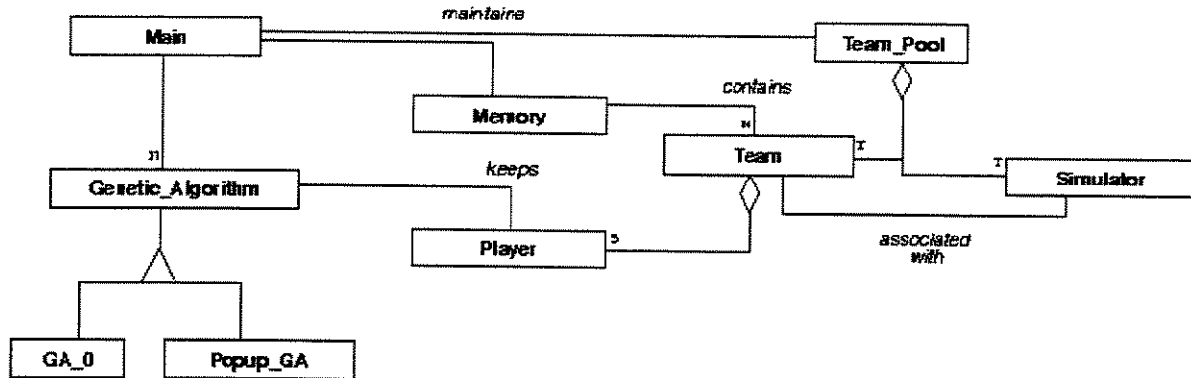


Fig. 8.24: Object model showing the relationships among the main Java classes. N , T , and M are parameters indicating the maximum number of GA instances (5), the number of experimental teams per generation (25) and the number of buffered teams (10).

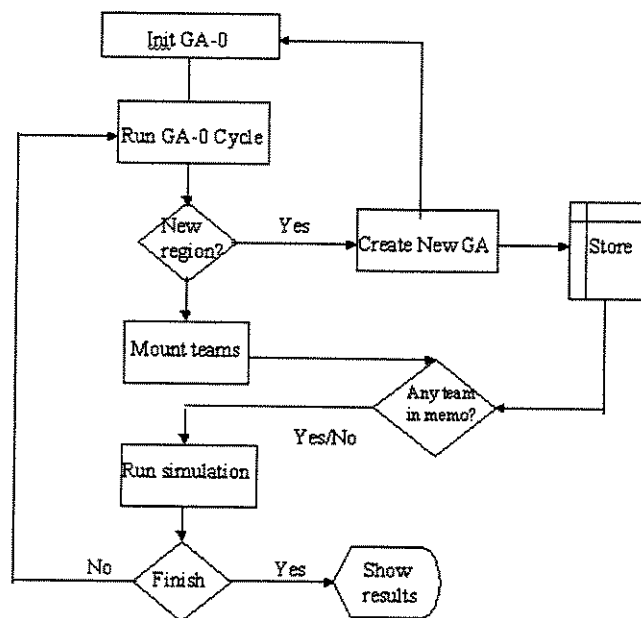


Fig. 8.25: Execution flow in a typical run of the proposed coevolutionary approach.

Eq. (8.12) brings such evaluation function. “ S_{team} ” and “ $S_{control}$ ” are the scores of an evolved team and from the control group, respectively:

$$f(player) = \begin{cases} 3 + \frac{(S_{team} - S_{control})}{10}, & \text{for } S_{team} > S_{control}, \\ -1 + \frac{(S_{team} - S_{control})}{10}, & \text{for } S_{team} < S_{control}, \\ 1 + \frac{S_{team}}{10}, & \text{for } S_{team} = S_{control} \end{cases} \quad (8.12)$$

It may happen that a player is chosen to play in more than one team. In this case it will keep the highest fitness of all teams it participated. In the case that a player does not play any match, it will receive the reward as being the average reward of all players on its region.

VI. Simulation results

Experiments were conducted by running the algorithm described in Appendix B and employing Balch’s Java-based soccer simulator for 50 players and 25 teams. The most important parameter settings can be found in Table 8.12. For each GA_0 generation, all created teams played an 8min long match against the control team. Each simulation of a soccer match was performed on a separate Java *thread* and those simulations were the most time-consuming tasks, taking about 1 hour⁽⁸⁾ to simulate the 25 matches of an evolutionary cycle. The evaluation of the results was focused on three criteria: spring of new formations, cooperative coevolution of the agents and convergence of the genetic algorithm.

VI.1 Formations

Since the only difference between the emerged teams and the control team is the formation, we could verify that it plays a prominent role in robotic soccer. The 10-best evolved teams achieved for the best experimental running were able to defeat the control team with an average of four goals of difference. It is worth to remind that our approach created formations automatically, without human soccer expertise, and could be used to train different team formations according to different adversaries. The formations of the control team and of three winning evolved teams can be seen in Fig. 8.26.

VI.2 Coevolution

Looking at Figs. 8.26b-8.26d, we can notice that the players’ actuation areas that emerged were complementary. Since each of the players came from a different GA instance, and since those GAs were

⁽⁸⁾The experiments were performed using an Enterprise 450 Machine with two 300-Mhz Ultra Sparc-2 processors, 512MB RAM, running SunOS 5.7, and using JDK 1.2 for code implementation.

Tab. 8.12: System configuration parameters.

Parameter	Meaning	Value
POP_LENGTH	Number of players in the initial GA_0	50
MEM_SIZE	Number of best teams that are kept in memory	10
TEAMS	Number of teams to be formed	25
PLAYERS	Number of players per team	5
MAX_NEW_GA_POP_LENGTH	Maximum number of players that a new GA may have	$\frac{TEAMS}{2}$
TIME_TO_LIVE	Maximum number of generations	120
THRESHOLD	Percentage of players a region must have to form a new GA	40%
MIN_TIME_TO_POPUP	Minimum number of generations for a new GA to be formed	5
MAX_TIME_TO_POPUP	Maximum number of generations for a new GA to be formed	20
GENE_CROSSOVER_CHANCE	Chance of crossover occurrence	25%
GENE_MUTATION_CHANCE	Chance of mutation occurrence	1%
ELITIST	Number of best players copied to next generation (elitism)	20%

created in different time steps, we can conclude that the constructive coadaptation approach was pivotal for the arrangement of teams whose players actuate cooperatively for the field covering. To assist the reader in this assessment, Table 8.13 brings the order in which the new GA instances were progressively created for the best experimental running performed so far, displaying the respective regions to which their initial populations (emerged niches of the GA_0) belong.

VI.3 Convergence of GAs

Using the 50-players/25-teams configuration, we observed, through some experiments, that GA_0 was very susceptible to quick convergence. Almost always, the new GA instances tended to emerge within the minimum necessary number of generations (Table 8.12), and their initial populations were formed by only one or two different classes of players (breeds or subspecies). This was typically a non-diversity problem. To avoid this misbehavior, we increased the number of players and teams to 500/250 and observed that the population was indeed more diverse and, thus, did not converge so fast. However, such decision incurred, as a negative side effect, unaffordable simulation cost increases in such a manner as to hamper the performance assessment process. This “diversity X computational cost” tradeoff is typical in any evolutionary-based technique, being generally dealt with via empirical calibrations of the configuration parameters.

VII. Conclusion and future work

Applying coevolutionary techniques in complex problem domains has been proved to be a promising alternative strategy for achieving both performance and quality improvements. Recent research works have addressed the employment of such approach in a variety of problems, including single/multicriteria function

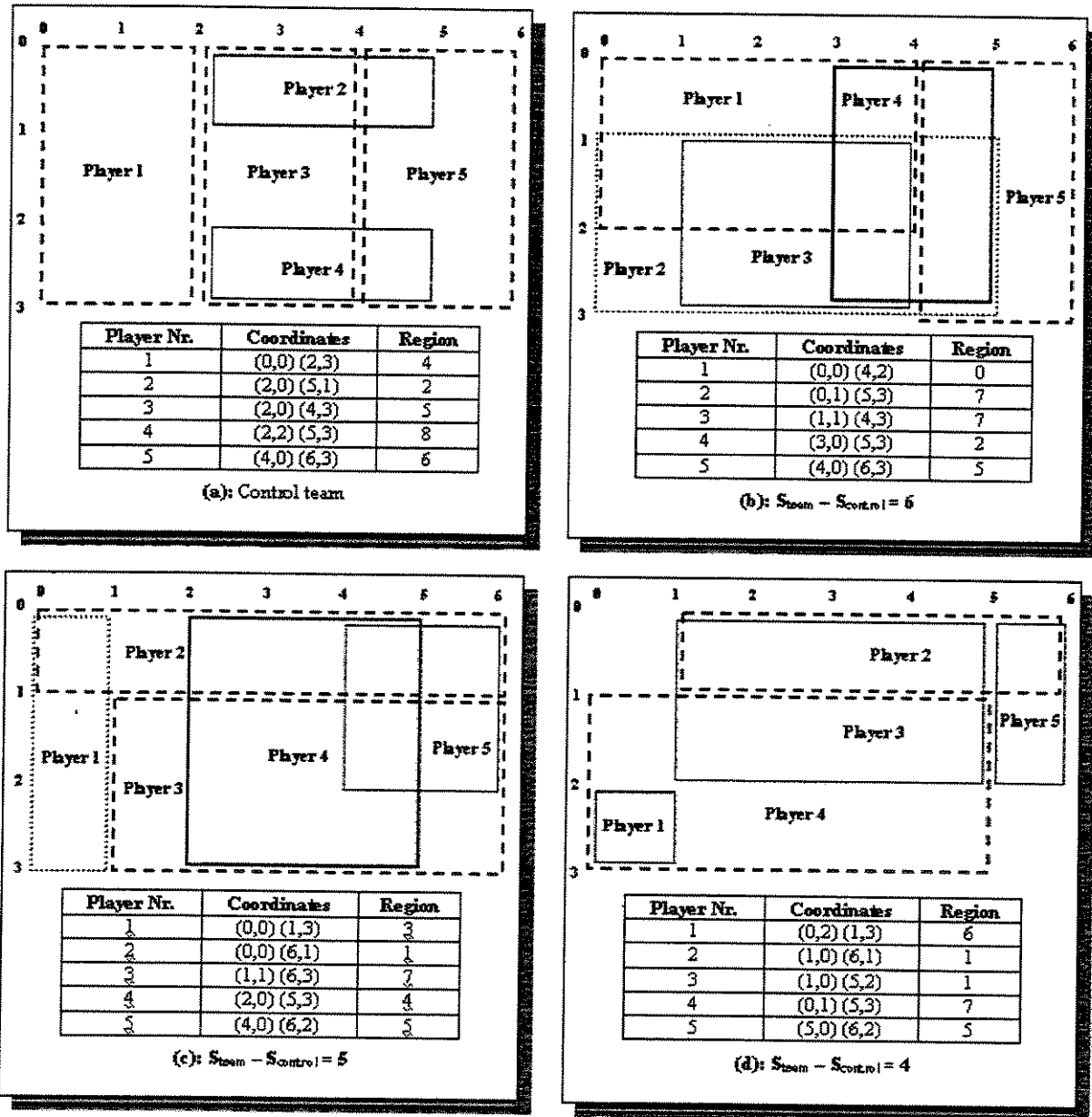


Fig. 8.26: Team formations. (a) Formation of the control team. (b), (c) and (d) are evolved formations that were able to defeat the control team with a goal difference of 6, 5 and 4, respectively. We verify that the players occupy complementary and overlapping parts of the field, in a way that almost all the field is covered by the team.

Tab. 8.13: Order in which the new GAs were created and the regions to which their initial populations belonged.

	GA_1	GA_2	GA_3	GA_4	GA_5
r_1					
r_2	•		•		
r_3					
r_4					
r_5					•
r_6					
r_7				•	
r_8		•			
r_6					

optimization [216, 217, 375] and multiagent scenarios [300, 376, 380]. In this work, a new coevolutionary-based architecture for robot soccer teams spatial coordination was depicted and evaluated, confirming: (i) the feasibility of obtaining an automatic method for the generation of implicit coordination rules; (ii) that the spatial distribution of homogeneous players across the field may direct influence the behavior and performance of the whole team; (iii) that the approach encourages the formation of stable niches of cooperative sub-components (players) whose acting regions tend to be complementary on the field covering; and (iv) that artificial life techniques (particularly coevolutionary-based ones) are a step towards the automatic synthesis of complex behavior and control rules in societies of various autonomous entities.

Some problems were detected during the simulations execution of our approach, demanding for new design or parameter setting corrections. For instance, the fast convergence in the fitness of the new created GAs' populations surely had a great bad effect on some attained results. Increasing the size of all GAs, however, would complicate the players evaluation process and augment the computational time required at each running cycle. As a consequence, there is an intrinsic hard trade-off for configuring parameters of this sort, as well as for those relating to the GA operators (higher mutation rates shall also provide a means to overcome such fast convergence).

Another problem that deserves attention relates to the possibility of losing the constituents of the best existing teams (those already in the memory). Although applying an elitist selection process, it is not assured that these players will be maintained in their respective GAs' populations. (What is maintained is just a copy of the whole team in the pool.) Therefore, we observed that some possibly successful teams that emerged during the generation-based process had their evolutionary "customization" hindered by the extinction of one or more of its components in a posterior generation.

The formations that emerged during our tests are suitable only to play against the control team used in the experiments, performing inefficiently against teams with different formations. This is a big limitation if the team is intended to participate in a competition like the RoboCup. Therefore, an interesting extension to this work would be to train many different formations against distinct configurations of control teams,

store the best formations in a run-time memory and then, during the contests, dynamically adapt the team spatial distribution in conformance with the current opponent's strategy.

Appendix A. Algorithm that controls the actions of each player

```
Compute: (A)attack, (M)middle, (D)defense
do
  if( Player outside its area )
    Move to area;
  else
    if( Ball inside area )
      if( Closest player to ball )
        Move to ball;
      else
        Move close to the ball;
      end
    else // Ball outside the area
      if( Ball on defense side of field )
        Move to (D);
      else
        if( Ball on attack field )
          Move to (A);
        else
          Move to (M);
        end
      end
    end
  end
end
if( Can kick and Is worth kicking )
  Kick the ball;
end
while( Not end of game )
```

Appendix B. Algorithm for the creation of teams through coevolution

```

1. Create GA_0's initial population;
2. Classify players in regions;
3. Create N teams;
4. for( G_1 generations )
    Elitist selection;
    Crossover;
    Mutation;

    Classify players in regions;
    Create N teams;
    Take one player from each GA_i;
    Complete team with players from GA_0;

    //Simulate the games
    for( each team )
        Play against static team;
        //Compute fitness;
        if( S_team > S_control )
            fitness <= 3 + ( S_team - S_control )/10;
        else
            if ( S_team < S_control )
                fitness <= -1 + ( S_team - S_control )/10;
            else
                fitness <= 1 + S_team/10;
            end
            //where S_team and S_control are the score of each team
            //at the end of the game
        end
        Set fitness of the players;
    end
    Save M best teams;
    for( each region )
        Compute fitness of the regions (F_r);
    end
    if( number(GA_i's) < number( players per team ) and

```



```
        G_1 > MIN_TIME_TO_POPUP )
        if(size(region_j) > THRESHOLD )
            Create new GA_i;
            Initialize it with population of region_j;
        else
            if(G_1 >= MAX_TIME_TO_POPUP )
                New GA_i <== Biggest region;
                GA_0 <== New initial population;
            end
        end
    end
end
5. for( G_2 generations )
    //Coevolve especies
    for( each GA_i)
        Elitist selection;
        Crossover;
        Mutation;
    end
    //Simulate the games;
    for( each team )
        Play against static team;
        //Compute fitness;
        if( S_team > S_control )
            fitness <== 3 + ( S_team - S_control )/10;
        else
            if ( S_team < S_control )
                fitness <== -1 + ( S_team - S_control )/10;
            else
                fitness <== 1 + S_team/10;
            end
        end
    end
    //where S_team and S_control are the score of each team
    //at the end of the game
    Set fitness of the players;
end
Save M best teams;
end
6. Print best teams
```

8.4.1 Comentários adicionais

Há de se registrar, primeiramente, que os resultados positivos alcançados com o método coevolutivo apresentado neste projeto devem ser encarados como de caráter preliminar, fruto de um primeiro “laboratório” destinado à observação da possibilidade de se desenvolver automática e emergencialmente estruturas organizacionais artificiais. Não se pode, portanto, concluir que tal abordagem obterá êxito similar quando aplicada a outros ambientes (domínios) de estudo exibindo outras peculiaridades (propriedades) — como, por exemplo, aqueles embasados na Internet [179]. Um estudo interessante, complementar a este realizado, seria o de avaliar a real *escalabilidade* de abordagens coevolutivas cooperativas quando do projeto não-manual de padrões organizacionais de maior complexidade, tais como aqueles ilustrados na Fig. 5.2. Para tanto, seria necessária, pois, a disponibilidade de ambientes simulados (p. ex., mundos de realidade virtual) executando sobre redes dedicadas de computadores, de modo a que seja permissível a formação dinâmica de padrões de interação entre centenas, milhares, ou mesmo milhões de entes artificiais autônomos.

Por outra via, a fim de se aliviar o problema do alto custo computacional decorrente das várias simulações de partidas de futebol, estas destinadas a discriminarem o potencial dos times (e, por conseguinte, dos jogadores), parece ser pertinente lançar-se mão da estratégia pragmática de se trabalhar com valores aproximados da função de *fitness* (vide Seção 6.4), ficando uma instância de RNA responsável por essa tarefa. Sendo o caso, tal modelo neural passaria a reproduzir o mapeamento inerente à função de *fitness* original, tendo como dados amostrais os resultados das simulações realizadas no decorrer das primeiras gerações (ou mesmo em outras execuções) — cada amostra de treinamento passaria a ser composta, pois, pelo cromossomo inteiro de um indivíduo (time ou jogador) e pelo seu valor de *fitness* correspondente. Como a medida de adaptação não necessita ser muito precisa para se avaliar os times (p. ex., os coeficientes da Eq. 8.12 foram atribuídos arbitrariamente), poder-se-ia tolerar uma certa margem de erro para o mapeamento gerado pela RNA.

8.5 Trabalhos correlatos

Primeiramente, passa-se a comentar sobre outros trabalhos realizados na linha de Comitês de Máquinas. Lima *et al.* [284, 287, 289, 291], trazem informações mais detalhadas sobre a metodologia HE-SVM contemplada na Seção 8.2, proporcionando (i) uma melhor formalização do processo de treinamento, na forma de novos problemas de programação quadrática (tanto para classificação como para regressão); (ii) uma análise prospectiva (e pictórica) acerca das vantagens de se combinar funções-*kernels* com espaços de características dissimilares; e (iii) um conjunto rico de experimentos, onde se avalia, sistematicamente, o desempenho da abordagem em relação a aspectos específicos, tais como tolerância a ruídos, escalabilidade (crescimento do número de SVs em função do tamanho do conjunto de amostras de treinamento), e definição dos tipos de *kernels* mais usados para cada tipo de problema (tanto para classificação como para regressão).

Seguindo a estratégia alternativa de se “hibridizar” as abordagens ME e SVM num só *framework*

conceitual, Lima *et al.* propuseram a metodologia *Localized Mixture of Support Vector Machine Experts* (L-MSVME) [288, 292]. Sucintamente, pode-se dizer que, na concepção dessa abordagem, partiu-se do modelo-padrão de MEs (ver Seção 4.2) e demonstrou-se que, por intermédio da introdução de um termo de regularização especial (para controlar a complexidade do modelo) na função-custo de cada especialista e após certas manipulações algébricas, pode-se chegar a um conjunto de equações similares àsquelas encontradas na formalização de SVMs. O resultado teórico relevante por trás desse desenvolvimento é que, na implementação de L-MSVMEs, passa-se a treinar cada especialista sob a perspectiva de SVMs (i. e., via um PPQ específico), ficando ainda assegurado o desacoplamento entre redes *gating* e especialistas durante o seu treinamento via algoritmo EM. Uma discussão mais detalhada acerca de tal abordagem será propiciada ao longo da tese de doutoramento de seu primeiro autor [115].

Por sua vez, a Ref. [285] traz a caracterização e análise de desempenho, em problemas de regressão, predição de séries temporais e identificação de sistemas não-lineares, de *ensembles* compostas por instâncias distintas de redes *neurofuzzy* ANFIS⁽⁹⁾, cada instância sendo configurada com um tipo de função de pertinência diferente (dentre oito possíveis). Abreviadamente, os passos dessa nova proposta, E-ANFIS, compreendem: (a) o treinamento independente de cada componente *neurofuzzy*; (b) o cálculo das saídas dos componentes sobre um conjunto de seleção à parte, distinto do de treinamento; (c) a eleição das melhores instâncias via critério de seleção de Perrone e Cooper (ver Subseção 4.1.4); (d) o cálculo dos pesos de combinação via método CLO-EQM [196]; e (e) a geração das saídas dos componentes e, por conseguinte, da E-ANFIS resultante sobre o conjunto de teste. Como esperado, os resultados obtidos em experimentos controlados foram bastante promissores, endossando o caráter universal (i. e., aplicável a qualquer tipo de estimador) por trás do conceito de *ensemble*. Um ponto ainda passível de ser explorado nessa abordagem diz respeito à questão de se extrair conhecimento mais elaborado dos modelos E-ANFIS resultantes pela interpretação das regras nebulosas neles embutidas (obtidas pela combinação das regras das redes-componentes).

Em uma outra empreitada, Lima e colaboradores [286] articulam uma análise bem detalhada sobre o desempenho de diferentes variantes de MEs — envolvendo tanto modelos lineares como não-lineares (MEs engatilhados e localizados) — para problemas não-triviais de identificação de sistemas não-lineares. Os resultados obtidos em quatro grupos de experimentos confirmaram, com certa regularidade, a superioridade da abordagem ME (qualquer que seja a variante) frente a modelos simples de MLPs (estes, porém, com número de nós escondidos na mesma ordem de grandeza do número de nós escondidos da mistura como um todo). Além do mais, variantes não-lineares (engatilhadas, principalmente) apresentaram maior êxito que aquelas lineares (modelo canônico).

Ainda considerando sistemas híbridos *fuzzy*-neurais, no artigo intitulado “*On the cooperation of fuzzy neural networks via a coevolutionary approach*”, é esboçada uma primeira tentativa de se caracterizar sistemas multiredes sob a ótica de sistemas multiagentes, delineando-se o conceito de *Redes Fuzzy-Neurais Cooperativas* (RFNCs) [86]: “qualquer arquétipo em que duas ou mais redes *neurofuzzy* passam a trabalhar cooperativamente para o atendimento de objetivos mais intrincados, ficando o desempenho de uma rede

⁽⁹⁾De *Adaptive Network-based Fuzzy Inference Systems* [228, 229].

contingente ao desempenho do grupo como um todo” — similar, portanto, ao conceito de time discutido no Cap. 5. Nesse trabalho, a possível aplicação da técnica de coevolução cooperativa baseada em memória, proposta originalmente por Puppala *et al.* [380], para fins de treinamento automático de RFNCs foi justificada, tendo como pano de fundo um problema fictício de reconhecimento de padrões.

No artigo “*A multiagent-based constructive approach for feedforward neural networks*” [290], uma abordagem híbrida — chamada de MASCoNN —, que também estipula pela combinação das vantagens trazidas por sistemas conexionistas e multiagentes, é apresentada, propiciando uma metodologia construtiva alternativa para a configuração progressiva de redes *feedforward*. Nesse projeto, uma hierarquia composta por dois níveis, de coordenação e de operação, foi concebida como meio de se construir redes *feedforward* cujos neurônios escondidos (modelados como *agentes trabalhadores* reativos com funções de ativação unidimensionais) passam a ser “arrebanhados” e ter suas atividades coordenadas por agentes de decisão mais granuloso (*agentes coordenadores* deliberativos). Dentre as vantagens proporcionadas pela abordagem MASCoNN (vide Seção 2.2), destacam-se: (a) a possível interpretação de uma instância de rede neural como um tipo de SMA, além de ser intuitiva, facilita a identificação e extração do conhecimento embutido nos elementos da arquitetura; e (b) o emprego dos agentes coordenadores estimula o uso parcimonioso de recursos — número e formato dos neurônios trabalhadores —, o que, por sua vez, leva à melhor generalização dos modelos neurais finais.

Seguindo uma outra vertente, no artigo “*Devising adaptive migration policies for cooperative distributed genetic algorithms*” [347], especifica-se um novo *framework* arquitetural para AGs distribuídos (ver Seção 6.5), denominado MAS-DGA, em que cada *deme* passa a ser encapsulado por um agente autônomo independente que coopera com os demais para a resolução de problemas complexos de otimização ou aprendizado. O foco aqui está na caracterização de políticas de migração mais adaptativas, no sentido de que a escolha de quais indivíduos devam ser migrados ou substituídos não é definida *a priori* pelo projetista, mas sim de acordo com a dinâmica de execução dos próprios agentes. Tal estratégia contrasta-se diretamente com a abordagem convencional de se empregar heurísticas estáticas (tais como emigrar o melhor indivíduo) [71], as quais podem incorrer, muitas vezes, em perda de eficiência (p. ex., devido à cópia de material redundante). Diferentes configurações de MAS-DGA (relativas a diferentes políticas de migração adaptativas) foram implementadas e avaliadas mediante uma série de experimentos em quatro instâncias de problemas de *modelagem de dependência*⁽¹⁰⁾, mostrando resultados bem promissores quando comparados com aqueles exibidos por AGs convencionais.

Finalmente, é oportuno fazer rápida menção a dois outros projetos conduzidos por este autor na linha de computação flexível aplicada a problemas de sincronização de interdependências. O objetivo por trás do primeiro projeto [383] foi o de se configurar mecanismos de coordenação mais flexíveis (modelados via redes

⁽¹⁰⁾Tarefa de mineração de dados que envolve a descoberta irrestrita de dependências entre atributos. Pode ser vista como uma generalização do problema de classificação em que não se especifica, *a priori*, qual dos atributos disponíveis atuará como classe. Ou seja, cada atributo pode ocorrer *ou* no antecedente *ou* no conseqüente de uma regra de classificação qualquer. Deste modo, o espaço de busca torna-se bem maior que aquele onde antecedentes e conseqüentes já estejam predefinidos [159].

de Petri *fuzzy* generalizadas [366]), almejando atender a requisitos temporais impostos pelo escalonamento de atividades colaborativas [382]. Por outro lado, nas Refs. [89, 90, 91], propõe-se um novo *framework* de modelagem, sob a alcunha de MuSCof, orientado para a especificação, em mais alto nível, de possíveis relacionamentos espaço-temporais entre objetos multimídia encapsulados em componentes de *software*. Para tanto, foi formalizado um conjunto de primitivas de sincronização *fuzzy* bastante extenso, generalizando a semântica “*crisp*” de diretivas topológicas e temporais (estas últimas baseadas na lógica temporal de Allen [12]) convencionais. Desse modo, torna-se possível, por parte do projetista de aplicações multimídia, (i) capturar, antecipadamente, relações imprecisas/incertas entre entidades multimídia distribuídas; e (ii) especificar a possibilidade de eventos inesperados (tais como aqueles gerados como suporte à interação do usuário), indicando como estes devam ser tratados pelo sistema de sincronização da plataforma multimídia utilizada.

8.6 Encerramento

No curso deste capítulo, discorreu-se sobre os trabalhos publicados pelo autor na área de sistemas inteligentes modulares. Na sequência, far-se-á um retrospecto sobre as contribuições veiculadas neste capítulo e no anterior, tendo como pivô o elenco de pontos/requisitos levantados no capítulo de introdução.

Capítulo 9

Conclusão

*“Gigante pela própria natureza.
És belo, és forte, impávido colosso,
E o teu futuro espelha essa grandeza.”*
Joaquim Osório Duque Estrada

Complementando a exposição técnica feita no decorrer deste documento, passa-se a tecer as últimas impressões acerca das atividades desenvolvidas em torno do tema de estudo. Este capítulo é reservado a esse remate, articulando considerações sucintas em duas direções distintas. Na primeira, encaminha-se uma análise pragmática e retrospectiva em torno das contribuições propiciadas. Na segunda, inicia-se um exercício de ampliação das vertentes atuais do trabalho com a projeção de possíveis extensões e novas frentes de estudo.

9.1 Análise das contribuições

É possível analisar as contribuições associadas a este tratado sob dois prismas distintos. O primeiro deles diz respeito ao lado *técnico* propriamente dito. Sob essa perspectiva, é pertinente avaliar *como e em que proporção* as abordagens concebidas e os resultados obtidos junto aos projetos reportados nos Caps. 7 e 8 conseguiram satisfazer à meta primal do trabalho delineada no capítulo de introdução. Para esse propósito, a Tab. 9.1 encaminha um cruzamento entre os três domínios de investigação e os 12 pontos-referência caracterizados na Seção 1.2, comentando sobre como os últimos foram contemplados pelos primeiros.

Com base nessa apreciação e no desempenho promissor exibido pelas ferramentas desenhadas, estas decorrentes diretamente da sinergia de conceitos de cunho evolutivo, simbiótico e construtivista, a ilação a que se pode chegar é que há justificação empírica para se aplicar abordagens híbridas dessa natureza quando do tratamento de problemas relacionados ao projeto não-manual de sistemas inteligentes modulares de elevada complexidade. Tal constatação vem a corroborar e complementar os resultados alcançados por

outros pesquisadores trabalhando em empreendimentos similares ou com certa correlação [104, 119, 167, 363, 374, 483].

Tab. 9.1: Análise retrospectiva sinalizando como os domínios de investigação da tese contemplaram os pontos levantados no capítulo introdutório. A ordem dos pontos e dos domínios segue aquela estabelecida anteriormente.

Pontos	Análise
1	Os problemas abarcados nos três domínios podem ser considerados de alta complexidade. Enquanto os problemas de classificação abordados por HCGA, COHENN, GASE1 exibem características de não-linearidade, os de previsão de séries temporais abordados por HGT-GAME exibem requisitos de não-linearidade, não-continuidade, multiestacionariedade e caoticidade e o problema de coordenação de sistemas multiagentes exhibe propriedades interessantes de ser não-determinístico e não-estacionário.
2	Os três tipos de problemas abarcados são passíveis de serem decompostos, cada um sob perspectivas diferentes. Enquanto para os problemas de classificação e previsão, a decomposição é guiada diretamente pelos dados, para o problema de coordenação, a decomposição assume um caráter mais funcional. Por outro lado, enquanto a decomposição estipulada para os problemas de previsão ocorreu no domínio temporal, para o problema de coordenação, a decomposição emergiu de forma espacial. Nos três tipos de problemas, foi permitido haver sobreposição entre as subtarefas.
3	Para os três tipos de problemas estudados, existem técnicas alternativas para a sua solução, todas elas exibindo algum tipo de deficiência. Por exemplo, os protocolos/mecanismos de coordenação usualmente adotados têm, geralmente, seu escopo de atuação restrito a um certo domínio de problema, necessitando, pois, de intensa reconfiguração para lidarem com situações antes não previstas. Ademais, os métodos de otimização não-linear comumente empregados para o treinamento de arquiteturas conexionistas (como a metodologia <i>Back-propagation</i>) apresentam problemas com relação à convergência a mínimos locais.
4	Nos três domínios, noções de simbiose, construtivismo e holismo foram empregadas como forma de incrementar as potencialidades exibidas por AEs. Quanto ao conceito de simbiose, o seu potencial passou a ser explorado diretamente ao se adotar a Coevolução Cooperativa como paradigma primário das abordagens HCGA e COHENN. Por outro lado, ao se viabilizar a seleção adaptativa de modelos neurais (ver item 6), bem como a definição progressiva dos perfis dos agentes em times de futebol de robôs, aspectos de crescimento/poda evolucionários (muitas vezes, visando ao uso parcimonioso de recursos) passaram a ser estimulados. A complementariedade de técnicas também foi algo bem explorado, notadamente no que diz respeito às abordagens com aprendizado híbrido (COHENN e HGT-GAME).

Continua na próxima página

Tab. 9.1: Análise retrospectiva dos pontos (cont.)

5	Além de adotar uma arquitetura com dois níveis de resolução (descrições de redes e neurônios, respectivamente), a solução gerada por HCGA (i. e., RNA <i>feedforward</i> heterogênea) pode ser visualizada como de natureza hierárquica, haja vista a sua disposição em camadas, cada uma realizando operações simples sobre os resultados produzidos pela anterior. No caso do modelo coevolutivo concebido para a definição automática de estratégias de coordenação de times de agentes autônomos, pode-se dizer que a própria arquitetura do sistema (e, por conseguinte, a solução gerada por ela) passa a emergir de forma progressiva, conquanto a solução final (i. e., time) pode ser visualizada como pertencente a um só nível de organização: horizontal (parceria) ⁽¹⁾ . No que diz respeito ao segundo domínio de investigação, este possibilitou o desenho automático de soluções ainda mais elaboradas, com vários níveis de organização — p. ex., neurônios → redes-componentes → <i>ensembles</i> → <i>ensembles</i> de <i>ensembles</i> .
6	O aspecto da seleção de modelos foi abordado de maneira sistemática nos primeiro e segundo domínios de investigação, quer seja mediante a definição do número de neurônios intermediários de RNAs heterogêneas, quer seja mediante a definição do número de especialistas ou componentes em arranjos de comitês de máquinas (COHENN, GASE1, HGT-GAME).
7	No domínio de projeto automático de comitês de máquinas, a noção de “redundância <i>versus</i> complementariedade” pôde ser investigada. No caso de <i>ensembles</i> coevolutivas ou de <i>ensembles</i> de SVMs, os modelos de componentes são projetados para resolverem a mesma tarefa, ao passo que, nos arranjos de MEs evolutivas, os módulos especialistas são criados para lidar com partes complementares da mesma tarefa ⁽²⁾ .
8	Enquanto nos outros dois domínios as interações entre os módulos (neurônios, redes, SVMs, <i>ensembles</i> , misturas de especialistas) podem ser consideradas como de natureza estática (pois só ocorrem em conformidade com a estrutura imposta pelo <i>framework</i> em que se encaixam), no domínio relativo à coordenação de agentes, os módulos têm maior “liberdade” para instanciarem, dinamicamente, novas formas de interação antes não-planejadas. Por outro lado, enquanto em HCGA, GASE1 e COHENN aspectos de interação positiva (cooperativa) passam a ser encorajados, em HGT-GAME aspectos de interação negativa (competitiva) podem ser analisados. No caso do domínio de coordenação de agentes, tanto interações positivas (parceria intra-time) como negativas (time contra time) podem servir de alvo de investigação.

Continua na próxima página

⁽¹⁾Obviamente que se o modelo coevolutivo fosse estendido para gerar padrões organizacionais mais rebuscados, como hierarquias, heterarquias... (ver Subseção 5.2.2), seria possível observar o surgimento de hierarquias com vários níveis de resolução de (sub)problemas.

⁽²⁾Vale frisar que, na abordagem de *ensembles* coevolutivas, as redes-componentes também foram instigadas (via decomposição automática do espaço de entradas e definição implícita dos nichos de atuação) a “enxergar” o problema sob perspectivas distintas.

Tab. 9.1: Análise retrospectiva dos pontos (cont.)

9	Embora a questão da coordenação coerente de módulos tenha sido avaliada <i>explicitamente</i> no terceiro domínio, pode-se dizer que ela também foi, em parte, contemplada nos demais, uma vez que as interações entre os módulos são controladas diretamente pelo <i>framework</i> arquitetural intrínseco aos modelos de RNAs <i>feedforward</i> , <i>ensembles</i> e MEs, exibindo certa semelhança com a perspectiva de coordenação adotada em sistemas RDP.
10	Ao passo que a abordagem GASEl lança mão de módulos pré-configurados (modelos de SVMs) para fins de combinação, nas demais abordagens os módulos foram projetados automaticamente. Por sua vez, enquanto no domínio de coordenação de agentes a propriedade de autonomia é considerada como intrínseca à concepção dos módulos, os componentes relativos aos modelos de RNAs e comitês de máquinas apresentam limitações quanto a essa propriedade, uma vez que sua atuação é regulada de acordo com o <i>framework</i> arquitetural em que se inserem.
11	Nos dois primeiros domínios de investigação, a questão de se decidir entre módulos homogêneos ou heterogêneos é investigada sistematicamente, quer seja de forma explícita (p. ex., recrutando modelos de SVMs que já se sabiam ser distintos no que concerne à sua capacidade de generalização) ou implícita (deixando a decisão a cargo do próprio processo evolutivo).
12	Lançando mão do mecanismo de atribuição de crédito inerente ao paradigma coevolutivo cooperativo, as abordagens HCGA e COHENN, bem como aquela projetada para o terceiro domínio, têm possibilidade de tratar, de forma automática, com esse difícil dilema entre recompensar ou punir cada módulo em razão da sua efetiva participação para a qualidade atual da saída produzida pelo sistema como um todo.

Sob um outro prisma, pode-se afirmar que o material aqui exposto apresenta um notório valor *didático*. A despeito de existirem referências já consagradas na literatura abordando aspectos relevantes relativos ao tema de engenharia de sistemas inteligentes modulares [250] e/ou ao tópico de combinação de técnicas de computação inteligente [180, 317], este texto proporciona uma visão holística, atualizada e bem fundamentada sobre as diferentes abordagens que suportam o paradigma da modularidade no âmbito de sistemas inteligentes, dispondo-as sob um mesmo contexto analítico e teórico. Enquanto os Caps. 2 a 5 propiciam uma revisão conceitual e formal acerca de RNAs, SVMs, Comitês de Máquinas e Sistemas Multiagentes, traçando, sempre que possível, interligações com aspectos importantes dos projetos realizados, o Cap. 6 anuncia um panorama abrangente sobre as diferentes facetas de aplicação de métodos de Computação Evolutiva junto a essas abordagens modulares. Por outro lado, as inúmeras referências bibliográficas apontadas ao longo do texto representam um retrato bem considerável do estado-da-arte, podendo servir como ponto de partida para novas incursões ao tema.

9.2 Perspectivas futuras

Dada a natureza eclética deste trabalho, existe um número bem elevado de possíveis extensões passíveis de serem enumeradas aqui. Passa-se, na seqüência, a anunciar somente aquelas que, de algum modo, já se encontram sob investigação. As subseções intituladas “Considerações adicionais” do Cap. 8 e a Seção 7.4 contêm material complementar àquele exposto a seguir.

Uma das extensões diz respeito ao emprego de outros dialetos da CE, principalmente no contexto dos métodos coevolutivos. Assim, pode-se atribuir instâncias de AEs distintos (no que tange às suas codificações e seus operadores genéticos) a espécies distintas. Isso pode incorrer em ganhos de desempenho para os modelos resultantes, além de permitir vislumbrar modos interessantes de combinar as suas vantagens e eliminar eventuais desvantagens. Por exemplo, mediante o uso de técnicas eficientes de PG [37], fica possível se realizar buscas em espaços de elevada dimensionalidade, algo interessante de ser considerado para a evolução dos módulos-combinadores das *ensembles* coevolucionárias. Nesse caso, AGs (ou mesmo PE/EEs) continuariam (passariam) a ser adotados para as outras duas populações.

Como complementação a essa estratégia de construir arquiteturas coevolutivas híbridas com AEs “heterogêneos”, está-se projetando o emprego de parâmetros de controle auto-adaptativos (tais como o tamanho da população e taxas de operadores de variação) para cada uma das populações em evolução, em similaridade ao proposto, recentemente, por Iorio & Li [220], no que concerne às arquiteturas coevolutivas com AEs “homogêneos”. Ademais, com base nos trabalhos de Wiegand *et al.* [484, 486] e de Aickelin & Bull [10, 11], está sendo cogitada a possibilidade de se conceber e/ou avaliar outros métodos de parceria de modo a incrementar os ganhos obtidos com as abordagens HCGA e COHENN.

Três outras frentes de investigação, também em andamento, compreendem (a) a adoção de conceitos da área de sistemas paralelos (distribuídos) [71], (b) a adoção de conceitos da área de otimização multi-objetivo [95] e (c) a adoção de conceitos da área de otimização dinâmica [60], visando ao incremento das potencialidades das abordagens concebidas. (AEs paralelos e AEs customizados para problemas de otimização multiobjetivo/não-estacionária foram introduzidos na Seção 6.5.) No caso das extensões distribuídas, o objetivo principal passa a ser o de melhorar a eficiência do processo de busca dos AGs (via manutenção prolongada dos níveis de diversidade), bem como lidar, de maneira mais pragmática, com a questão relativa aos elevados tempos computacionais geralmente associados às execuções dos protótipos construídos. Como exemplo, está sendo implementada uma nova versão de COHENN em que ambas as populações do nível inferior (i. e., de componentes e de combinadores) passam a ser representadas, em verdade, por conjuntos de subpopulações executando paralelamente, havendo a possibilidade de intercâmbio de cromossomos mediante processos migratórios. De um outro modo, está sendo avaliada a pertinência de se empregar uma arquitetura mestre-escravo a fim de se aliviar, pelo menos em parte, a alta demanda por recursos computacionais envolvida na fase de avaliação dos times de futebol de robôs simulado.

Por sua vez, tomando como ponto de partida as abordagens MOCCGA [248] e MOBNET [167], está sendo alvo de consideração adotar noções de otimização multiobjetivo com o intuito de se lidar efetivamente

com o problema de atribuição de crédito aos módulos nas abordagens coevolutivas aqui reportadas (HCGA e COHENN). Desse modo, passará a ser possível contemplar, de maneira mais natural, diferentes critérios de desempenho de gêneros conflitantes, tais como acuidade *versus* generalização ou acuidade *versus* complexidade. Ademais, seria interessante realizar uma análise comparativa entre alguns dos métodos de AEs multiobjetivos enumerados na Seção 6.5, almejando-se revelar as suas potencialidades/deficiências. Estudos comparativos dessa natureza já foram realizados para AEs convencionais [95], contudo, ao que parece, ainda não foram conduzidos para AEs coevolutivos. De maneira similar, está sendo cogitada também a idéia de se evoluir arquiteturas neurais simples ou modulares para lidar com problemas de aprendizado/otimização em que a função-objetivo varie com o tempo — por exemplo, para sistemas de controle ou para séries temporais com vários regimes. Para tanto, está-se avaliando os “prós” e “contras” de algumas técnicas já existentes concebidas especialmente para esse domínio de estudo [60].

Uma extensão natural para o trabalho apresentado na Seção 8.4 seria a de se investigar a possibilidade de emergência de padrões de interação mais rebuscados entre os agentes. Por exemplo, tentar avaliar o surgimento de “jogadas ensaiadas”, como fizeram Stone e Veloso [444], só que a partir de regras dinâmicas de posicionamento espacial dos agentes no campo — ou seja, os papéis realizados pelos agentes poderiam ser alterados ao longo da execução de uma partida, como forma de surpreender o time adversário (*a la* carrossel holandês). Outra extensão seria a de se descartar o time de controle e se estipular que as partidas fossem realizadas entre dois times compostos por agentes em evolução. Isso tornaria a abordagem como um todo como sendo coevolutiva “cooperativa-competitiva”: no nível dos agentes seria cooperativa e no nível dos times seria competitiva (promovendo uma espécie de corrida armamentista) — ver Seção 6.7.

Finalmente, em se tratando especificamente do tema de Comitês de Máquinas, os seguintes tópicos de investigação já fazem parte da agenda de pesquisa do autor⁽³⁾:

1. Estender a arquitetura COHENN de modo a que se possa projetar automaticamente *ensembles* de redes neurais recorrentes [20, 315], almejando-se, com isso, operar sobre problemas de previsão de séries temporais — nesse caso, está-se avaliando a aplicabilidade de parte dos 44 combinadores fixos empregados nesta tese (voltados para problemas de classificação) e dos operadores CLO-EQM propostos por Hashem [196], bem como de outros algoritmos de combinação lineares/não-lineares além daqueles triviais comumente adotados em previsão (p. ex., média simples, ponderada, mediana, produto) [83, 181, 194];
2. Estender a arquitetura COHENN de modo a incluir outros tipos de indutores além de RNAs *feedforward* (ou recorrentes). Estão sob consideração árvores de decisão [326, 491], redes *neurofuzzy* [148, 229, 244] e redes de Petri *fuzzy* [366];
3. Estender o repertório atual dos 44 operadores fixos de combinação e avaliar a possibilidade de aplicar outros tipos de indutores (como aqueles citados no item anterior) para gerarem os combinadores ajustáveis (i. e., configuráveis por evolução);

⁽³⁾ Alguns deles já estão em fase de modelagem ou de implementação.

4. Verificar, num primeiro momento, a aplicabilidade dos procedimentos dirigidos à extração de conhecimento em instâncias simples de RNAs [50, 173, 447, 454] quando do tratamento de *ensembles* de RNAs (ou outros sistemas multiestimadores), quer sejam estas obtidas via processo (co)evolutivo ou não. Num segundo momento, apontadas as suas vantagens/deficiências, pode-se passar a vislumbrar extensões para tais procedimentos de modo a se lidar, de forma mais efetiva, com aspectos específicos por eles não abarcados — p. ex., como extrair a informação contida nos combinadores ajustáveis ou como arcar com possíveis conflitos entre as regras extraídas de estimadores distintos.
5. Particularmente com relação às misturas de especialistas evolutivas, pretende-se aplicar algoritmos de otimização de segunda ordem [386] no lugar do *Backpropagation* para o ajuste dos parâmetros livres no passo-M. Redes recorrentes e redes *neurofuzzy* estão sendo também avaliadas como possíveis modelos para os especialistas (ou mesmo para a rede *gating*) [141, 177]. Além de MEs engatilhadas, planeja-se evoluir também MEs localizadas, como fizeram Hong *et al.* [208], lançando mão dos recursos providos pela abordagem HGT-GAME. Finalmente, está sendo projetado um modelo generalizado de HGT-GAME destinado à evolução de arranjos hierárquicos de MEs [239].

Apêndice A

Aprendizado de máquina: Terminologia básica, tratamento dos dados e análise de desempenho para modelos supervisionados

“Encontre um bug num programa e conserte-o, e o programa irá executar por hoje. Mostre ao programa como encontrar e consertar um bug por si só, e ele irá executar para sempre.”

Oliver G. Selfridge

A área multidisciplinar conhecida como Aprendizado de Máquina [275, 326] detém-se ao estudo dos aspectos computacionais inerentes ao fenômeno do aprendizado, quer seja este observado em máquinas naturais ou artificiais. Um dos seus objetivos basais é o de propiciar a construção de sistemas capazes de adquirir e integrar, por conta própria, conhecimento acerca do domínio (classe de problemas) em que atuam, de modo a melhorar progressivamente seu desempenho em relação ao cumprimento de seus objetivos. A capacidade de “aprender com a própria experiência”, considerada como chavão da área, tem propiciado bons frutos no que concerne ao desenvolvimento de sistemas computacionais cada vez mais adaptativos, com habilidade de se auto-ajustarem continuamente aos requisitos impostos pelo domínio. Nesse sentido, a AM apresenta-se como uma metodologia alternativa (e geralmente mais eficaz) a abordagens tradicionais mais voltadas para a especificação manual (de “tentativa-e-erro”) de tais sistemas.

Tradicionalmente, é possível se distinguir quatro “paradigmas de aprendizado” influenciando os algoritmos de construção de máquinas “inteligentes”, a saber [243]:

Aprendizado supervisionado, cujo propósito está na indução de modelos de mapeamento E/S a partir dos dados que descrevem o problema em questão, sendo tais modelos usados, posteriormente, para a dedução de novas dependências acerca de novos objetos. Assim, durante a sessão de treinamento de uma certa máquina, estímulos de entrada são introduzidos, gerando sinais-resposta como saída. Cada resposta da máquina é comparada ao valor de saída desejado (valor-alvo) previamente estabelecido.

Se a resposta difere do alvo, é gerado um sinal de erro cuja amplitude serve para indicar a magnitude do ajuste a ser dado aos parâmetros da máquina. Esse processo de minimização do erro simula o papel de um supervisor, sendo geralmente implementado mediante métodos de otimização (busca).

Aprendizado não-supervisionado, segundo o qual, durante o processo de treinamento, a máquina de aprendizado tem alterada a sua topologia em função dos estímulos a ela apresentados. Com isso, mediante mecanismos que promovem auto-organização, interação local e realimentação positiva/negativa, o sistema passa a refletir a natureza de distribuição dos dados em diferentes agrupamentos (*clusters*), sendo que membros de um dado agrupamento apresentam similaridades entre si. Vale ressaltar que o número de grupos não é especificado inicialmente pelo projetista, ou seja, não há saída-alvo disponível como no paradigma anterior.

Aprendizado por reforço, em que, embora exista um módulo de supervisão, o seu papel se resume à geração de sinais binários de mais alto nível, tais como “acertou” ou “falhou”, a fim de designar se a saída produzida pela máquina de aprendizado é ou não similar ao valor-alvo procurado. Assim, o processo de correção dos parâmetros livres da máquina segue estratégias diferentes daquelas empregadas para treinamento supervisionado. Fatores importantes aqui abrangem o tempo levado em cada iteração e o número de iterações para se atingir a solução desejada.

Aprendizado competitivo, sendo este, às vezes, considerado como um subcaso do paradigma supervisionado, geralmente aplicado a modelos neurais. Nesse contexto, cada neurônio da camada de saída passa a ser treinado para responder a um tipo particular de estímulo, caracterizando-se, basicamente, como um processo de *especialização*.

O propósito do restante deste anexo é o de prover um suporte conceitual para as discussões realizadas ao longo dos Caps. 2–4, 7 e 8, apresentando, concisamente, (i) uma terminologia básica comumente empregada em trabalhos na área de aprendizado supervisionado, envolvendo critérios de avaliação de estimadores para problemas de classificação; (ii) estratégias de manipulação (normalização e particionamento) dos dados para a geração de modelos; e (iii) uma metodologia estatística para a comparação de desempenho entre estimadores. Grande parte dos conceitos e definições foi retirada diretamente de Freitas [159], Kohavi & Provost [255], Mitchell [326], Monard & Baranauskas [330] e Witten & Frank [491].

A.1 Terminologia básica

Indutor X Estimador. Informalmente, o objetivo de um *indutor* (ou programa de aprendizado ou algoritmo de indução) consiste em extrair um bom *estimador* (também denominado hipótese, máquina de aprendizado ou preditor) a partir de um conjunto de exemplos rotulados extraídos de uma mesma distribuição (ou função-geratriz) F_g , de forma a que esse estimador possa, posteriormente, prever, com a maior exatidão possível, os rótulos (i. e., saídas discretas, para problemas de classificação, ou contínuas, para problemas

de regressão) associados a outros exemplos gerados por F_g . Formalmente, a tarefa de um indutor é induzir uma função h que aproxime F_g , normalmente desconhecida; diz-se, daí, que h é uma hipótese sobre a função-geratriz F_g , ou seja, $h(\mathbf{x}_i) \approx F_g(\mathbf{x}_i)$.

Amostra X Atributo. Em problemas de classificação ou regressão, cada *amostra* (exemplo, caso ou padrão) consiste de um par de valores de E/S dado por $\langle \mathbf{x}_i, F_g(\mathbf{x}_i) \rangle$, sendo $\mathbf{x}_i, i = 1, \dots, N$, uma tupla de *atributos* (características⁽¹⁾) $x_{ij}, j = 1, \dots, D$, compondo a entrada e $F_g(\mathbf{x}_i)$ o(s) valor(es) do mapeamento desejado. Os atributos podem ser *nominais* (ou discretos, categóricos), quando não existe uma ordem entre os seus possíveis valores, ou *contínuos* (quantitativos), quando há uma ordem linear entre os valores (geralmente, numéricos). Sob uma outra classificação, atributos podem ser considerados *relevantes*, caso apresentem boa “capacidade preditiva” (ou seja, sua presença é imprescindível para o indutor gerar o mapeamento), ou *irrelevantes*, caso haja pelo menos uma descrição *completa* e *consistente* (ver tais conceitos adiante) do mapeamento a ser aprendido que não lança mão de tal atributo.

Linguagem de representação de hipótese. Refere-se à notação usada pelo indutor para representar os conceitos apreendidos pela máquina até o estágio corrente de seu aprendizado. Por exemplo, para geradores de árvores de indução, tal notação indicaria a representação atual da árvore (hierarquia de condições “atributo-valor”). Para RNAs (*Backpropagation*) e SVMs, tal “linguagem” envolveria funções matemáticas (funções de ativação e *kernels*, respectivamente), bem como o conjunto corrente de pesos ou coeficientes de Lagrange (vetores-suporte).

Polarização. Qualquer preferência de uma hipótese sobre outra, que não seja decorrente da simples consistência com os exemplos, é denominada *bias* (polarização) de aprendizado. Devido ao fato de quase sempre existir um número grande de hipóteses consistentes, todos os indutores possuem alguma forma de polarização. Com efeito, consoante Mitchell [326], “aprendizado sem *bias* é impossível”.

Distribuição de classes. Para problemas de classificação, dado um conjunto de exemplos D_N , é possível calcular-se a sua distribuição de classes. Para cada classe $C_j, j = 1, \dots, M$, sua distribuição $distr(C_j)$ é calculada como sendo o número de exemplos em D_N que possuem o rótulo C_j dividido pelo número total de exemplos N , ou seja, $distr(C_j) = \frac{1}{N} \sum_{i=1}^N \|F_g(\mathbf{x}_i) = C_j\|$. Por exemplo, seja um conjunto com 100 amostras no total, sendo 60 para a classe C_1 , 15 para a classe C_2 e 25 para a classe C_3 , então sua distribuição de classes seria $distr(C_1, C_2, C_3) = (0, 60; 0, 15; 0, 25) = (60\%; 15\%; 25\%)$. Nesse exemplo, a classe C_1 é dita ser *majoritária* ou prevalente (ver abaixo), ao tempo que a classe C_2 é a *minoritária*.

Prevalência de classes. Um ponto muito importante em AM supervisionada refere-se a como lidar com o desbalanceamento de classes em um conjunto de amostras. Por exemplo, suponha um conjunto de exemplos com a seguinte distribuição de classes $distr(C_1, C_2, C_3) = (99\%; 0, 25\%; 0, 75\%)$, ou seja, com prevalência elevada da classe C_1 . Um classificador simples que classifique sempre novos exemplos como pertencentes à classe majoritária teria, nesse caso, uma taxa de acerto de 99%. Isso pode ser indesejável

⁽¹⁾Alguns autores costumam adotar o termo “característica” num sentido um pouco diferente, designando a especificação conjunta de um atributo e seu valor. Nesse caso, “Cor é azul” seria uma possível característica de um exemplo.

quando as classes minoritárias possuem informação muito importante, relacionadas, por exemplo, a estados de saúde de pacientes ou a tomadas de decisão de cunho financeiro.

Sobre- e Subajuste. Ao induzir um estimador, é possível que este se torne muito específico (sobreajustado) ao conjunto de treinamento utilizado. Como o conjunto de treinamento é apenas um subconjunto amostral daquele composto por todos os exemplos possíveis da distribuição F_g , em situações como essa, o desempenho do estimador tende a ser pior em outros subconjuntos de dados oriundos da mesma distribuição, acarretando má *generalização* do estimador. Por outro lado, também é possível que poucos exemplos representativos sejam dados ao programa de aprendizado, ou que a máquina de aprendizado tenha seu número de parâmetros livres subdimensionado pelo projetista, ou mesmo que ambos os fatos ocorram concomitantemente. Em situações como essas, é possível induzir hipóteses com índices de desempenho muito aquém do esperado, tanto para o conjunto de treinamento como para conjuntos de teste. Diz-se, daí, que houve *underfitting* (subajuste) aos dados por parte do preditor.

Erro e outras métricas. Em problemas de classificação, uma medida de desempenho comumente usada é a taxa de classificação incorreta (*misclassification*) de um estimador h , $err(h)$. Usualmente, tal taxa pode ser obtida via expressão $err(h) = \frac{1}{N} \sum_{i=1}^N \|F_g(\mathbf{x}_i) \neq h(\mathbf{x}_i)\|$, a qual confronta a classe verdadeira de cada exemplo com o rótulo atribuído pelo classificador induzido. O operador $\|E\|$ retorna 1 se a expressão E for verdadeira, e zero em caso contrário. A taxa de acerto $acc(h)$, por sua vez, é vista como o complemento da anterior, ou seja, $acc(h) = 1 - err(h)$. Além da taxa de erro (acerto), pode-se avaliar o estimador considerando outras métricas, tais como *compreensibilidade* (i. e., facilidade de interpretação do modelo), *grau de relevância* (*interestingness*) ou grau de compactação por trás do conhecimento gerado, velocidade de aprendizado e requisitos de armazenagem.

Taxas de reconhecimento, substituição e rejeição. A *taxa de reconhecimento* é outro nome para a taxa de acerto, ao passo que a soma entre a *taxa de substituição* e a *taxa de rejeição* equivale à taxa de classificação incorreta. A taxa de substituição indica o percentual de amostras em que o estimador *trocou* os rótulos da classe e a taxa de rejeição designa o percentual de amostras em que o mesmo não conseguiu gerar uma decisão não-ambígua (i. e., não conseguiu discriminar entre dois ou mais rótulos igualmente prováveis).

Completude e Consistência. Uma vez induzido, um estimador pode ser avaliado acerca de seu *grau de completude*, ou seja, se ele classifica todos os exemplos, mesmo que erroneamente (taxa de rejeição nula), e de seu *grau de consistência*, isto é, se ele classifica *corretamente* os exemplos não-rejeitados. Portanto, dado um estimador, ele pode ser caracterizado como: (a) completo e consistente; (b) incompleto e consistente; (c) completo e inconsistente; ou (d) incompleto e inconsistente.

Matriz de confusão. A matriz de confusão de uma hipótese h oferece uma medida efetiva do seu modelo de classificação intrínseco, ao mostrar o número de classificações corretas *versus* as classificações preditas para cada classe sobre um conjunto de exemplos D_N . Como mostrado na Tab. A.1, os resultados são dispostos em duas dimensões, a de classes verdadeiras e a de classes preditas, considerando-se M classes diferentes $\{C_1, C_2, \dots, C_M\}$. Cada elemento $M(C_j, C_j^*)$ da matriz, $j, j^* = 1, 2, \dots, M$, que é calculado pela

Tab. A.1: Matriz de confusão de um classificador.

Classe	Predita C_1	Predita C_2	...	Predita C_M
Verdadeira C_1	$M(C_1, C_1)$	$M(C_1, C_2)$	...	$M(C_1, C_M)$
Verdadeira C_2	$M(C_2, C_1)$	$M(C_2, C_2)$	...	$M(C_2, C_M)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Verdadeira C_M	$M(C_M, C_1)$	$M(C_M, C_2)$	...	$M(C_M, C_M)$

expressão

$$M(C_j, C_{j^*}) = \sum_{\{\forall(\mathbf{x}, F_g(\mathbf{x})) \in D_N : F_g(\mathbf{x}) = C_j\}} \|h(\mathbf{x}) = C_{j^*}\|,$$

representa o número de exemplos de D_N que pertencem à classe C_j e que foram classificados pelo estimador como sendo da classe C_{j^*} . O número de acertos, para cada classe, se localiza na diagonal principal $M(C_j, C_j)$ da matriz. Os demais elementos $M(C_j, C_{j^*})$, para $j \neq j^*$, representam erros na classificação. A matriz de confusão de um classificador ideal possui todos esses elementos iguais a zero, uma vez que ele não deve cometer erros.

Falso positivo e Falso negativo. Considerando um problema com apenas duas classes, rotuladas como C_p (positiva) e C_n (negativa), os dois tipos de erros possíveis são denominados *falso positivo* (F_p), relativo ao número de exemplos classificados erroneamente como positivos, e *falso negativo* (F_n), indicativo do número de exemplos classificados erroneamente como negativos. De posse desses parâmetros, a taxa de erro da classe C_p poderia ser escrita como $\frac{F_n}{T_p + F_n}$, ao passo que a taxa de erro da classe C_n seria dada por $\frac{F_p}{F_p + T_n}$, sendo T_p o número de exemplos positivos classificados corretamente e T_n o número de amostras negativas classificadas corretamente de um total de $N = (T_p + F_p + T_n + F_n)$ casos.

Custo do erro. Entende-se como *custo do erro*, ($cost(C_j, C_{j^*})$), um valor de penalidade atribuído ao estimador sempre que ele comete um erro de rotulação, no caso, substituindo a classe verdadeira C_j por uma outra classe C_{j^*} , com $j, j^* \in \{1, \dots, M\}$, sendo M o número de classes. Assim, $cost(C_j, C_j) = 0$, uma vez que não representa erro de substituição e $cost(C_j, C_{j^*}) > 0$, para os demais casos. Em geral, os indutores assumem $cost(C_j, C_{j^*}) = cost(C_{j^*}, C_j) = 1$. Pode-se alterar a taxa de classificação incorreta (taxa de erro) de modo a refletir os pesos impostos pelos custos $cost(C_j, C_{j^*})$; isso daria luz à taxa de erro ponderada $err\_cost(h) = \frac{1}{N} \sum_{i=1}^N \|F_g(\mathbf{x}_i) \neq h(\mathbf{x}_i)\| \times cost(F_g(\mathbf{x}_i), h(\mathbf{x}_i))$.

Precisão, Recuperação, Medida-E e Medida-F. Outras métricas, comumente usadas na área de Recuperação de Informação (*Information Retrieval*), podem ser obtidas a partir das definições acima⁽²⁾: (i) *precisão* (*precision*), dada pela razão entre o número de membros da classe (positiva ou negativa) classificados corretamente e o número total de instâncias classificadas como sendo da classe (positiva ou negativa), ou

⁽²⁾Deve-se atentar para o fato de que o cálculo dessas métricas pode ser facilmente estendido para quando $M > 2$ (múltiplas classes). Nesse caso, basta considerar, a cada iteração $j, j = 1, \dots, M$, a classe C_j como sendo a classe positiva e a junção de todas as outras classes $C_{j^*}, j^* \neq j$, como sendo a classe negativa.

seja, $prec_c(h) = \frac{T_c}{T_c + F_c}$ com $c = \{p, n\}$; (ii) *recuperação (recall)*, dada pela razão entre o número de membros da classe (positiva ou negativa) classificados corretamente e o número total de instâncias da classe, isto é, $recall_c(h) = \frac{T_c}{T_c + F_{c^*}}$ com $c = \{p, n\}$ e $c^* \neq c$; e (iii) *medida-F (F-measure)*, dada pela combinação das duas anteriores, ou seja,

$$Fmeas_c(h) = \frac{2 \times recall_c(h) \times prec_c(h)}{recall_c(h) + prec_c(h)} = \frac{2T_c}{2T_c + F_c + F_{c^*}},$$

com $c = \{p, n\}$ e $c^* \neq c$. A *medida-E*, por sua vez, pode ser obtida fazendo $Emeas_c(h) = 1 - Fmeas_c$.

A.2 Tratamento dos dados

Quatro formas de normalização de dados são geralmente discriminadas na literatura [509]:

Normalização intra-canal: Um *canal* pode ser definido como o conjunto de todos os valores relativos ao mesmo atributo para todas as amostras, de treinamento ou teste. Ou seja, cada variável de entrada independente é um canal. Portanto, esse tipo de normalização, também conhecido como *não-isotrópico*, é realizado coluna por coluna no caso de se dispor todos os vetores de entrada sob uma só matriz. De uma outra forma: a normalização é feita individualmente para cada canal.

Normalização cruzada: Esse tipo de normalização é realizado de forma independente para cada vetor de entrada, cruzando todos os canais para uma única amostra.

Normalização mista: Usa algum tipo de combinação entre a normalização intra-canal e a normalização cruzada.

Normalização externa: Também conhecida como *isotrópica*, faz com que os dados relativos às amostras sejam *todos* normalizados em um intervalo específico de uma só vez.

Dados um conjunto de exemplos de tamanho finito D_N e um indutor, é importante *estimar* qual seria o desempenho futuro do classificador induzido sobre outras amostras pertencentes à mesma distribuição F_g que gerou D_N . Isso é possível ao se lançar mão do recurso da *amostragem* dos dados do conjunto D_N para fins de treinamento e posterior avaliação do modelo. As abordagens discutidas na sequência seguem essa estratégia.

Método holdout. O método mais usual de particionamento de dados envolve a separação aleatória das amostras de D_N em dois subconjuntos fixos e mutuamente exclusivos, chamados de *conjunto de treinamento*, usado para a geração da hipótese, e *conjunto de teste*, empregado para medir o grau de efetividade do conceito aprendido pela hipótese. Os subconjuntos devem ser disjuntos, assim como as amostras de teste devem ser representativas da distribuição da função-geratriz dos dados, para se certificar se o estimador apresenta boa capacidade de generalização aos dados ou não, tornando a medida de sua acuidade estatisticamente válida.

Método *holdout* modificado. Como variação da estratégia anterior, é comum dividir-se aleatoriamente o conjunto de treinamento em duas partes, uma para o treinamento propriamente dito do indutor, e outra para a *validação* do modelo no curso de seu processo de aprendizado. Esse conjunto de validação serve como guia para se analisar o grau de generalização do preditor, antes de este ser avaliado na fase de teste. A abordagem *early stopping* [386], por exemplo, faz uso desse conjunto à parte almejando terminar o processo de treinamento de uma RNA sempre que o nível de generalização do modelo começar a cair; ou seja, a partir do ponto em que a taxa de erro medida sobre os dados de validação começar a subir. Um dos problemas com esse método de particionamento é que nem sempre o conjunto de amostras é grande o suficiente para permitir a separação dos dados em conjuntos representativos; por exemplo, ao se retirar dados de treinamento para formar o conjunto de validação, pode-se estar perdendo informação valiosa a ser apreendida pelo indutor. O inverso também é verdadeiro.

Método de validação cruzada (*cross-validation*). Apresenta-se como uma estratégia mais eficaz para a avaliação da generalização do estimador quando cotejada com o método *holdout*, uma vez que permite que todos os dados disponíveis em D_N sejam usados tanto para treinamento como teste. Na validação cruzada com Q partições (*folds*), divide-se o conjunto de dados, num total de N amostras, em Q partições de tamanhos aproximadamente iguais, ou seja, cada um com $\approx N/Q$ amostras. Tipicamente, o valor de Q é 10. Aplica-se o indutor e treina-se Q vezes a máquina, deixando, em cada iteração, uma das partições de fora do treinamento, a ser usada, posteriormente, para efeito de teste, consoante um ou mais critérios de avaliação. Assim, cada amostra é usada uma única vez para teste e $Q - 1$ vezes para treinamento. A taxa de erro média, aqui, é calculada pela média das taxas de erro obtidas com relação as Q partições. Outras medidas de avaliação, como precisão, recuperação e medida-F, podem também ser adotadas. Pode-se apelar para este método de particionamento almejando-se realizar seleção de modelo (*model selection*) para um dado algoritmo de aprendizado. Nesse caso, a configuração dos parâmetros relativa ao modelo escolhido seria aquela concernente à melhor execução dentre as Q realizadas. Por outro lado, quando da comparação entre diferentes algoritmos de indução, o modelo escolhido corresponderá à melhor execução do indutor que apresentar o menor erro médio em suas Q execuções. Vale notar que o método de validação cruzada é bem diferente do método *holdout* modificado discutido acima. Nesse último, somente um único subconjunto (o de validação) é usado para estimar o erro de generalização do estimador antes deste estar “pronto” para ser aplicado (no caso, sobre o conjunto de teste). Para a validação cruzada, Q subconjuntos são usados para esse propósito, produzindo, geralmente, uma medida mais precisa da qualidade do estimador para conjuntos de dados de tamanho moderado.

Método de validação cruzada estratificada. Compreende uma variação da estratégia anterior. Ao se gerar os *folds* mutuamente exclusivos, a distribuição das classes passa a ser também considerada durante a amostragem aleatória. Isso significa, por exemplo, que se o conjunto original de exemplos possui duas classes com distribuição de 20% e 80%, cada partição também terá essa mesma proporção.

Método “deixe-um-de-fora” (*leave-one-out*). Também compreende uma variação do método de validação cruzada, consoante a qual o número de partições Q torna-se igual ao número de amostras do

conjunto D_N , isto é, $Q = N$. Assim, o processo de avaliação é repetido N vezes, sendo o erro final dado pela média dos N erros de teste obtidos. Uma grande desvantagem dessa variante é que ela é bastante dispendiosa em termos computacionais, sendo mais indicada para conjuntos D_N de tamanhos não-elevados.

Método de *bootstrap*. A idéia básica aqui é a de se repetir o processo de estimação diversas vezes, sendo cada experimento conduzido independentemente dos demais. Cada ensaio firma-se em um novo conjunto de treinamento, de tamanho igual ao do original (N), obtido por reamostragem (i. e., amostragem *com reposição*) aleatória do conjunto original de exemplos D_N . Isso significa que algumas amostras podem não aparecer num dado conjunto de treinamento D_N^q , relativo ao q -ésimo experimento. Por outro lado, também torna-se muito provável que uma única amostra apareça repetidas vezes nesse mesmo conjunto. Os exemplos que não integrarem D_N^q serão usados para efeito de teste. Considerando Q experimentos no total, uma dada amostra tem probabilidade $1 - (1 - \frac{1}{N})^Q$ de ser selecionada uma única vez em cada uma das Q reamostragens aleatórias sobre D_N . Para $N \rightarrow \infty$, tal probabilidade deve convergir, aproximadamente, para $1 - 1/e = 0,632$; portanto, para esse método, a *fração média* de amostras não-repetidas é de 63,2% no conjunto de treinamento e de 36,8% no conjunto de teste.

A.3 Comparação entre algoritmos

Métodos de AM se apresentam como ferramentas poderosas para a geração de sistemas adaptativos. Contudo, não existe um único algoritmo que apresente melhor desempenho frente a todos os problemas possíveis [121], o que, de certo modo, vai ao encontro dos resultados teóricos sumarizados pelos teoremas NFL [493]. Para se determinar qual indutor é o mais indicado para resolver um dado problema, seria necessário identificar, primeiramente, quais dos algoritmos disponíveis são mais apropriados para lidar com o *domínio* em que o problema se insere. No entanto, é muito difícil fazer um julgamento prévio nesse sentido, já que o conceito a ser induzido não é conhecido *a priori*. Por esse motivo, passa a ser fundamental realizar testes estatísticos confiáveis de sorte a se confrontar os desempenhos *relativos* apresentados por estimadores gerados por tais algoritmos. Existem muitas metodologias estatísticas para sinalizar quantitativamente se a diferença entre os desempenhos exibidos por dois algoritmos quaisquer é significativa ou não para um certo problema, sendo que uma boa revisão acerca desses métodos foi propiciada recentemente por Dietterich [122] e Salzberg [406]. No que segue, serão expostos os passos do teste estatístico conhecido como *teste-T emparelhado*, quando aplicado junto com o método de validação cruzada [491].

Para a aplicação do teste, é assumida a existência de dois indutores A e B e um único conjunto de amostras D_N , sendo que A e B geram hipóteses via o método de validação cruzada utilizando-se das *mesmas* Q partições. Sejam $\langle a_1, a_2, \dots, a_Q \rangle$ e $\langle b_1, b_2, \dots, b_Q \rangle$ as taxas de erro dos algoritmos A e B , respectivamente, sobre as Q partições, cujas médias são denotadas por $\bar{a}$ e $\bar{b}$. Sejam também $\langle d_1, d_2, \dots, d_Q \rangle$ as diferenças entre as taxas de erro de A e B , isto é, $d_i = a_i - b_i$, com $\bar{d} = (\sum_{i=1}^Q d_i)/Q$ indicando a média dessas diferenças

e $\sigma_d^2 = \frac{1}{Q} \left[\frac{1}{Q-1} \sum_{i=1}^Q (d_i - \bar{d})^2 \right]$ significando a variância⁽³⁾ relativa a essa média. De posse de todas essas variáveis, pode-se calcular a estatística t como sendo:

$$t = \frac{\bar{d}}{\sqrt{\frac{\sigma_d^2}{Q}}}.$$

Verifica-se, em seguida, se a diferença entre os algoritmos é significativa (hipótese alternativa) ou não (hipótese nula). Para tanto, encontra-se, primeiramente, o *grau de significância*, dado por

$$s = 1 - T(t, 2(Q - 1)),$$

com $T(\cdot)$ denotando a distribuição *t-student* cumulativa. Posteriormente, compara-se o valor de s com o *threshold de confiança* α prefixado pelo projetista (tipicamente, $\alpha = 0,95$). Caso $s \geq \alpha$, então a hipótese nula é satisfeita e os algoritmos A e B têm desempenhos equivalentes; caso contrário, as seqüências de médias $\{a_i\}$ e $\{b_i\}$ não são estatisticamente iguais e o algoritmo que apresentar a maior média ($\bar{a}$ ou $\bar{b}$) será aquele com o melhor desempenho.

⁽³⁾Assumindo que $\bar{d}$ pertença a uma distribuição normal e que cada d_i seja uma diferença entre valores médios, a_i e b_i , cabe observar que o termo $1/(Q - 1)$ na expressão acima tem origem na definição de um estimador não-viciado para a variância, enquanto o termo $1/Q$ advém diretamente do Teorema do Limite Central como forma de se estimar a variância das médias [330].

Apêndice B

Métodos de combinação em sistemas com múltiplos classificadores

“A unanimidade é burra.”
Nelson Rodrigues

Este adendo provê uma caracterização de alto nível dos 44 operadores fixos de combinação de estimadores empregados nos experimentos conduzidos no Cap. 7. Os 13 primeiros operadores *não* levam em conta informações relativas à fase de treinamento como forma de ponderar as saídas dos classificadores-componentes (ou seja, são considerados estáticos), ao passo que os restantes sim (são adaptativos). Para auxiliar a discussão dos métodos de combinação calcados na teoria da evidência de Dempster-Shafer [9, 175, 499], fica a Seção B.1 reservada à introdução de conceitos dessa área.

Na descrição formal a seguir, DN_{tr} e DN_{ts} representam, respectivamente, conjuntos de amostras de treinamento e de teste, à medida que N_{tr} indica o número total de amostras de treinamento⁽¹⁾; M denota o número de classes, contabilizando também a de rejeição, sendo esta associada ao último rótulo; $C = \{1, \dots, j, \dots, M\}$ refere-se ao conjunto formado pelos índices das M classes; K indica o número de componentes que formam a *ensemble*, sendo que $\Gamma = \{1, \dots, k, \dots, K\}$ refere-se ao conjunto formado pelos índices desses K estimadores; $\mathbf{v}(\mathbf{x}) = [v_1(\mathbf{x}), \dots, v_k(\mathbf{x}), \dots, v_K(\mathbf{x})]^T$ é o vetor formado pelos votos (ou seja, índices das classes) de saída dos componentes em referência à amostra $\mathbf{x}$; $\mathbf{r}_k(\mathbf{x}) = [r_k^1(\mathbf{x}), \dots, r_k^j(\mathbf{x}), \dots, r_k^M(\mathbf{x})]^T$ é o vetor de *ranking* (decrecente) das M classes relativo ao k -ésimo componente; $\mathbf{a}_k(\mathbf{x}) = [a_k^1(\mathbf{x}), \dots, a_k^j(\mathbf{x}), \dots, a_k^M(\mathbf{x})]^T$ denota o vetor de graus de suporte⁽²⁾ (i. e., $\sum_j a_k^j(\mathbf{x}) = 1$) para o k -ésimo componente em relação as M classes e à amostra $\mathbf{x}$, ao tempo que $\mathbf{DP}(\mathbf{x}) = [\mathbf{a}_1(\mathbf{x}), \dots, \mathbf{a}_k(\mathbf{x}), \dots, \mathbf{a}_K(\mathbf{x})]^T$ simboliza uma matriz $K \times M$ chamada de *perfil de decisão* (*decision profile*) [265] para o padrão $\mathbf{x}$; $\mathbf{e} = [e_1, \dots, e_k, \dots, e_K]^T$ é o vetor de taxas médias percentuais de erro

⁽¹⁾Para os casos onde há aprendizado local dos componentes (ver Cap. 7), o valor de N_{tr} compreende tanto os dados pertencentes ao conjunto de treinamento como aqueles de validação.

⁽²⁾Passíveis de serem interpretados também como probabilidades *a posteriori* (ver Seção 2.2 e Subseção 4.1.5).

exibidas pelos K componentes sobre os N_{tr} dados; e $\Omega(\mathbf{x})$ captura a saída da *ensemble* para a amostra $\mathbf{x}$. No mais, para evitar redundância na escrita, algumas definições de símbolos matemáticos auxiliares feitas para um operador serão usadas para outros operadores subseqüentes.

Eis a lista dos 44 operadores:

1. *Unanimidade* [32]: A *ensemble* decide que um dado padrão $\mathbf{x} \in DN_{ts}$ pertence à classe j (ou seja, $\Omega(\mathbf{x}) = j$) se e somente se todos os classificadores integrantes de Γ decidirem que $\mathbf{x}$ pertence à classe j . Formalmente: $v_k(\mathbf{x}) = j, \forall k \in \Gamma$. Não sendo o caso, rejeita-se $\mathbf{x}$.
2. *Unanimidade modificada* [32]: A *ensemble* decide que uma observação $\mathbf{x} \in DN_{ts}$ pertence à classe j (i. e., $\Omega(\mathbf{x}) = j$) se *alguns* estimadores de Γ indicarem que $\mathbf{x}$ pertence à classe j e os demais rejeitarem essa amostra. Formalmente: $(\exists k \in \Gamma \mid v_k(\mathbf{x}) = j) \wedge (\forall k' \in \Gamma, k' \neq k, v_{k'}(\mathbf{x}) = \{j, M\})$; de outro modo, rejeita-se $\mathbf{x}$.
3. *Voto majoritário* [32, 40, 273]: O grupo decide que um dado padrão $\mathbf{x} \in DN_{ts}$ pertence à classe j (i. e., $\Omega(\mathbf{x}) = j$) se mais da metade ($50\% + 1$) dos componentes indicarem que $\mathbf{x}$ pertence à classe j . De um modo formal, seja $\Gamma_j \subseteq \Gamma$ o subconjunto de classificadores cujo voto de saída é para a classe j , ou seja, $v_k(\mathbf{x}) = j, \forall k \in \Gamma_j$, então a asserção acima fica capturada pela condição $Card(\Gamma_j) \geq \frac{1}{2}Card(\Gamma) + 1$, onde $Card(\Gamma_j)$ indica o número de elementos de Γ_j . Não satisfeita tal condição, rejeita-se $\mathbf{x}$.
4. *Pluralidade* [32, 40]: A *ensemble* decide que uma observação $\mathbf{x} \in DN_{ts}$ pertence à classe j (i. e., $\Omega(\mathbf{x}) = j$) se o número de componentes de Γ que decidirem que $\mathbf{x}$ pertença à classe j seja consideravelmente maior (limiar τ_P predefinido) que o número de classificadores que suportarem qualquer outra classe. Formalmente: $\Gamma_j, \Gamma_j^* \subset \Gamma, Card(\Gamma_j) \geq (1 + \tau_P) \times Card(\Gamma_j^*), \forall j^* \neq j$. De um outro modo, rejeita-se $\mathbf{x}$.
5. *Borda* [205]: Calcula-se, primeiramente, o *escore Borda* para cada classe j ($j = 1, \dots, M$), somando-se os fatores $(M - r_k^j)$ referentes a todos os K estimadores. A decisão da *ensemble* será dada pela classe com maior escore Borda, ou seja, $\Omega(\mathbf{x}) = \arg \max_{j \in C} \sum_{k=1}^K (M - r_k^j(\mathbf{x}))$.
6. *Condorcet* [281]: Para a j -ésima classe, verificam-se quantos são os componentes da *ensemble* que a colocam num posto maior que uma outra classe $j^* \neq j$ qualquer, sendo esse número representado por $\#_{jj^*}(\Gamma : r_k^j > r_k^{j^*})$. O *escore Condorcet* da classe j , $Con_j(\mathbf{x})$, é o mínimo entre os $M - 1$ valores resultantes, ou seja, $Con_j(\mathbf{x}) = \min_{j^* \in C - \{j\}} \#_{jj^*}(\Gamma : r_k^j > r_k^{j^*})$. A decisão da *ensemble* será dada pela classe com maior escore Condorcet, ou seja, $\Omega(\mathbf{x}) = \arg \max_{j \in C} Con_j(\mathbf{x})$.
7. *Média* [260, 281]: Pode ser considerado como um tipo de operador OWA⁽³⁾. A decisão da *ensemble* será a classe j com maior valor-suporte agregado, μ_j , sendo este calculado pela *média* dos graus de suporte a essa classe gerados por todos os classificadores, isto é, $\mu_j(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^K a_k^j(\mathbf{x})$. Disso: $\Omega(\mathbf{x}) = \arg \max_{j \in C} \mu_j(\mathbf{x})$.

⁽³⁾Operadores OWA (de *Ordered Weighted Averaging*) são comumente aplicados para fins de agregação *fuzzy* de informações imprecisas ou vagas advindas de fontes distintas/conflitantes [498].

8. *Produto* [260]: Outro tipo de operador OWA. Similar ao anterior, só que o valor-suporte agregado passa a ser calculado via *produto* dos graus de suporte gerados por todos os estimadores, ou seja, $\mu_j(\mathbf{x}) = \prod_{k=1}^K a_k^j(\mathbf{x})$.
9. *Mediana* [260]: Outro tipo de operador OWA. Similar ao anterior, só que o valor-suporte agregado passa a ser calculado pela *mediana* dos graus de suporte produzidos por todos os classificadores, isto é, $\mu_j(\mathbf{x}) = \text{median}(a_k^j(\mathbf{x})), \forall k \in \Gamma$.
10. *Máximo* [260]: Outro tipo de operador OWA. Similar ao anterior, só que o valor-suporte agregado passa a ser dado pelo *supremo* dentre os graus de suporte produzidos por todos os classificadores, isto é, $\mu_j(\mathbf{x}) = \sup(a_k^j(\mathbf{x})), \forall k \in \Gamma$.
11. *Mínimo* [260]: Outro tipo de operador OWA. Similar ao anterior, só que o valor-suporte agregado passa a ser dado pelo *ínfimo* dentre os graus de suporte produzidos por todos os estimadores, isto é, $\mu_j(\mathbf{x}) = \inf(a_k^j(\mathbf{x})), \forall k \in \Gamma$.
12. *Júri competitivo* [260]: Outro tipo de operador OWA. Se $K < 3$, então aplica-se o operador *Média* descrito acima. Caso contrário, para cada classe j : (i) ordenam-se, decrescentemente, os índices dos K classificadores de acordo com a força de seus graus de suporte em relação a essa classe (a_k^j), o que gera uma nova lista de índices $\Gamma_j^* = \{1_j^*, \dots, k_j^*, \dots, K_j^*\} \subseteq \Gamma$; (ii) retiram-se os estimadores com maior e menor grau de suporte (1_j^* e K_j^* , respectivamente); e (iii) calcula-se o valor-suporte agregado, μ_j , como feito para o operador *média*, sem levar em consideração os graus de suporte dos classificadores retirados, ou seja, $\mu_j(\mathbf{x}) = \frac{1}{K-2} \sum_{k=2_j^*}^{(K-1)_j^*} a_k^j(\mathbf{x})$. A saída da *ensemble* será dada como anteriormente, quer dizer, $\Omega(\mathbf{x}) = \arg \max_{j \in C} \mu_j(\mathbf{x})$.
13. *Média ponderada* [281, 353]: Esta é uma variação do operador *média* que aplica um peso $w_k = \frac{1-e_k}{\sum_{t=1}^K (1-e_t)}$ para ponderar o grau de suporte do k -ésimo estimador quando do cálculo do valor-suporte de cada classe. Assim, o valor de μ_j passaria a ser calculado como sendo $\mu_j(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^K w_k a_k^j(\mathbf{x})$ e $\Omega(\mathbf{x}) = \arg \max_{j \in C} \mu_j(\mathbf{x})$.
14. *OWA com operador de compensação de Zimmermann & Zysno* [466]: Outro tipo de operador OWA, em que o valor-suporte agregado para a j -ésima classe passa a ser dado pelo operador de compensação definido por Zimmermann & Zysno, ou seja, $\mu_j(\mathbf{x}) = \left(\prod_{k=1}^K [a_k^j(\mathbf{x})]^{w_k} \right)^{1-\beta} \left(1 - \prod_{k=1}^K [1 - a_k^j(\mathbf{x})]^{w_k} \right)^\beta$, sendo que os w_k 's são coeficientes indicando a “competência global” dos componentes — estimados, entretanto, sobre o conjunto de treinamento —, consoante a restrição $\sum_k w_k = K$, ao passo que $\beta \in [0, 1]$ é um parâmetro de compensação. Para os experimentos conduzidos, $\beta = 0,5$ e $w_k = K \times \frac{1-e_k}{\sum_{t=1}^K (1-e_t)}$. A saída da *ensemble* será dada como antes, ou seja, $\Omega(\mathbf{x}) = \arg \max_{j \in C} \mu_j(\mathbf{x})$.
15. *Método Bayesiano ingênuo* [497]: Para se gerar a saída da *ensemble* para um padrão $\mathbf{x} \in DN_{ts}$, são levadas em consideração as probabilidades de erro de cada componente para cada classe, extraídas

diretamente da sua matriz de confusão. Neste método, colhem-se, primeiramente, todos os votos dos classificadores, cujas distribuições de erro são consideradas mutuamente independentes — daí, a denominação “ingênuo” (*naïve*) para o método, dada por alguns. Em seguida, para cada classe j , passa-se a calcular a sua *função-crença* associada, levando-se em conta os votos e desempenhos dos componentes. Tal função-crença é dada por $BEL(j) = \eta \prod_{k=1}^K P(\mathbf{x} \in j \mid v_k(\mathbf{x}) = j^*)$, $j, j^* \in C$, onde $P(\mathbf{x} \in j \mid v_k(\mathbf{x}) = j^*)$ denota a probabilidade de a classe j ser a verdadeira mesmo que o voto do k -ésimo classificador seja para a classe j^* — veja que j^* pode ser igual a j —, ao tempo que η descreve um fator de normalização; no caso, $\eta = 1/(\sum_j \prod_k P(\mathbf{x} \in j \mid v_k(\mathbf{x}) = j^*))$. A saída do grupo, por sua vez, passa a ser dada por

$$\Omega(\mathbf{x}) = \begin{cases} j & \text{se } j = \arg \max_{j^* \in C} BEL(j^*) \text{ e } BEL(j) \geq \tau_{Bayes}, \\ M \text{ (rejeição)} & \text{caso contrário,} \end{cases}$$

sendo $0 \leq \tau_{Bayes} \leq 1$ um limiar de decisão predefinido.

16. *Behavior Knowledge Space* (BKS) [212]: Para K componentes e M classes, uma matriz BKS conterà M^K células, cada célula $BKS(v_1(\mathbf{x}) \dots v_K(\mathbf{x}))$ representando uma possível permutação dos votos (índices das classes) desses estimadores. Para esse método, a saída da *ensemble* é dada pela seguinte função de combinação:

$$\Omega(\mathbf{x}) = \begin{cases} R_{v_1(\mathbf{x}) \dots v_K(\mathbf{x})} & \text{se } T_{v_1(\mathbf{x}) \dots v_K(\mathbf{x})} > 0 \text{ e} \\ & \frac{nd_{v_1(\mathbf{x}) \dots v_K(\mathbf{x})}(R_{v_1(\mathbf{x}) \dots v_K(\mathbf{x})})}{T_{v_1(\mathbf{x}) \dots v_K(\mathbf{x})}} \geq \tau_{BKS} \\ M \text{ (rejeição)} & \text{caso contrário,} \end{cases}$$

onde $nd_{v_1(\mathbf{x}) \dots v_K(\mathbf{x})}(j)$ designa o número de amostras de treinamento pertencentes à classe j na célula $BKS(v_1(\mathbf{x}) \dots v_K(\mathbf{x}))$, ao passo que $T_{v_1(\mathbf{x}) \dots v_K(\mathbf{x})} = \sum_{j=1}^M nd_{v_1(\mathbf{x}) \dots v_K(\mathbf{x})}(j)$ e $R_{v_1(\mathbf{x}) \dots v_K(\mathbf{x})}$ indicam, respectivamente, o número total de amostras e a classe representativa da célula $BKS(v_1(\mathbf{x}) \dots v_K(\mathbf{x}))$. Ademais, $0 \leq \tau_{BKS} \leq 1$ simboliza um limiar de confiança predefinido.

17. *Gating-1* [281, 384]: Dentre os K componentes, ir-se-á selecionar apenas um deles para gerar a saída da *ensemble*, assemelhando-se, portanto, à abordagem ME (Seção 4.2). Contudo, em vez de se estabelecer ao longo do treinamento qual(is) região(ões) deva(m) ser alocada(s) a que classificador(es), aqui, a seleção do estimador será feita *após* o treinamento (ou seja, na fase de operação), tendo como base o seu desempenho numa região R_i que engloba a amostra sob análise $\mathbf{x}_i \in DN_{ts}$. Para tanto, para o k -ésimo classificador, $k \in \Gamma$, cria-se uma lista l_k com os índices das amostras de treinamento para as quais ele gerou resposta certa. Em seguida, procura-se, dentre as N_{tr} amostras de treinamento, aquela mais *vizinha* a $\mathbf{x}_i$, digamos $\mathbf{x}^* \in R_i$, segundo uma medida de similaridade preestabelecida (no caso, inverso da distância euclidiana). Para $\mathbf{x}^*$, escolhe-se arbitrariamente um estimador $k^* \in \Gamma$ que a tenha acertado, ou seja, $\mathbf{x}^* \in l_{k^*}$, e a saída de k^* , $v_{k^*}(\mathbf{x}_i)$, será, portanto, aquela da *ensemble* ($\Omega(\mathbf{x})$). Caso $\mathbf{x}^* \notin l_k, \forall k \in \Gamma$, então rejeita-se $\mathbf{x}_i$.

18. *Gating-2* [281, 384]: Similar ao anterior só que são *duas* as amostras mais similares, pertencentes a R_i , a serem usadas para fins de seleção do componente que irá produzir a saída do grupo.
19. *Gating-5* [281, 384]: Similar ao anterior só que são *cinco* as amostras mais similares, pertencentes a R_i , a serem usadas para fins de seleção do componente que irá produzir a saída do grupo.
20. *Gating Modificado-1* [281, 384]: Variação do operador *Gating* proposta por este autor em que, no lugar de se trabalhar com um único componente que tenha acertado a amostra mais similar de $\mathbf{x}_i$, passa-se a levar em consideração *todos* os que a tenham acertado, formando o subconjunto $\Gamma^* \subseteq \Gamma$. Desse modo, a saída final da *ensemble* ($\Omega(\mathbf{x}_i)$) passa a ser dada por meio do *Voto majoritário* sobre os componentes de Γ^* .
21. *Gating Modificado-2* [281, 384]: Similar ao anterior só que são *duas* as amostras mais vizinhas, pertencentes a R_i , a serem usadas para o propósito de seleção dos componentes que irão integrar Γ^* .
22. *Gating Modificado-5* [281, 384]: Similar ao anterior só que são *cinco* as amostras mais vizinhas, pertencentes a R_i , a serem usadas para a seleção dos componentes que irão integrar Γ^* .
23. *Fuzzy integral de Sugeno* [82]: O conceito de integral *fuzzy* [445] também pode ser adotado com o objetivo de se combinar classificadores, podendo ser visto como uma generalização de operadores OWA. Este método de combinação compreende sete passos distintos: (1) Fixam-se K *densidades fuzzy* $g^1, \dots, g^k, \dots, g^K$, cada g^k representando a probabilidade estimada de classificação correta relativa ao k -ésimo componente, ou seja, sua taxa de reconhecimento no treinamento; (2) Calcula-se uma *medida fuzzy* [445] g sobre os K valores g^k de modo a levar-se em conta a *importância de quaisquer subconjuntos* de componentes de Γ com respeito a cada classe $j \in C$. Para tanto, pode-se encontrar uma medida λ -*fuzzy* consistente com essas densidades, sendo o valor de λ obtido como a única raiz real maior que -1 do polinômio $\lambda + 1 = \prod_{k=1}^K (1 + \lambda g^k)$, $\lambda \neq 0$; (3) Para cada nova amostra $\mathbf{x} \in DN_{ts}$, ordenam-se, decrescentemente, em relação a cada classe (coluna) j , os elementos a_k^j (graus de decisão) da matriz $\mathbf{DP}(\mathbf{x})$, o que gera uma nova lista de índices $\Gamma_j^* = \{1_j^*, \dots, k_j^*, \dots, K_j^*\} \subseteq \Gamma$; (4) Para cada classe j , ordenam-se as densidades *fuzzy* de acordo com a nova lista de índices Γ_j^* e atribui-se $g(1) = g^{1_j^*}$; (5) Para cada classe j , calcula-se, recursivamente, $g(t) = g^{t_j^*} + g(t-1) + \lambda g^{t_j^*} g(t-1)$, com t variando de 2 a K ; (6) Calcula-se o valor-suporte μ_j para cada classe j como sendo $\mu_j(\mathbf{x}) = \max \left\{ \min(a_{t^*}^j(\mathbf{x}), g(t)) \right\}_{t=1}^K$; e (7) Finalmente, calcula-se a saída da *ensemble* como sendo $\Omega(\mathbf{x}) = \arg \max_{j \in C} \mu_j(\mathbf{x})$.
24. *Fuzzy integral de Choquet* [82]: Variação do método acima em que, em vez de se usar a integral *fuzzy* de Sugeno, emprega-se a de Choquet. Considerando a mesma medida λ -*fuzzy*, a fórmula referente ao cálculo do valor-suporte μ_j (passo 6) passa a assumir uma nova roupagem, sendo dada por $\mu_j(\mathbf{x}) = a_{1^*}^j(\mathbf{x}) + \sum_{t=2}^K \left(a_{(t-1)^*}^j(\mathbf{x}) - a_{t^*}^j(\mathbf{x}) \right) g(t-1)$. A saída da *ensemble* continua sendo calculada como $\Omega(\mathbf{x}) = \arg \max_{j \in C} \mu_j(\mathbf{x})$.

25. *DCS_LA*⁽⁴⁾-1 [494]: Método semelhante ao operador *Gating*. A escolha do classificador também é feita na fase de operação, só que a indicação de quais componentes são passíveis de serem selecionados é feita *dinamicamente* para cada nova amostra $\mathbf{x}_i \in DN_{ts}$ com base no *grau de certeza* por trás da decisão corrente, sendo esta embasada no grau de precisão dos componentes sobre a região local R_i englobando $\mathbf{x}_i$. Este operador envolve os seguintes passos: (1) Se todos os votos $v_k(\mathbf{x}_i), k \in \Gamma$, são idênticos, então o rótulo da classe indicada será a saída da *ensemble*; (2) Caso contrário, para cada classificador k , calcula-se uma estimativa de sua precisão local $acc_k(\mathbf{x}_i)$. Isso é feito coletando-se o voto de k , digamos $v_k(\mathbf{x}_i) = j$, e depois encontrando as N_v amostras de treinamento mais vizinhas de $\mathbf{x}_i$ para as quais tal estimador produziu o mesmo rótulo j , o que gera o conjunto $DN_k^{N_v} \subset DN_{tr}$. A proporção de amostras de $DN_k^{N_v}$ cujos rótulos verdadeiros sejam realmente j serve como estimativa de $acc_k(\mathbf{x}_i)$; (3) Caso haja um único classificador k^* com maior precisão local ($\Gamma_1^* = \{k^*\}$), então o seu voto sinalizará a saída da *ensemble*, ou seja, $\Omega(\mathbf{x}) = v_{k^*}(\mathbf{x}_i)$. Senão, verifica-se se todos os classificadores-vencedores (i. e., elementos de Γ_1^*) produzem o mesmo rótulo de saída. Sendo o caso, este será a saída da *ensemble*. De outro modo, se um único rótulo puder ser selecionado via operador *Pluralidade* entre as saídas dos elementos de Γ_1^* , então tal rótulo será a saída da *ensemble*; (4) Caso contrário, considera-se que há um empate entre rótulos produzidos por Γ_1^* . Passa-se, daí, a considerar também o(s) voto(s) daquele(s) estimador(es) que ficaram em segundo lugar no torneio de precisões locais, de modo a que se chegue em um desempate. Isso passa a gerar um outro conjunto Γ_2^* , incluindo tanto os classificadores campeões como vice-campeões. Se houver um único rótulo vencedor em Γ_2^* , ou seja, $v_{k^*}(\mathbf{x}_i) = j^*, \forall k^* \in \Gamma_2^*$, então esse rótulo será a saída da *ensemble* ($\Omega(\mathbf{x}) = j^*$); (5) Caso continue havendo empate, continua-se iterativamente o processo acima, considerando a cada iteração t um novo conjunto Γ_t^* , que inclui todos os estimadores ocupando da primeira até a t -ésima posição no torneio de precisão local. Se houver, em algum momento, um desempate entre os rótulos escolhidos originariamente por Γ_1^* , então o rótulo que ganhou o desempate será a saída da *ensemble*. Caso $\Gamma_t^* = \Gamma$, isto é, todos os classificadores já tenham seus votos considerados, então o desempate é feito de modo aleatório entre os rótulos escolhidos por Γ_1^* . Para este caso, $N_v = 1$.
26. *DCS_LA*-2 [494]: Similar ao anterior só que são *duas* ($N_v = 2$) as amostras mais similares, pertencentes a R_i , a serem usadas com o intuito de avaliação da precisão local de cada componente.
27. *DCS_LA*-5 [494]: Similar ao anterior só que são *cinco* ($N_v = 5$) as amostras mais similares, pertencentes a R_i , a serem usadas com o intuito de avaliação da precisão local de cada componente.
28. *DT*⁽⁵⁾-E [263, 265]: O padrão $\mathbf{x} \in DN_{ts}$ é submetido a todos os K componentes, gerando, como resultado, o seu perfil de decisão $\mathbf{DP}(\mathbf{x})$. Tal matriz passa a ser comparada, via uma medida de similaridade $\mathcal{S}$, com M *gabaritos de decisão* (*decision templates*) de tamanho $K \times M$ calculados previamente. Um gabarito de decisão $\mathbf{DT}_j$ é interpretado como o perfil de decisão mais típico para a j -ésima classe e é obtido pela média dos perfis de decisão das amostras $\mathbf{z}_i, i = 1, \dots, N_{tr}$, de DN_{tr}

⁽⁴⁾ *Dynamic Classifier Selection based on Local Accuracy*.

⁽⁵⁾ *Decision Templates*.

exibindo rótulo de classe j . Cada (k, t) -ésimo elemento de $\mathbf{DT}_j$ é calculado como sendo $dt_j(k, t) = \frac{\sum_{i=1}^{DN_{tr}} \text{Ind}(\mathbf{z}_i, j) a_k^t(\mathbf{z}_i)}{\sum_{i=1}^{DN_{tr}} \text{Ind}(\mathbf{z}_i, j)}$, onde $\text{Ind}(\mathbf{z}_i, j)$ denota uma função-indicadora auxiliar assumindo valor 1, se $\mathbf{z}_i$ tem rótulo de classe j , e 0, em caso contrário. Os resultados das comparações geram os valores-suporte μ_j para cada classe $j \in C$, ou seja, $\mu_j = \mathcal{S}(\mathbf{DT}_j, \mathbf{DP}(\mathbf{x}))$. Tais valores-suporte passam a ser cotejados, e a saída da *ensemble* será a classe com maior valor-suporte, isto é, $\Omega(\mathbf{x}) = \arg \max_{j \in C} \mu_j(\mathbf{x})$. Aqui, $\mathcal{S}$ é representada pela distância euclideana.

29. *DT-S1* [263, 265]: Similar ao anterior, só que a medida de similaridade $\mathcal{S}$ passa a ser dada por $\mathcal{S}_1(A, B) \equiv \frac{\|A \cap B\|}{\|A \cup B\|}$, sendo que A, B são interpretados como conjuntos *fuzzy* e $\cap$ e $\cup$ são t - e s -normas implementadas aqui como operadores de mínimo e máximo, respectivamente. Ademais, $\|\zeta\|$ denota a cardinalidade relativa ao conjunto *fuzzy* ζ sobre seu universo de discurso $U = \{u_1, \dots, u_q\}$, isto é, $\|\zeta\| = \frac{1}{q} \sum_{i=1}^q \rho_\zeta(u_i)$, com ρ_ζ representando a função de pertinência associada a ζ .
30. *DT-S2* [263, 265]: Similar ao anterior, só que a medida de similaridade $\mathcal{S}$ passa a ser dada por $\mathcal{S}_2(A, B) \equiv 1 - \|A \nabla B\|$, sendo $\|\cdot\|$ interpretado como anteriormente, ao passo que ∇ denota a diferença simétrica, ponto-a-ponto, entre seus operandos, ou seja, $\rho_{A \nabla B}(u) = |\rho_A(u) - \rho_B(u)|, u \in U$.
31. *DT-S3* [263, 265]: Similar ao anterior, só que a medida de similaridade $\mathcal{S}$ passa a ser dada por $\mathcal{S}_3(A, B) = 1 - \|A \Delta B\|$, sendo $\|\cdot\|$ interpretado como anteriormente, ao passo que Δ denota a seguinte operação: $\rho_{A \Delta B}(u) = \max\{\rho_{A \cap B^c}(u), \rho_{A^c \cap B}(u)\}, u \in U$, onde $\cap$ indica uma t -norma qualquer (no caso, o mínimo) e A^c indica o conjunto *fuzzy* complemento de A (i. e., $\rho_{A^c}(u) = 1 - \rho_A(u), u \in U$).
32. *DT-S4* [263, 265]: Similar ao anterior, só que a medida de similaridade $\mathcal{S}$ passa a ser dada por $\mathcal{S}_4(A, B) = 1 - \sup(A \nabla B)$, sendo ∇ interpretado como anteriormente. Aqui, “sup” denota o supremo dentre todos os valores de pertinência do conjunto *fuzzy* resultante de $A \nabla B$.
33. *DT-C* [263, 265]: Similar ao anterior, só que a medida de similaridade $\mathcal{S}$ passa a ser dada pelo índice de consistência $\mathcal{C}(A, B) = \sup(A \cap B)$, sendo “sup” e $\cap$ interpretados como anteriormente.
34. *DS⁽⁶⁾-A* [497]: Em contraposição a outros métodos de combinação embasados em medidas puramente estatísticas, operadores calcados na teoria da evidência proposta por Dempster & Shafer [499] apresentam a capacidade interessante de capturar incertezas ou conflitos nas decisões produzidas pelos componentes, cujas competências também podem ser estimadas de modo mais subjetivo. De acordo com este operador proposto por Xu *et al.* [497], para cada classificador $k \in \Gamma$, recupera-se, a partir da sua matriz de confusão, as suas taxas de reconhecimento, substituição e rejeição (aqui denotadas por $\epsilon_r^k, \epsilon_s^k, 1 - \epsilon_r^k - \epsilon_s^k$), a serem usadas para a definição da sua função de massa associada m_k sobre os elementos do conjunto de hipóteses Θ . Considerando uma nova amostra $\mathbf{x} \in DN_{ts}$, se o voto do k -ésimo componente for a classe de rejeição (M), então m_k só terá um ponto focal, o próprio Θ , ou

⁽⁶⁾Dempster-Shafer.

seja, $m_k(\Theta) = 1$. Caso contrário, supondo que $v_k(\mathbf{x}) = j$ e considerando θ_j hipótese atômica de Θ , então têm-se dois pontos focais, θ_j e $\bar{\theta}_j$, com $m_k(\theta_j) = \epsilon_r^k$ e $m_k(\bar{\theta}_j) = \epsilon_s^k$. Nesse caso, como esse classificador não “se pronuncia” a respeito de qualquer outra proposição $\theta_j^* \in \Theta, j^* \neq j$, então tem-se $m_k(\Theta) = 1 - m_k(\theta_j) - m_k(\bar{\theta}_j)$. Com todas as funções de massa relativas aos K componentes definidas, passa-se a aplicar a soma ortogonal com o intuito de se calcular a função de massa agregada para cada classe j (ou seja, para θ_j), considerando somente os componentes de Γ que tenham votado nessa classe, o que é denotado pelo subconjunto $\Gamma_j \subseteq \Gamma$. Isso se traduz em $m_j(\theta_j) = \bigoplus_{k_j \in \Gamma_j} m_{k_j}(\theta_j)$. Ademais, seja $C^* \subseteq C$ o conjunto de classes que tenham recebido pelo menos um voto dos K componentes. De posse de todos os m_j 's, com $j \in C^*$, pode-se, novamente, aplicar a soma ortogonal sobre tais funções de massa, de modo a se obter uma função de massa agregada m_{all} , dada por $m_{all} = \bigoplus_{j \in C^*} m_j(\theta_j)$. A partir de m_{all} , pode-se calcular os valores das funções-crença agregadas $bel(\theta_j, \mathbf{x}), \forall j \in C^*$. Xu *et al.* [497] realizaram uma série de manipulações algébricas, eliminando os valores de crença relativos às hipóteses $\theta_{j'}, j' \notin C^*$, com o intuito de se chegar a uma formulação aproximada que fosse mais eficiente computacionalmente — segundo os autores, o custo computacional de todo o método, com essas aproximações, passou a ser da ordem de $O(M)$. Tal formulação aproximada leva à definição de $bel(\theta_j, \mathbf{x})$ como sendo:

$$bel(\theta_j, \mathbf{x}) = \begin{cases} \kappa \left\{ \frac{m_j(\theta_j)}{1-m_j(\theta_j)} \Lambda_2 + \frac{m_j(\Theta)}{m_j(\theta_j)} \Lambda_3 \right\} & \text{se } Card(C^*) = M \text{ ou} \\ & Card(C^*) = M - 1 \text{ e } j = M \text{ (rejeição)} \\ \kappa \frac{m_j(\theta_j)}{1-m_j(\theta_j)} \Lambda_2 & \text{em todos os outros casos,} \end{cases}$$

onde $\Lambda_r (r = \{2, 3\})$ e κ são constantes definidas via fórmulas fechadas, envolvendo as funções de massa de todas as classes pertencentes a C^* . Finalmente, o supremo dentre todos os valores da função-crença agregada $bel(\cdot, \mathbf{x})$ indicará a classe vencedora da *ensemble*, digamos j^+ , caso esse valor seja maior que um *threshold* de segurança predefinido, ou seja, $bel(\theta_{j^+}, \mathbf{x}) \geq \tau_{DSA}$. Se essa restrição não for satisfeita, o sistema rejeita a amostra.

35. DS-B [497]: Variação do método anterior, em que também se considera, para efeito de determinação da classe vencedora j^+ , o valor da função-crença relativa ao complemento de cada classe $j \in C^*$ (quer dizer, com relação à hipótese $\bar{\theta}_j$). A fórmula aproximada para $bel(\bar{\theta}_j, \mathbf{x})$ é:

$$bel(\bar{\theta}_j, \mathbf{x}) = \begin{cases} \kappa \left\{ \left(\Lambda_1 - \frac{m_j(\theta_j) - m_j(\bar{\theta}_j)}{1 - m_j(\theta_j)} \right) \Lambda_2 - \Lambda_3 \right\} & \text{se } Card(C^*) = M \\ \kappa \left(\Lambda_1 - \frac{m_j(\theta_j) - m_j(\bar{\theta}_j)}{1 - m_j(\theta_j)} \right) \Lambda_2 & \text{se } Card(C^*) < M \text{ e } \kappa \leq Card(C^*) \\ \kappa \left(\Lambda_1 - \frac{m_j(\theta_j)}{1 - m_j(\theta_j)} \right) \Lambda_2 & \text{em todos os outros casos,} \end{cases}$$

onde Λ_1 é outra constante de cálculo exato. Agora, o supremo dentre todas as *diferenças* entre os valores pró e contra uma hipótese atômica qualquer, relativas à função-crença $bel(\cdot, \mathbf{x})$, indicará a classe vencedora j^+ . Caso o valor resultante seja maior que um *threshold* de segurança predefinido, ou seja, $bel(\theta_{j^+}, \mathbf{x}) - bel(\bar{\theta}_{j^+}, \mathbf{x}) \geq \tau_{DSB}$, então $\Omega(\mathbf{x}) = j^+$. Se essa restrição não for satisfeita, o sistema rejeita a amostra.

36. *DS-D1* [395]: O procedimento proposto por Xu *et al.* [497] de combinação de estimadores baseando-se na teoria de Dempster & Shafer só utiliza informações simples (i. e., votos dos componentes) quando da formação das funções de massa. Além do mais, ignora o fato de que classificadores normalmente não mantêm o mesmo nível de desempenho para diferentes classes ou regiões de entrada. Nesse sentido, Rogova [395] projetou um método alternativo de se calcular as evidências, empregando *medidas de proximidade* ϕ entre vetores de referência $\mathbf{E}_j^k = [E_{j1}^k, \dots, E_{jj}^k, \dots, E_{jM}^k]$ e vetores de suporte $\mathbf{a}_k(\mathbf{x})$ para cada componente. O vetor de referência associado à classe j e ao componente k é obtido pela média dos vetores de grau de suporte de k para as amostras de treinamento rotuladas por j , ou seja, $\mathbf{E}_j^k = \frac{1}{N_j} \sum_{n \in C_j} \mathbf{a}_k(\mathbf{z}_n)$, onde $C_j \subseteq C$ contém os índices das amostras de treinamento $\mathbf{z}_n \in DN_{tr}$ referentes à classe j e N_j é a cardinalidade de C_j . Para o k -ésimo classificador, a sua medida de proximidade para com a classe j passa a ser mapeada nas seguintes funções de massa:

$$\begin{aligned} m_j^k(\theta_j) &= \phi(\mathbf{E}_j^k, \mathbf{a}_k) & m_j^k(\Theta) &= 1 - \phi(\mathbf{E}_j^k, \mathbf{a}_k) \\ m_{-j}^k(\bar{\theta}_j) &= 1 - \prod_{l \neq j} (1 - \phi(\mathbf{E}_l^k, \mathbf{a}_k)) & m_{-j}^k(\Theta) &= \prod_{l \neq j} (1 - \phi(\mathbf{E}_l^k, \mathbf{a}_k)), \end{aligned}$$

considerando o mesmo quadro de discernimento Θ definido para o método anterior (ver Seção B.1). Combinando todo o conhecimento disponível acerca de θ_j via classificador k , obtém-se a evidência $e_j(\mathbf{a}_k) = m_j^k \oplus m_{-j}^k$ pró classe j como sendo $e_j(\mathbf{a}_k) = \frac{\phi(\mathbf{E}_j^k, \mathbf{a}_k) \prod_{l \neq j} (1 - \phi(\mathbf{E}_l^k, \mathbf{a}_k))}{1 - \phi(\mathbf{E}_j^k, \mathbf{a}_k) \left[1 - \prod_{l \neq j} (1 - \phi(\mathbf{E}_l^k, \mathbf{a}_k)) \right]}$. Finalmente, evidências coletadas por todos os classificadores podem ser combinadas, via soma ortogonal, para se obter uma medida de confiança geral para cada classe j em relação à entrada $\mathbf{x}$: $e_j(\mathbf{x}) = \bigoplus_{k \in \Gamma} e_j(\mathbf{a}_k)$. Esse $e_j(\mathbf{x})$, depois de uma normalização adequada, pode ser interpretado como uma função de evidência bayesiana associada a funções de massa não-nulas atuando somente sobre hipóteses atômicas. Disso, $e_j(\mathbf{x}) = \Lambda \prod_{k \in \Gamma} e_j(\mathbf{a}_k)$, sendo Λ uma constante de normalização. Finalmente, a saída da *ensemble* para $\mathbf{x}$ será o rótulo j^* , $\Omega(\mathbf{x}) = j^*$, caso $e_{j^*} = \max_{j \in C} e_j(\mathbf{x})$. Para esta variação, a função de proximidade usada foi $\phi_1(\mathbf{E}_j^k, \mathbf{a}_k) = 1 - |E_{jj}^k - a_k^j|$.

37. *DS-D2* [395]: Similar ao anterior, só que a função de proximidade passa a ser dada por $\phi_2(\mathbf{E}_j^k, \mathbf{a}_k) = 1 - \|\mathbf{E}_j^k, \mathbf{a}_k\|$, sendo $\|\cdot\|$ uma norma qualquer (no caso, a euclidiana).
38. *DS-D3* [395]: Similar ao anterior, só que a função de proximidade passa a ser dada por $\phi_3(\mathbf{E}_j^k, \mathbf{a}_k) = \exp(-\|\mathbf{E}_j^k - \mathbf{a}_k\|^m)$, sendo $m = 2$ nos experimentos conduzidos no Cap. 7.
39. *DS-D4* [395]: Similar ao anterior, só que a função de proximidade passa a ser dada por $\phi_4(\mathbf{E}_j^k, \mathbf{a}_k) = \frac{(1 + \|\mathbf{E}_j^k - \mathbf{a}_k\|^m)^{-1}}{\sum_{z \in C} (1 + \|\mathbf{E}_z^k - \mathbf{a}_k\|^m)^{-1}}$, sendo $m = 2$ nos experimentos conduzidos no Cap. 7.
40. *DS-D5* [395]: Similar ao anterior, só que a função de proximidade passa a ser dada por $\phi_5(\mathbf{E}_j^k, \mathbf{a}_k) = (1 + \|\mathbf{E}_j^k - \mathbf{a}_k\|^m)^{-1}$, sendo $m = 2$ nos experimentos conduzidos no Cap. 7.
41. *DS-D6* [395]: Similar ao anterior, só que a função de proximidade passa a ser dada por $\phi_6(\mathbf{E}_j^k, \mathbf{a}_k) = \cos^m(\alpha_j^k)$, sendo α_j^k o ângulo entre $\mathbf{E}_j^k$ e $\mathbf{a}_k$, e $m = 2$ nos experimentos conduzidos no Cap. 7.

42. *Sequel* [23]: Outro método que aloca regiões a componentes, guardando semelhança com a abordagem ME. Aqui, interpreta-se cada grau de suporte a_k^j de cada classificador k como uma estimativa da probabilidade *a posteriori* suportando a hipótese de que “uma certa amostra $\mathbf{x}$ seja uma instância de um dado conceito-alvo H_j ”. Para cada componente $k \in \Gamma$, define-se o limiar δ_k como o *mínimo* entre o mais alto grau de suporte dado por ele para uma amostra *negativa* qualquer e o mais alto grau de suporte produzido por ele para uma amostra *positiva* qualquer, sendo consideradas negativas as amostras de DN_{tr} em que gerou resultado errôneo e positivas aquelas em que acertou. Em seguida, passa-se a calcular o *escore de competência* de cada componente k , sendo este dado pela razão entre o número de casos positivos em que seu maior grau de suporte foi maior que δ_k e o número total de amostras de treinamento pertencentes à região do espaço de entrada localizada acima desse limiar (i. e., contando tanto os exemplos positivos como negativos). O componente com maior escore de competência, digamos k_1^* , ficará responsável por uma certa região de competência $DN_1^* \subseteq DN_{tr}$. Para determinar o próximo componente com maior competência, k_2^* , e a sua região de competência, DN_2^* , repete-se a rotina acima, desconsiderando as amostras pertencentes a DN_1^* . Esse processo é repetido até que o escore de competência do $(z+1)$ -ésimo componente mais competente fique abaixo de um *threshold* predefinido, denominado τ_{seq} . O resultado desse procedimento iterativo é uma lista de componentes $\Gamma^* = \{k_1^*, \dots, k_z^*\}$ (e seus escores de competência associados) a serem aplicados em sequência. Para cada novo padrão $\mathbf{x} \in DN_{ts}$, recorre-se ao componente k_1^* e verifica-se se o seu grau de suporte mais alto para um dado conceito (classe) j_1 ($a_{k_1^*}^{j_1}$) é maior que o seu limiar $\delta_{k_1^*}$. Se sim, atribui-se essa probabilidade à amostra $\mathbf{x}$, tornando j_1 a saída da *ensemble* (i. e., $\Omega(\mathbf{x}) = j_1$). Se não, o segundo componente mais competente, k_2^* , passa a ser convocado. Se a maior probabilidade produzida por k_2^* ($a_{k_2^*}^{j_2}$) for maior que seu limiar $\delta_{k_2^*}$, então multiplica-se tal probabilidade pelo limiar do componente anterior, ou seja, $a_{k_2^*}^{j_2} \times \delta_{k_1^*}$, sendo esta a saída “probabilística” agregada da combinação e j_2 o rótulo vencedor da *ensemble* (i. e., $\Omega(\mathbf{x}) = j_2$). Se o segundo componente também não conseguir decidir, repassa-se a amostra para o terceiro mais competente e daí por diante, até que algum componente de Γ^* consiga decidir. Para o z -ésimo componente de Γ^* , a saída “probabilística” agregada será dada por $\left(\prod_{t=1}^{z-1} \delta_{k_t^*}\right) \times a_{k_z^*}^{j_z}$, ao passo que j_z será o rótulo de saída da *ensemble* (i. e., $\Omega(\mathbf{x}) = j_z$). Caso nenhum componente de Γ^* consiga produzir uma decisão confiável para $\mathbf{x}$, apela-se para um operador *default* que atuará sobre os graus de suporte de todos os classificadores — no caso, tal operador é a *Média*.
43. $CS^{(7)}$ -KMeans [264]: Outro método baseado em seleção de estimadores, só que o particionamento das regiões do espaço de entradas (a serem alocadas aos componentes) é feito mediante um processo de *agrupamento* (*clustering*) sobre os dados de DN_{tr} . Aqui, o agrupamento é feito via método *K-Means* [197], com N_{cl} denotando o número (predefinido) de *clusters* e $\mathbf{p}_l$, o protótipo referente ao *cluster* C_l , para $l = 1, \dots, N_{cl}$. Para cada *cluster* C_l (definindo a região R_l), passa-se a estimar a competência local dos K componentes, considerando somente a taxa de reconhecimento sobre as amostras de DN_{tr} que pertençam a C_l . Para a l -ésima região, guarda-se o classificador mais competente, indicado aqui por k_l^* . Dada uma nova amostra $\mathbf{x} \in DN_{ts}$, encontra-se, via uma certa medida

⁽⁷⁾ *Clustering & Selection.*

de distância — no caso, a usada foi a euclideana —, o seu protótipo mais próximo, digamos, $\mathbf{p}_x$. Posteriormente, recupera-se o classificador k_x^* referente à região de $\mathbf{p}_x$, sendo que seu voto determinará a saída da *ensemble*, ou seja, $\Omega(\mathbf{x}) = v_{k_x^*}(\mathbf{x})$. Nos experimentos conduzidos, $N_{cl} = 4$.

44. *CS-FCMeans* [264]: Idêntico ao anterior, só que o particionamento das regiões é *fuzzy* e propiciado mediante o algoritmo *Fuzzy C-Means* [52].

B.1 Definições básicas da teoria da evidência de Dempster-Shafer

Seja $\Theta = \{\theta_1, \dots, \theta_M\}$ um conjunto finito de hipóteses possíveis — neste contexto, cada uma concernente a uma proposição “ $\mathbf{x} \in j$ ”, $j \in C$ —, conhecido como *quadro de discernimento* (de *frame of discernment*). Uma *função de massa* m é uma função que atribui um valor em $[0, 1]$ a todo subconjunto $\mathcal{A} \subseteq \Theta$ e satisfaz as seguintes restrições: $m(\emptyset) = 0$, $\sum_{\mathcal{A} \subseteq \Theta} m(\mathcal{A}) = 1$. Enquanto, pela Teoria de Probabilidade, uma *medida de probabilidade* é atribuída a hipóteses atômicas θ_i , $m(\mathcal{A})$ representa a parte da *crença* total que suporta a hipótese composta $\mathcal{A}$ mas que não suporta qualquer coisa mais específica que $\mathcal{A}$ (i. e., seus subconjuntos). Um subconjunto $\mathcal{A}^*$ de Θ que satisfaz a $m(\mathcal{A}^*) > 0$ é considerado como um *ponto focal* de Θ , sendo que a seguinte desigualdade deve ser sempre satisfeita: $m(\mathcal{A}^*) + m(\bar{\mathcal{A}}^*) = 1$, onde $\bar{\mathcal{A}}^*$ representa o complemento de $\mathcal{A}^*$. Por sua vez, uma função-crença $bel(\cdot)$, associada à função de massa $m(\cdot)$, atribui um valor em $[0, 1]$ a todo subconjunto *não-vazio* $\mathcal{B} \subseteq \Theta$. Ela indica um “grau de crença em $\mathcal{B}$ ” e é definida como $bel(\mathcal{B}) = \sum_{\mathcal{A} \subseteq \mathcal{B}} m(\mathcal{A})$. Destarte, $m(\cdot)$ pode ser interpretada como uma generalização de uma função densidade de probabilidade, ao tempo que $bel(\cdot)$ seria uma generalização de uma função de probabilidade. Considerando duas funções de massa $m_1(\cdot)$ e $m_2(\cdot)$ associadas a duas funções-crença $bel_1(\cdot)$ e $bel_2(\cdot)$, então $m_1(\cdot)$ e $m_2(\cdot)$ podem ser combinadas a fim de se gerar a função de massa atribuída a $\mathcal{C} \subset \Theta$, de acordo com a seguinte regra, conhecida como *soma ortogonal*:

$$m(\mathcal{C}) = (m_1 \oplus m_2)(\mathcal{C}) = \frac{\sum_{r,s, \mathcal{A}_r^* \cap \mathcal{B}_s^* = \mathcal{C}} m_1(\mathcal{A}_r^*) m_2(\mathcal{B}_s^*)}{1 - \sum_{r,s, \mathcal{A}_r^* \cap \mathcal{B}_s^* = \emptyset} m_1(\mathcal{A}_r^*) m_2(\mathcal{B}_s^*)}, \mathcal{C} \neq \emptyset,$$

sendo $\mathcal{A}_r^*$ e $\mathcal{B}_s^*$ elementos focais de bel_1 e bel_2 , respectivamente. Essa regra pode ser estendida para lidar com M funções-crença repetindo-se a fórmula acima para cada nova função, ou seja, $((bel_1 \oplus bel_2) \oplus bel_3) \dots \oplus bel_M) = \bigoplus_{j=1}^M bel_j$.

Apêndice C

Descrição dos problemas de classificação

“A fábrica do futuro terá apenas dois operários: Um homem e um cachorro. Função do homem: alimentar o cachorro. Função do cachorro: não deixar o homem tocar nas máquinas.”

Walter Block

A Tab. C.1 traz uma breve caracterização dos conjuntos de dados empregados nos experimentos de classificação de padrões conduzidos ao longo desta tese (Caps. 7 e 8), sendo que uma descrição textual desses conjuntos é fornecida na Tab. C.2.

Tab. C.1: Conjunto de problemas de classificação investigados ao longo da tese. Legenda: NC: Número de classes; NA: Número de amostras; DC: Distribuição das classes; NAt: Número de atributos; TA: Tipos dos atributos (Numérico ou Discreto); AF: Atributos faltantes? (Sim ou Não); R: Ruído (Sim ou Não). (Fontes: Alon *et al.* [13], Blake & Merz [54] e Trippi & Turban [453]).

Nome	NC	NA	DC	NAt	TA	AF	R
<i>Bankruptcy</i>	2	129	(48,8%;51,2%)	5	N	N	N
<i>BCD699</i>	2	699	(65,5%;34,5%)	9	N	S	N
<i>Car</i>	4	1728	(70,023%;22,222%;3,993%;3,762%)	6	D	N	N
<i>Credit</i>	2	690	(44,5%;55,5%)	14	D & N	S	N
<i>Dermatology</i>	6	366	(30,60%;16,67%;19,67%;13,39%;14,21%;5,46%)	34	N	S	N
<i>Diabetes</i>	2	768	(65,1%;34,9%)	8	N	N	N
<i>Glass</i>	6	214	(32,71%;7,94%;35,51%;6,07%;4,21%;13,55%)	8	N	N	N
<i>Heart</i>	2	303	(54,12%;45,87%)	13	D & N	S	N

Continua na próxima página

Tab. C.1: Conjunto de problemas de classificação. (cont.)

Nome	NC	NA	DC	NAt	TA	AF	R
<i>Hepatitis</i>	2	155	(20,64%;79,36%)	19	D & N	S	N
<i>Ionosphere</i>	2	351	(64%;36%)	34	D & N	N	N
<i>Iris</i>	3	150	(33,33%;33,33%;33,33%)	4	N	N	N
<i>Monk1</i>	2	124	(50%;50%)	6	D	N	N
<i>Monk3</i>	2	122	(49,2%;50,8%)	6	D	N	S
<i>Sonar</i>	2	208	(53,36%;46,64%)	60	N	N	N
<i>Thyroid</i>	3	7200	(92,58%;2,31%;5,11%)	21	D & N	N	N
<i>Vehicle</i>	4	946	(25,37%;25,37%;25,37%;23,89%)	18	N	N	N
<i>Wine</i>	3	178	(33,15%;39,88%;26,97%)	13	N	N	N
<i>Zoo</i>	7	101	(40,59%;19,8%;4,95%;12,87%;3,96%;7,92%;9,90%)	17	D & N	N	N
<i>MCR</i>	2	62	(35,5%;64,5%)	2000	D	N	N

Tab. C.2: Descrição textual dos problemas da Tab. C.1.

Índice	Nome	Objetivo da tarefa de estimação
1	<i>Bankruptcy</i>	Discriminar entre empresas com ou sem risco de falência
2	<i>BCD699</i>	Discriminar tumores malignos e benignos baseando-se em características de células específicas
3	<i>Car</i>	Prever qual conceito (ruim, regular, bom, muito bom) foi dado a um certo leiaute de automóvel
4	<i>Credit</i>	Aprovar ou não o crédito a um dado cliente mediante a análise de seu perfil de crédito
5	<i>Dermatology</i>	Determinar o tipo de doença de pele
6	<i>Diabetes</i>	Diagnosticar a presença ou não de diabetes a partir de várias medidas fisiológicas e testes clínicos
7	<i>Glass</i>	Identificar o tipo de vidro com base nas características físicas da amostra
8	<i>Heart</i>	Diagnosticar a presença ou ausência de risco de doença coronária (enfarto) a partir dos resultados de vários testes clínicos
9	<i>Hepatitis</i>	Dentre pacientes com hepatite, indicar quais deles irão falecer ou não
10	<i>Ionosphere</i>	Classificar sinais de radar provenientes da ionosfera
11	<i>Iris</i>	Predizer a classe de uma planta (íris) de acordo com o formato de sua folha
12	<i>Monk1</i>	Conjunto de dados artificial usado para testar certas propriedades de indução
13	<i>Monk3</i>	Variação do problema anterior com introdução de ruído nas classes

Continua na próxima página

Tab. C.2: Descrição textual dos problemas. (cont.)

Índice	Nome	Objetivo da tarefa de estimação
14	<i>Sonar</i>	Distinguir entre materiais metálicos ou rochosos de acordo com os níveis de intensidade de sinais de sonar enviados sob diferentes condições (ângulos, frequências, etc.)
15	<i>Thyroid</i>	Diagnosticar a condição física de pacientes conduzidos a uma clínica em um dentre três possíveis estados: “normal”, “hipotireoidismo primário”, ou “hipotireoidismo compensado”
16	<i>Vehicle</i>	Classificar uma dada silhueta 2D em um dentre quatro tipos de veículos usando um conjunto de atributos físicos
17	<i>Wine</i>	Determinar a origem do vinho mediante a análise de seus componentes químicos
18	<i>Zoo</i>	Conjunto de dados artificial em que diferentes espécies devem ser categorizadas de acordo com suas características fisiológicas, etológicas e ecológicas
19	<i>MCR</i>	Diagnosticar a predisposição de pacientes a desenvolver câncer de intestino com base nos níveis de intensidade de expressão gênica (obtidos via técnica de <i>microarray</i>) de amostras de tecido dessa região

Referências Bibliográficas

- [1] Statlog *home page*: <http://www.phys.uni.torun.pl/kmk/projects/datasets-stat.html>.
- [2] RoboCup *official home page*: <http://www.robocup.org>.
- [3] TeamBots *home page*: <http://www.teambots.org>.
- [4] H. ABBASS. An evolutionary artificial neural network approach for breast cancer diagnosis. *Artificial Intelligence in Medicine* **25**, 265–281 (2002).
- [5] Y. S. ABU-MOSTAFA. The Vapnik-Chervonenkis dimension: Information versus complexity in learning. *Neural Computation* **1**, 312–317 (1989).
- [6] Y. S. ABU-MOSTAFA E A. F. ATYIA. Introduction to financial forecasting. *Applied Intelligence* **6**, 205–213 (1996).
- [7] D. ACKLEY E M. LITTMAN. A case for Lamarckian evolution. Em C. G. LANGTON, editor, “Artificial Life III”, páginas 3–11. Addison Wesley (1994).
- [8] A. AGAR, B. DOYLE, K. KUOK E K. TANIE. Emergent cooperative strategies for robot team sports. *International Journal of Intelligent Automation and Soft Computing* **6**(1), 45–56 (2000).
- [9] M. R. AHMADZADEH E M. PETROU. Use of Dempster-Shafer theory to combine classifiers which use different class boundaries. *Pattern Analysis and Applications* **6**, 41–46 (2003).
- [10] U. AICKELIN E L. BULL. Partnering strategies for fitness evaluation in a pyramidal evolutionary algorithm. Em “Procs. of Genetic and Evolutionary Computation Conference”, páginas 263–270 (2002).
- [11] U. AICKELIN E L. BULL. On the application of hierarchical coevolutionary genetic algorithms: Recombination and evaluation partners. *Journal of Applied System Sciences* **4**(2) (2003). Fascículo especial sobre “Real Life Applications of Nature Inspired Combinatorial Heuristics”.
- [12] J. F. ALLEN. Maintaining knowledge about temporal intervals. *Communications of ACM* **26**(11), 123–154 (1983).
- [13] U. ALON, N. BARKAI, D. A. NOTTERMAN, K. GISH, S. YBARRA, D. MACK E A. J. LEVINE. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon cancer tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences* **96**(12), 6745–6750 (1999).
- [14] E. ALPAYDIN E M. I. JORDAN. Local linear perceptrons for classification. *IEEE Transactions on Neural Networks* **7**(3), 788–792 (1996).
- [15] J. M. AMABIS, G. R. MARTHO E Y. MIZUGUCHI. “Biologia”, tomo 3. Editora Moderna (1982).

- [16] L. ANASTASAKIS E N. MORT. Applying a feedforward neural network for the prediction of the USD/GBP exchange rate. Em "Procs. of Third Conference in Technology and Automation", Piraeus, Greece (outubro 2000). Dept. of Automation—TEI of Piraeus.
- [17] H. C. ANDERSEN E A. C. TSOY. A constructive algorithm for the training of a multilayer perceptron based on the genetic algorithm. *Complex Systems* 7(4), 249–268 (1993).
- [18] T. ANDOU. Refinement of soccer agents' positions using reinforcement learning. Em H. KITANO, editor, "RoboCup-97: Robot Soccer World Cup I", páginas 373–388. Springer-Verlag (1998).
- [19] P. J. ANGELINE E J. B. POLLACK. Competitive environments evolve better solutions for complex tasks. Em "Procs. of the Fifth International Conference on Genetic Algorithms", páginas 264–270. Morgan Kaufmann (1995).
- [20] P. J. ANGELINE, G. M. SAUNDERS E J. P. POLLACK. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks* 5(1), 54–65 (janeiro 1994).
- [21] J. ARMSTRONG E F. COLLOPY. Error measures for generalizing about forecasting methods—Empirical comparisons. *International Journal of Forecasting* 8(1), 69–80 (1992).
- [22] J. S. ARMSTRONG, editor. "Principles of Forecasting: A Handbook for Researchers and Practitioners". Kluwer Academic Publishers (2001).
- [23] L. ASKER E R. MACLIN. Ensembles as a sequence of classifiers. Em "Procs. of 15th. International Joint Conference on Artificial Intelligence", tomo 2, páginas 860–865 (1997).
- [24] W. ATMAR. Notes on the simulation of evolution. *IEEE Transactions on Neural Networks* 5(1), 130–148 (1994).
- [25] G. AUDA E M. KAMEL. CMNN: Cooperative modular neural networks for pattern recognition. *Pattern Recognition Letters* 18, 1391–1398 (1997).
- [26] G. AUDA E M. KAMEL. EVOL: Ensembles voting on-line. Em "Procs. of IEEE International Conference on Neural Networks", páginas 1356–1360 (1998).
- [27] G. AUDA E M. KAMEL. Modular neural network classifiers: A comparative study. *Journal of Intelligent and Robotic Systems* 21, 117–129 (1998).
- [28] R. AXELROD. The evolution of strategies in the iterated prisoners' dilemma. Em Davis [109], páginas 32–41.
- [29] R. AXELROD. "The Complexity of Cooperation: Agent-based Models of Competition and Collaboration". Princeton University Press (1997).
- [30] T. BÄCK, D. FOGEL E Z. MICHALEWICZ, editores. "Handbook of Evolutionary Computation". Oxford University Press, New York (1997).
- [31] T. BÄCK, U. HAMMEL E H.-P. SCHWEFEL. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation* 1(1), 3–17 (1997).
- [32] D. BAHLER E L. NAVARRO. Methods for combining heterogeneous sets of classifiers. Em "Procs. of 17th National Conference on Artificial Intelligence (AAAI)—Workshop on New Research Problems for Machine Learning" (2000).
- [33] J. BAKER. Reducing bias and inefficiency in the selection algorithm. Em "Procs. of the Second International Conference on Genetic Algorithms", páginas 14–21 (1987).

- [34] T. BALCH. Learning roles: Behavioral diversity in robot teams. Relatório técnico GIT-CC-97-12, College of Computing, Georgia Institute of Technology (1997).
- [35] J. M. BALDWIN. A new factor in evolution. *American Naturalist* **30**, 441–451 (1896).
- [36] R. BALLINI. “Análise e previsão de vazões utilizando modelos de séries temporais, redes neurais e redes neurais nebulosas”. Tese de doutorado, Faculdade de Engenharia Elétrica e de Computação — Unicamp (2000).
- [37] W. BANTHAF, P. NORDIN, R. E. KELLER E F. D. FRANCONI. “Genetic Programming—An Introduction”. Morgan Kaufmann (1999).
- [38] M. BARBUCEANU E M. S. FOX. Capturing and modeling coordination knowledge for multi-agent systems. *International Journal of Cooperative Information Systems* **5**(2–3), 275–314 (1996).
- [39] A. R. BARRON. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory* **39**(3), 930–945 (1993).
- [40] R. BATTITI E A. M. COLLA. Democracy in neural nets: Voting schemes for classification. *Neural Networks* **7**, 691–707 (1994).
- [41] E. BAUER E R. KOHAVI. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning* **36**, 105–142 (1999).
- [42] E. BAUM E D. HAUSLER. What size net gives valid generalization? *Neural Computation* **1**, 151–160 (1989).
- [43] W. G. BAXT. Improving the accuracy of an artificial neural network using multiple differently trained networks. *Neural Computation* **4**(5), 135–144 (1992).
- [44] A. L. C. BAZZAN. Evolution of coordination as a metaphor for learning in multi-agent systems. Em Weiß e Sen [477], páginas 117–136.
- [45] D. BEASLEY, D. R. BULL E R. R. MARTIN. An overview of genetic algorithms: Parts 1 & 2. *University Computing* **15**(2 & 4), 58–69, 170–181 (1993).
- [46] L. BECKER E M. SESHADRI. Cooperative coevolution of technical trading rules. Relatório técnico WPI-CS-TR-03-15, Worcester Polytechnic Institute Computer Science Department (2003).
- [47] R. K. BELEW, J. MCINERNEY E N. N. SCHRAUDOLPH. Evolving networks: Using the genetic algorithm with connectionist learning. Em C. G. LANGTON, C. TAYLOR, J. D. FARMER E S. RASMUSSEN, editores, “Artificial Life II”, páginas 511–547. Addison-Wesley, Redwood City, CA (1992).
- [48] R. K. BELEW E M. MITCHELL, editores. “Adaptive Individuals in Evolving Populations: Models and Algorithms”. Addison-Wesley (1996).
- [49] J. A. BENEDIKTSSON E P. H. SWAIN. Consensus theoretic classification methods. *IEEE Transactions on Systems, Man, and Cybernetics* **22**, 688–704 (1992).
- [50] J. M. BENÍTEZ, J. L. CASTRO E I. REQUENA. Are artificial neural networks black boxes? *IEEE Transactions on Neural Networks* **8**(5), 1156–1164 (1997).
- [51] P. J. BENTLEY, editor. “Evolutionary Design by Computers”. Morgan Kaufmann, San Francisco (1999).
- [52] J. C. BEZDEK, R. J. HATHAWAY, M. J. SABIN E W. T. TUCKER. Convergence theory for fuzzy c-means: Counterexamples and repairs. *IEEE Transactions on Systems, Man, and Cybernetics* **17**(5), 873–877 (1987).
- [53] C. M. BISHOP, editor. “Neural Networks for Pattern Recognition”. Oxford University Press (1995).

- [54] C. L. BLAKE E C. J. MERZ. "The UCI Repository of Machine Learning". University of California at Irvine, <http://ics.uci.edu/~mllearn/MLRepository.html> (1998).
- [55] A. L. BLUM E R. L. RIVEST. Training a 3-node neural network is NP-complete. *Neural Networks* 5, 117–127 (1992).
- [56] E. BONABEAU, M. DORIGO E G. THERAULAZ. "Swarm Intelligence: From Natural to Artificial Systems". Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press (1999).
- [57] B. BOSER, I. GUYON E V. VAPNIK. A training algorithm for optimal margin classifiers. Em "Procs. of the Fifth Conference on Computational Learning Theory", páginas 144–152, New York (1992). ACM.
- [58] A. P. BRAGA, A. C. P. L. F. DE CARVALHO E T. B. LUDERMIR. Redes neurais artificiais. Em Rezende [389], páginas 141–168.
- [59] M. BRAMEIER E W. BANZHAF. Evolving teams of predictors with genetic programming. *Genetic Programming and Evolvable Machines* 2(4), 381–407 (2001).
- [60] J. BRANKE. "Evolutionary Optimization in Dynamic Environments". Kluwer Academic Publishers (2001).
- [61] L. BREIMAN. Bagging predictors. *Machine Learning* 24(2), 123–140 (1996).
- [62] L. BREIMAN. Arcing classifiers. *Annals of Statistics* 26(3), 801–849 (1998).
- [63] L. BREIMAN. Combining predictors. Em Sharkey [426], páginas 31–50.
- [64] R. A. BROOKS. Intelligence without representation. *Artificial Intelligence* 47, 139–159 (1991).
- [65] D. BRUCE E R. MIKKULAINEN. Evolving populations of expert neural networks. Em "Procs. of the Genetic and Evolutionary Computation Conference (GECCO)", páginas 251–257, San Francisco, California, USA (2001). Morgan Kaufmann.
- [66] L. BULL. On coevolutionary genetic algorithms. *Soft Computing* 5, 201–207 (2001).
- [67] L. BULL E T. C. FOGARTY. Artificial symbiogenesis. *Artificial Life* 2(3), 269–292 (1995).
- [68] C. J. C. BURGESS. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2(2), 121–167 (1998).
- [69] T. CAELLI, L. GUAN E W. WEN. Modularity in neural computing. *Proceedings of the IEEE* 87(9), 1497–1518 (1999).
- [70] R. CALABRETTA, S. NOLFI, D. PARISI E G. P. WAGNER. Duplication of modules facilitates the evolution of functional specialization. *Artificial Life* 6, 69–84 (2000).
- [71] E. CANTÚ-PAZ. "Efficient and Accurate Parallel Genetic Algorithms". Kluwer Academic Publishers (2000).
- [72] G. CASTELLANO, A. M. FANELLI E M. PELILLO. An iterative pruning algorithm for feedforward neural networks. *IEEE Transactions on Neural Networks* 8(3), 519–531 (1997).
- [73] T. CATFOLIS E K. MEERT. Hybridization and specialization of real-time recurrent learning-based neural networks. *Connection Science* 9(1), 51–70 (1997).
- [74] D. CHALLET E Y. C. ZHANG. Emergence of cooperation and organization in an evolutionary game. *Physica A* 246, 407 (1997).
- [75] L. D. CHAMBERS, editor. "Practical Handbook of Genetic Algorithms", tomo 1–3. CRC Press (1995, 1999).

- [76] Y. CHAUVIN E D. E. RUMELHART, editores. "Backpropagation: Theory, Architectures, and Applications". Lawrence Erlbaum Associates (1995).
- [77] V. CHERKASSKY, J. H. FRIEDMAN E H. WECHSLER, editores. "From Statistics to Neural Networks", tomo 136 de "Computer and Systems Sciences". Springer-Verlag (1994).
- [78] V. CHERKASSKY E Y. MA. Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Networks* **17**(1), 113–126 (2004).
- [79] V. CHERKASSKY E Y. Q. MA. Selection of meta-parameters for support vector regression. Em "Procs. of International Conference on Neural Networks", tomo 2415 de "LNCS", páginas 687–693. Springer-Verlag (2002).
- [80] V. CHERKASSKY E F. MULIER. "Learning from Data: Concepts, Theory, and Methods". John Wiley (1998).
- [81] S.-B. CHO. Pattern recognition with neural networks combined by genetic algorithms. *Fuzzy Sets and Systems* **103**, 339–347 (1999).
- [82] S.-B. CHO E J. KIM. Combining multiple neural networks by fuzzy integral for robust classification. *IEEE Transactions on Systems, Man, and Cybernetics* **25**(2), 380–384 (fevereiro 1995).
- [83] R. CLEMEN. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting* **5**, 559–583 (1989).
- [84] D. CLIFF E G. F. MILLER. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. Em "Advances in Artificial Life: Third European Conference on Artificial Life", tomo 929 de "LNAI", páginas 200–218. Springer (1995).
- [85] D. CLIFF E G. F. MILLER. Co-evolution of pursuit and evasion II: Simulation methods and results. Em "From Animals to Animats IV: Procs. of the Fourth International Conference on Simulation of Adaptive Behavior", páginas 506–515. MIT Press (1996).
- [86] A. L. V. COELHO. On the cooperation of fuzzy neural networks via a coevolutionary approach. Em "Procs. of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference", páginas 785–790, Vancouver, Canada (2001).
- [87] A. L. V. COELHO, C. A. M. LIMA E F. J. VON ZUBEN. GA-based selection of components for heterogeneous ensembles of support vector machines. Em "Procs. of IEEE International Conference on Evolutionary Computation (CEC)", tomo 6, páginas 2238–2245, Canberra, Australia (dezembro 2003). IEEE Press.
- [88] A. L. V. COELHO, C. A. M. LIMA E F. J. VON ZUBEN. Hybrid genetic training of gated mixtures of experts for nonlinear time series forecasting. Em "Procs. of IEEE International Conference on Systems, Man and Cybernetics (SMC)", tomo 5, páginas 4625–4630, Washington, D.C. (outubro 2003). IEEE Press.
- [89] A. L. V. COELHO, A. B. RAPOSO E I. L. M. RICARTE. Bringing flexibility to the specification and coordination of temporal dependencies among multimedia components. Em "Procs. of VII Brazilian Symposium on Multimedia and Hypermedia Systems (SBMIDIA)", páginas 37–51, Florianópolis, Brazil (outubro 2001). Brazilian Computing Society (SBC).
- [90] A. L. V. COELHO E I. L. M. RICARTE. A fuzzy authoring language for the high-level specification of component-based multimedia documents. Em A. LOTFI, J. GARIBALDI E R. JOHN, editores, "Procs. of 4th International Conference on Recent Advances on Soft Computing (RASC)", páginas 500–505, Nottingham (dezembro 2002). The Nottingham Trent University.

- [91] A. L. V. COELHO E I. L. M. RICARTE. On the fuzzy spatio-temporal specification of multimedia synchronisation scenarios. Em A. LOTFI E J. M. GARIBALDI, editores, "Applications and Science in Soft Computing", Advances in Soft Computing Series, páginas 53–60. Springer-Verlag (2004).
- [92] A. L. V. COELHO, D. WEINGAERTNER, R. R. GUDWIN E I. L. M. RICARTE. Emergence of multiagent spatial coordination strategies via artificial coevolution. *Computers & Graphics* **25**(6), 1013–1023 (dezembro 2001). Fascículo especial "Artificial Life: Towards a new generation of computer animation".
- [93] A. L. V. COELHO, D. WEINGAERTNER E F. J. VON ZUBEN. Assembling neural networks through the coevolution of heterogeneous neuronal templates. Em "Procs. of the International Joint Conference on Neural Networks (IJCNN)", Washington, D.C., USA (2001). IEEE Press.
- [94] A. L. V. COELHO, D. WEINGAERTNER E F. J. VON ZUBEN. Evolving heterogeneous neural networks for classification problems. Em "Procs. of the Genetic and Evolutionary Computation Conference (GECCO)", páginas 266–273, San Francisco, California, USA (2001). Morgan Kaufmann.
- [95] C. A. COELLO COELLO, D. A. VAN VELDHIJZEN E G. B. LAMONT. "Evolutionary Algorithms for Solving Multi-Objective Problems". Kluwer Academic Publishers (2002).
- [96] C. CORTES E V. VAPNIK. Support vector networks. *Machine Learning* **20**, 273–297 (1995).
- [97] T. M. COVER. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers* **14**, 326–334 (1965).
- [98] O. CORDON, F. HERRERA, F. HOFFMANN E L. MAGDALEÑA, editores. "Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases", tomo 19 de "Advances in Fuzzy Systems - Applications and Theory". World Scientific (2001).
- [99] O. CORDON, F. HERRERA E M. LOZANO. On the combination of fuzzy logic and evolutionary computation: A short review and bibliography. Em W. PEDRYCZ, editor, "Fuzzy Evolutionary Computation", páginas 57–77. Kluwer Academic Pub. (1997).
- [100] N. CRISTIANINI E J. SHAW-TAYLOR. "An Introduction to Support Vector Machines". Cambridge Press (2000).
- [101] J. M. DAIDA, C. S. GRASSO, S. A. STANHOPE E S. J. ROSS. Symbioticism and complex adaptive systems I: Implications of having symbiosis occur in nature. Em L. J. FOGEL, P. J. ANGELINE E T. BÄCK, editores, "Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming", páginas 177–186. MIT Press (1996).
- [102] J. M. DAIDA, S. J. ROSS E B. C. HANNAN. Biological symbiosis as a metaphor for computational hybridization. Em L. ESHELMAN, editor, "Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)", páginas 248–255. Morgan Kaufmann (1995).
- [103] P. DARWEN E X. YAO. Every niching method has its niche: Fitness sharing and implicit sharing compared. Em "Parallel Problem Solving from Nature IV", tomo 1141 de "LNCS", páginas 398–407. Springer (1996).
- [104] P. J. DARWEN E X. YAO. Speciation as automatic categorical modularization. *IEEE Transactions on Evolutionary Computation* **1**(2), 100–108 (1997).
- [105] C. DARWIN. "On the Origin of Species by means of Natural Selection". John Murray (1859).
- [106] D. DASGUPTA. Incorporating redundancy and gene activation mechanisms in genetic search. Em Chambers [75], páginas 303–316.

- [107] D. DASGUPTA, editor. "Artificial Immune Systems and Their Applications". Springer-Verlag (1998).
- [108] D. DASGUPTA E D. MCGREGOR. Designing application-specific neural networks using the structured genetic algorithm. Em "Procs. of the International Conference on Combinations of Genetic Algorithms and Neural Networks (COGANN)", páginas 87–96 (1992).
- [109] L. DAVIS, editor. "Genetic Algorithms and Simulated Annealing". Morgan Kaufmann, (1987).
- [110] R. DAVIS E R. SMITH. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence* **20**, 63–109 (1983).
- [111] R. DAWKINS E J. R. KREBS. Arms races between and within species. *Proceedings of the Royal Society of London, Series B* **205**, 489–511 (1979).
- [112] H. A. DE BARY. "Die Erscheinung der Symbiose". Verlag von Karl J. Trubner (1879).
- [113] L. N. DE CASTRO E J. I. TIMMIS. "Artificial Immune Systems: A New Computational Intelligence Approach". Springer-Verlag (2002).
- [114] N. C. DE CONDORCET. "Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix". Imprimerie Royale (1785).
- [115] C. A. M. DE LIMA. "Sistemas multi-SVMs: Uma abordagem unificada para comitês de máquinas e *support vector machines*". Tese de doutorado, Faculdade de Engenharia Elétrica e de Computação — Unicamp (2004). Em preparação.
- [116] K. DEB. "Multi-objective Optimization using Evolutionary Algorithms". Wiley (2001).
- [117] K. DECKER. TAEMS: A framework for environment centered analysis & design of coordination mechanisms. Em O'Hare e Jennings [350], páginas 429–448.
- [118] K. S. DECKER. Distributed problem solving techniques: A survey. *IEEE Transactions on Systems, Man, and Cybernetics* **17**, 729–740 (1987).
- [119] M. R. B. S. DELGADO, F. J. VON ZUBEN E F. GOMIDE. Coevolutionary design of Takagi-Sugeno fuzzy systems. Em "Procs. of the IEEE International Conference on Evolutionary Computation", tomo 2, páginas 1384–1389, Hawaii (julho 2002).
- [120] A. P. DEMPSTER, N. M. LAIRD E D. B. RUBIN. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B* **39**, 1–38 (1977).
- [121] T. G. DIETTERICH. Machine learning research: Four current directions. *AI Magazine* **18**(4), 97–136 (1997).
- [122] T. G. DIETTERICH. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation* **10**(7), 1895–1924 (1998).
- [123] T. G. DIETTERICH. Ensemble methods in machine learning. Em J. KITTLER E F. ROLI, editores, "Procs. of International Workshop on Multi-classifier Systems", tomo 1857 de "LNCS", páginas 1–15. Springer (2000).
- [124] T. G. DIETTERICH. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning* **40**(2), 139–158 (2000).
- [125] J. DORAN, S. FRANKLIN, N. JENNINGS E T. NORMAN. On cooperation in multi-agent systems. *The Knowledge Engineering Review* **12**(3), 309–314 (1997).
- [126] M. DORIGO, V. MANIEZZO E A. COLONI. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics* **26**(1), 29–41 (1996).

- [127] A. E. DOUGLAS. "Symbiotic Interactions". Oxford University Press (1994).
- [128] H. DRUCKER. Boosting using neural networks. Em Sharkey [426], páginas 51–77.
- [129] H. DRUCKER, C. CORTES, L. D. JACKEL, Y. LECUN E V. VAPNIK. Boosting and other ensemble methods. *Neural Computation* 6, 1289–1301 (1994).
- [130] R. O. DUDA, P. E. HART E D. G. STORK. "Pattern Classification". John Wiley (2001).
- [131] E. DURFEE E T. A. MONTGOMERY. Coordination as distributed search in a hierarchical behavior space. *IEEE Transactions on Systems, Man, and Cybernetics* 21(6), 1363–1378 (novembro 1991).
- [132] E. H. DURFEE. The distributed artificial intelligence melting pot. *IEEE Transactions on Systems, Man, and Cybernetics* 21(6), 1301–1306 (1991).
- [133] E. H. DURFEE E J. S. ROSENSCHEIN. Distributed problem solving and multi-agent systems: Comparison and examples. Em "Procs. of International Workshop on Distributed Artificial Intelligence", páginas 94–104 (1994).
- [134] D. E. ECKHARDT E L. D. LEE. A theoretical basis for the analysis of multiversion software subject to coincident failures. *IEEE Transactions on Software Engineering* 11(12), 1511–1517 (1985).
- [135] P. R. EHRLICH E P. H. RAVEN. Butterflies and plants: A study in coevolution. *Ecology* 18, 586–608 (1964).
- [136] A. E. EIBEN, R. HINTERDING E Z. MICHALEWICZ. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 3(2), 124–141 (1999).
- [137] A. E. EIBEN E M. SCHOENAUER. Evolutionary computing. *Information Processing Letters* 82, 1–6 (2002).
- [138] J. L. ELMAN, E. A. BATES, M. H. JOHNSON, A. K. SMITH, D. PARISI E K. PLUNKETT. "Rethinking Innateness—A Connectionist Perspective on Development". MIT Press (1996).
- [139] D. ENSLEY E D. NELSON. Extrapolation of Mackey-Glass data using cascade correlation. *Simulation* 58(3), 333–339 (1992).
- [140] R. ERIKSSON. Applying cooperative coevolution to inventory control parameter optimization. Dissertação de mestrado, University of Skövde (1996).
- [141] C. FANCOURT E J. PRINCIPE. Modeling time dependencies in the mixture of experts. Em "Procs. of International Joint Conference on Neural Networks", tomo 1, páginas 2324–2327 (1998).
- [142] J. FANG E Y. XI. Neural network design based on evolutionary programming. *Artificial Intelligence in Engineering* 11, 155–161 (1997).
- [143] L. FAUSETT. "Fundamentals of Neural Networks". Prentice Hall International (1994).
- [144] M. E. FAYAD, D. C. SCHMIDT E R. E. JOHNSON, editores. "Building Application Frameworks". Wiley (1999).
- [145] M. FENSTER, S. KRAUS E J. ROSENSCHEIN. Coordination without communication: Experimental validation of focal point techniques. Em Huhns e Singh [214], páginas 380–386.
- [146] J. FERBER. Reactive distributed artificial intelligence. Em O'Hare e Jennings [350], páginas 287–317.
- [147] S. G. FICICI E J. B. POLLACK. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. Em "Procs. of Artificial Life VI", páginas 238–247. MIT Press (1998).
- [148] M. FIGUEIREDO E F. GOMIDE. Design of fuzzy systems using neurofuzzy networks. *IEEE Transactions on Neural Networks* 10(4), 815–827 (julho 1999).

- [149] T. FININ E R. FRITZSON. KQML—A language and protocol for knowledge and information exchange. Em “*Procs. of International Workshop on Distributed Artificial Intelligence*”, páginas 126–136 (1994).
- [150] J. FODOR. “*The Modularity of Mind: An Essay on Faculty Psychology*”. MIT Press, Cambridge, MA (1983).
- [151] D. FOGEL, E. WASSON III E E. BOUGHTON. Evolving neural networks for detecting breast cancer. *Cancer Letters* **96**, 49–53 (1995).
- [152] D. B. FOGEL. “*Evolutionary Computation*”. IEEE Press, Piscataway, NJ (1995).
- [153] D. B. FOGEL E C. J. ROBINSON, editores. “*Computational Intelligence: The Experts Speak*”. IEEE Press (2003).
- [154] S. FORREST. Emergent computation: Self-organizing, collective, and cooperative phenomena in natural and artificial computing networks. *Physica D* **42**, 1–11 (1990).
- [155] S. FORREST E J. MILLER. Emergent behaviors of classifier systems. *Physica D* **42**, 213–227 (1990).
- [156] M. S. FOX. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics* **11**(1), 70–80 (1981).
- [157] S. FRANKLIN. Coordination without communication. <http://www.msci.memphis.edu/~franklin/coord.html> (1996).
- [158] S. FRANKLIN E A. GRAESSER. Is it an agent, or just a program? A taxonomy for autonomous agentes. Em “*Intelligent Agents III—Procs. of the Third International Workshop on Agent Theories, and Languages*”, tomo 1193 de “*LNAI*”, páginas 21–35. Springer (1996).
- [159] A. A. FREITAS. “*Data Mining and Knowledge Discovery with Evolutionary Algorithms*”. Springer (2002).
- [160] Y. FREUND E R. E. SCHAPIRE. Experiments with a new boosting algorithm. Em “*Procs. of International Conference on Machine Learning*”, páginas 148–156 (1996).
- [161] Y. FREUND E R. E. SCHAPIRE. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55**(1), 119–139 (1997).
- [162] C. FRIEDRICH. Ensembles of evolutionary created artificial neural networks and nearest neighbour classifiers. Em “*Procs. of Third Online Conference on Soft Computing in Engineering Design and Manufacturing*”, páginas 288–298 (1998).
- [163] J. FRITSCH, M. FINKE E A. WAIBEL. Adaptively growing hierarchical mixtures of experts. Em M. I. JORDAN, M. C. MOZER E T. PETSCHKE, editores, “*Advances in Neural Information Processing Systems (NIPS)*”, tomo 9, páginas 459–465. MIT Press (1997).
- [164] R. FROZZA E L. O. ÁLVARES. Criteria for the analysis of coordination in multi-agent applications. Em F. ARBAB E C. TALCOTT, editores, “*Procs. of 5th International Conference on Coordination Models and Languages*”, tomo 2315 de “*LNCS*”, páginas 158–165. Springer (2002).
- [165] K. FUNAHASHI. “*Introduction to Statistical Pattern Recognition*”. Academic Press (1990).
- [166] D. FUTUYMA E M. SLATKIN, editores. “*Coevolution*”. Sinauer Associates (1983).
- [167] N. GARCÍA-PEDRAJAS, C. HERVÁS-MARTÍNEZ E J. MUÑOZ-PÉREZ. Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks). *Neural Networks* **15**(10), 1255–1274 (2002).
- [168] N. GARCÍA-PEDRAJAS, C. HERVÁS-MARTÍNEZ E J. MUÑOZ-PÉREZ. COVNET: A cooperative coevolutionary model for evolving artificial neural networks. *IEEE Transactions on Neural Networks* **14**(3), 575–596 (2003).

- [169] A. C. B. GARCIA E J. S. SICHMAN. Agentes e sistemas multiagentes. Em Rezende [389], páginas 269–306.
- [170] V. J. GARCIA. Algoritmos meméticos paralelos aplicados a problemas de otimização combinatória. Dissertação de mestrado, Faculdade de Engenharia Elétrica e de Computação — Unicamp (2002).
- [171] S. GEMAN, E. BIENENSTOCK E T. DOURSAT. Neural networks and the bias/variance dilemma. *Neural Computation* **5**, 1–58 (1992).
- [172] M. R. GENESERETH E S. P. KETCHPEL. Software agents. *Communications of ACM* **37**(7), 48–53, 147 (julho 1994).
- [173] J. GHOSH. Neural-symbolic hybrid systems. Em M. PADGET, N. KARAYIANNIS E L. ZADEH, editores, “The Handbook of Applied Computational Intelligence”. CRC Press (2001).
- [174] G. GIACINTO E F. ROLI. Dynamic classifier selection. Em J. KITTLER E F. ROLI, editores, “Procs. of International Workshop on Multi-classifier Systems”, tomo 1857 de “LNCS”, páginas 177–189. Springer (2000).
- [175] J. GIARRATANO E R. RILEY. “Expert Systems: Principles and Programming”. PWS Publishing Company (1994).
- [176] L. E. GILBERT E P. H. RAVEN, editores. “Co-evolution of Animals and Plants”. University of Texas Press (1975).
- [177] C. GILDE. Time series analysis and prediction using recurrent gated experts. Dissertação de mestrado, Department of Computer Science—University of Skövde (1996).
- [178] D. E. GOLDBERG. “Genetic Algorithms in Search, Optimization and Machine Learning”. Addison-Wesley (1989).
- [179] C. V. GOLDMAN E J. S. ROSENSCHEIN. Evolutionary patterns of agent organizations. *IEEE Transactions on Systems, Man, and Cybernetics, Part A* **32**(1), 135–148 (janeiro 2002).
- [180] S. GOONATILAKE E S. KHEBBAL, editores. “Intelligent Hybrid Systems”. John Wiley & Sons (1995).
- [181] C. W. J. GRANGER. Combining forecasts—twenty years later. *Journal of Forecasting* **8**(3), 167–173 (1989).
- [182] G. GREENWOOD. Training partially recurrent neural networks using evolutionary strategies. *IEEE Transactions on Speech & Audio Processing* **5**(2), 192–194 (1997).
- [183] J. J. GREFENSTETTE. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics* **16**(1), 122–128 (1986).
- [184] F. GRUAU. Automatic definition of modular neural networks. *Adaptive Behaviour* **3**(2), 151–183 (1995).
- [185] F. GRUAU E D. WHITLEY. Adding learning to the cellular development of neural networks: Evolution and the baldwin effect. *Evolutionary Computation* **1**(3), 213–233 (1993).
- [186] S. GUNN. Support vector machine for classification and regression. Relatório técnico ISIS-1-98, Image, Speech & Intelligent Systems Group, University of Southampton (1998).
- [187] S. GUTTA, J. R. J. HUANG, P. JONATHON E H. WECHSLER. Mixture of experts for classification of gender, ethnic origin, and pose of human faces. *IEEE Transactions on Neural Networks* **11**(4), 948–960 (2000).
- [188] I. GUYON, J. WESTON, S. BARNHILL E V. VAPNIK. Gene selection for cancer classification using support vector machines. *Machine Learning* **46**(1–3), 389–422 (2002).

- [189] J. B. HAMSHIRE E A. WAIBEL. The Meta-Pi network: Building distributed knowledge representations for robust multisource pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14(2), 751–769 (1992).
- [190] P. HANCOCK. An empirical comparison of selection methods in evolutionary algorithms. Em “Procs. of the Third Parallel Problem Solving from Nature Conference”, tomo 866 de “LNCS”, páginas 80–94. Springer (1994).
- [191] L. K. HANSEN E P. SALAMON. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(10), 993–1001 (1990).
- [192] B. L. M. HAPPEL E J. M. J. MURRE. The design and evolution of modular neural network architectures. *Neural Networks* 7, 985–1004 (1994).
- [193] A. HARA E T. NAGAO. Emergence of cooperative behavior using ADG; Automatically Defined Groups. Em “Procs. of Genetic and Evolutionary Computation Conference (GECCO)”, páginas 1039–1046 (1999).
- [194] P. G. HARRALD E M. KAMSTRA. Evolving artificial neural networks to combine financial forecasts. *IEEE Transactions on Evolutionary Computation* 1(1), 40–52 (1997).
- [195] W. HART E R. BELEW. Optimization with genetic algorithm hybrids that use local search. Em Belew e Mitchell [48], páginas 483–496.
- [196] S. HASHEM. Optimal linear combinations of neural networks. *Neural Networks* 10(4), 599–614 (agosto 1997).
- [197] T. HASTIE, R. TIBSHIRANI E J. FRIEDMAN. “The Elements of Statistical Learning”. Springer-Verlag (2001).
- [198] S. HAYKIN. “Neural Networks—A Comprehensive Foundation”. Prentice Hall, segunda edição (1999).
- [199] T. HAYNES E S. SEN. Crossover operators for evolving a team. Em “Genetic Programming 1997: Procs. of the Second Annual Conference”, páginas 162–167. Morgan Kaufmann (1997).
- [200] T. HAYNES, S. SEN, D. SCHOENEFELD E R. WAINWRIGHT. Evolving a team. Em E. V. SIEGEL E J. R. KOZA, editores, “Working Notes for the AAAI Symposium on Genetic Programming”, Cambridge, MA (1995). AAAI.
- [201] M. A. HEARST. Support vector machines. *IEEE Intelligent Systems* 13(4), 18–28 (julho 1998).
- [202] L. M. HERCOG E T. C. FOGARTY. Co-evolutionary classifier systems for multi-agent simulation. Em “Procs. of IEEE Congress on Evolutionary Computation”, páginas 1798–1803 (2002).
- [203] W. D. HILLIS. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D* 42, 228–234 (1990).
- [204] G. E. HINTON E S. J. NOWLAN. How learning can guide evolution. *Complex Systems* 1, 495–502 (1987).
- [205] T. K. HO, J. J. HULL E S. N. SRIHARI. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(1), 66–75 (janeiro 1994).
- [206] Y.-C. HO E K.-C. CHU. Team decision theory and information structures in optimal control problems—Part I. *IEEE Transactions on Automatic Control* 17(1), 15–22 (1972).
- [207] J. H. HOLLAND. “Adaptation in Natural and Artificial Systems”. MIT Press, segunda edição (1992).
- [208] S.-G. HONG, S.-K. OH, M.-S. KIM E J.-J. LEE. Evolving mixture of experts for nonlinear time series modelling and prediction. *Electronics Letters* 38, 34–35 (janeiro 2002).
- [209] K. HORNIK. Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4, 251–257 (1991).

- [210] K. HORNIK, M. STINCHCOMBE E H. WHITE. Multilayer feedforward networks are universal approximators. *Neural Networks* **2**, 359–366 (1989).
- [211] C.-W. HSU E C.J. LIN. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks* **13**(2), 415–425 (2002).
- [212] Y. S. HUANG E C. Y. SUEN. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**(1), 90–94 (janeiro 1995).
- [213] R. HUBER E H. A. MAYER. ERC—Evolutionary resample and combine for adaptive parallel training data set selection. Em “*Procs. of 14th International Conference on Pattern Recognition*”, páginas 882–885. IEEE Computer Society (agosto 1998).
- [214] M. N. HUHNS E M. P. SINGH, editores. “*Readings in Agents*”. Morgan Kaufmann Publishers (1998).
- [215] M. N. HUHNS E L. M. STEPHENS. Multiagent systems and societies of agents. Em Weiß [475], páginas 79–115.
- [216] P. HUSBANDS. Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. Em “*Evolutionary Computing, AISB Workshop*”, tomo 865 de “*LNCS*”, páginas 150–165 (1994).
- [217] P. HUSBANDS E F. MILL. Simulated co-evolution as the mechanism for emergent planning and scheduling. Em “*Procs. of the Fourth International Conference on Genetic Algorithms*”, páginas 264–270 (1991).
- [218] H. IBA. Emergent cooperation for multiple agents using genetic programming. Em “*Parallel Problem Solving from Nature IV*”, páginas 32–41. Springer (1996).
- [219] T. IKEGAMI E K. KANEKO. Computer symbiosis—Emergence of symbiotic behavior through evolution. Em S. FORREST, editor, “*Emergent Computation: Self-Organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks*”, páginas 235–243. MIT Press (1991).
- [220] A. IORIO E X. LI. Parameter control within a co-operative co-evolutionary genetic algorithm. Em “*Parallel Problem Solving from Nature - PPSN VII*”, tomo 2439 de “*LNCS*”, páginas 247–256. Springer (2002).
- [221] E. IYODA E F. VON ZUBEN. Evolutionary hybrid composition of activation functions in feedforward neural networks. Em “*Procs. of INNS-IEEE International Joint Conference on Neural Networks*”, tomo 6, páginas 4048–4053 (1999).
- [222] R. A. JACOBS. “*Task Decomposition through Competition in a Modular Connectionist Architecture*”. Tese de doutorado, University of Massachusetts, Amherst (1990).
- [223] R. A. JACOBS. Bias/variance analyzes of mixtures-of-experts architectures. *Neural Computation* **9**, 369–383 (1997).
- [224] R. A. JACOBS. Computational studies of the development of functionally specialized neural modules. *Trends in Cognitive Science* **3**(2), 31–38 (1999).
- [225] R. A. JACOBS E M. I. JORDAN. Learning piecewise control strategies in a modular neural network architecture. *IEEE Transactions on Systems, Man and Cybernetics* **23**(2), 337–345 (1993).
- [226] R. A. JACOBS, M. I. JORDAN, S. J. NOWLAN E G. E. HINTON. Adaptive mixtures of local experts. *Neural Computation* **3**, 79–87 (1991).
- [227] A. K. JAIN, R. P. W. DUIN E J. MAO. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(1), 4–37 (2000).

- [228] J.-S. R. JANG. ANFIS: Adaptive network-based fuzzy inference systems. *IEEE Transactions on Systems, Man, and Cybernetics* **23**(6), 665–685 (1993).
- [229] J.-S. R. JANG E C. SUN. Neuro-fuzzy modeling and control. *Proceedings of IEEE* **83**(3), 378–406 (1995).
- [230] D. H. JANZEN. When is it coevolution? *Evolution* **34**, 611–612 (1980).
- [231] N. R. JENNINGS. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review* **8**(3), 223–250 (1993).
- [232] N. R. JENNINGS. Coordination techniques for distributed artificial intelligence. Em O'Hare e Jennings [350], páginas 187–210.
- [233] C. JI E S. MA. Combination of weak classifiers. *IEEE Transactions of Neural Networks* **8**(1), 32–42 (1997).
- [234] D. A. JIMÉNEZ E N. WALSH. Dynamically weighted ensemble of neural networks for classification. Em “Procs. of International Joint Conference on Neural Networks”, páginas 753–756, Alaska (1998).
- [235] T. JOACHIMS. “Learning to Classify Text using Support Vector Machines”. Kluwer (2002).
- [236] S. JOHNSON. “Emergence—The Connected Lives of Ants, Brains, Cities and Software”. Scribner, New York (2001).
- [237] K. A. DE JONG. “An Analysis of the Behavior of a Class of Genetic Adaptive Systems”. Tese de doutorado, University of Michigan (1975).
- [238] K. DE JONG E M. A. POTTER. Evolving complex structures via cooperative coevolution. Em “Procs. of the Fourth Annual Conference on Evolutionary Programming”, páginas 307–317, Cambridge, MA (1995). MIT Press.
- [239] M. I. JORDAN E R. A. JACOBS. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* **6**, 181–214 (1994).
- [240] M. I. JORDAN E L. XU. Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks* **8**(9), 1409–1431 (1995).
- [241] C. SILVA JR. E S. SASSON. “Biologia”, tomo 3. Atual Editora (1990).
- [242] C. KARR. Genetic algorithms for fuzzy controllers. *AI Experts* **6**, 26–33 (1991).
- [243] S. V. KARTALOPOULOS. “Understanding Neural Networks and Fuzzy Logic”. IEEE Press (1996).
- [244] N. KASABOV, J. KIM, M. WATTS E A. GRAY. A fuzzy neural network architecture for adaptive learning and knowledge acquisition. *Information Sciences* **101**(3/4), 155–175 (1996).
- [245] S. A. KAUFFMAN. “The Origins of Order: Self-organization and Selection in Evolution”. Oxford University Press (1993).
- [246] M. KEARNS, Y. MANSOUR, A. NG E D. RON. An experimental and theoretical comparison of model selection methods. *Machine Learning* **27**(1), 7–50 (1997).
- [247] M. KEARNS E L. VALIANT. Learning Boolean formulae or finite automata is as hard as factoring. Relatório técnico, Harvard University Aiken Computation Laboratory (1988).
- [248] N. KEERATIVUTTITUMRONG, N. CHAYARATANA E V. VARAVITHYA. Multi-objective co-operative co-evolutionary genetic algorithm. Em “Parallel Problem Solving from Nature - PPSN VII”, tomo 2439 de “LNCS”, páginas 288–297. Springer (2002).

- [249] J. KENNEDY, R. EBERHART E Y. SHI, editores. "Swarm Intelligence". Morgan Kaufmann Publishers (2001).
- [250] R. KHOSLA E T. DILLON, editores. "Engineering Intelligent Hybrid Multi-agent Systems". Kluwer Academic Publisher (1997).
- [251] H.-C. KIM, S. PANG, H.-M. JE, D. KIM E S.-Y. BANG. Support vector machine ensemble with bagging. Em "Procs. of SVM 2002", tomo 2388 de "LNCS", páginas 397–408. Springer (2002).
- [252] H. KITANO. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems* 4(4), 461–476 (1990).
- [253] J. KITTLER, M. HATEF, R. P. W. DUIN E J. MATAS. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(3), 226–239 (1998).
- [254] R. M. KLEINBERG. Stochastic discrimination. *Annals of Mathematics and Artificial Intelligence* 1, 207–239 (1990).
- [255] R. KOHAVI E F. PROVOST. Glossary of terms (Special issue on applications of machine learning and the knowledge discovery process). *Machine Learning* 30, 271–274 (1998).
- [256] A. KROGH E J. VEDELSBY. Neural network ensembles, cross validation, and active learning. Em G. TESAURO, D. TOURETZKY E T. LEEN, editores, "Advances in Neural Information Processing Systems", tomo 7, páginas 231–238. MIT Press, Cambridge, MA (1995).
- [257] K. W. C. KU, M. W. MAK E W.-C. SIU. Adding learning to cellular genetic algorithms for training recurrent neural networks. *IEEE Transactions on Neural Networks* 10(2), 239–252 (1999).
- [258] K. W. C. KU, M. W. MAK E W.-C. SIU. A study of the Lamarckian evolution of recurrent neural networks. *IEEE Transactions on Evolutionary Computation* 4(1), 31–42 (2000).
- [259] L. I. KUNCHEVA. Genetic algorithm for feature selection for parallel classifiers. *Information Processing Letters* 46, 163–168 (1993).
- [260] L. I. KUNCHEVA. An application of OWA operators to the aggregation of multiple classifiers. Em Yager e Kacprzyk [498], páginas 330–343.
- [261] L. I. KUNCHEVA. Clustering-and-selection model for classifier combination. Em "Procs. of Knowledge-Based Intelligent Engineering Systems and Allied Technologies (KES'2000)", páginas 185–188, Brighton, UK (2000).
- [262] L. I. KUNCHEVA. Combining classifiers: Soft computing solutions. Em S. K. PAL E A. PAL, editores, "Pattern Recognition: From Classical to Modern Approaches", páginas 427–451. World Scientific (2001).
- [263] L. I. KUNCHEVA. Using measures of similarity and inclusion for multiple classifier fusion by decision templates. *Fuzzy Sets and Systems* 122(3), 401–407 (2001).
- [264] L. I. KUNCHEVA. Switching between selection and fusion in combining classifiers: An experiment. *IEEE Transactions On Systems, Man, and Cybernetics, Part B* 32(2), 146–156 (2002).
- [265] L. I. KUNCHEVA, J. C. BEZDEK E R. P. W. DUIN. Decision templates for multiple classifier fusion: An experimental comparison. *Pattern Recognition* 34(2), 299–314 (2001).
- [266] L. I. KUNCHEVA E L. C. JAIN. Designing classifier fusion systems by genetic algorithms. *IEEE Transactions on Evolutionary Computation* 4(4), 327–336 (novembro 2000).
- [267] L. I. KUNCHEVA E C.J. WHITAKER. Measures of diversity in classifier ensembles. *Machine Learning* 51, 181–207 (2003).

- [268] S.-Y. KUNG E J. TAUR. Decision-based neural networks with signal/image classification applications. *IEEE Transactions on Neural Networks* 6, 170–181 (1995).
- [269] S.-Y. KUNG, J. TAUR E S.-H. LIN. Synergistic modeling and applications of hierarchical fuzzy neural networks. *Proceedings of IEEE* 87(9), 1550–1573 (1999).
- [270] N. KWAK E C.-H. CHOI. Input feature selection for classification problems. *IEEE Transactions on Neural Networks* 13(1), 143–159 (janeiro 2002).
- [271] T.-Y. KWOK E D.-Y. YEUNG. Constructive algorithms for structure learning in feedforward neural networks for regression problems. *IEEE Transactions on Neural Networks* 8(3), 630–645 (maio 1997).
- [272] J. E. LAIRD, R. M. JONES E P. E. NIELSEN. Knowledge-based multiagent coordination. *Presence—Teleoperators and Virtual Environments* 7(6), 547–563 (1998).
- [273] L. LAM E C. Y SUEN. Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics* 27(5), 553–568 (1997).
- [274] J.-B. LAMARCK. “Système des animaux sans vertèbres”. Paris (1809).
- [275] P. LANGLEY, editor. “Elements of Machine Learning”. Morgan Kaufmann (1996).
- [276] C. G. LANGTON. “Artificial Life: An Overview”. MIT Press, Cambridge:MA (1995).
- [277] P. LANZI, W. STOLZMANN E S. W. WILSON, editores. “Learning Classifier Systems: From Foundations to Applications”, tomo 1813 de “LNCS”. Springer (2000).
- [278] O. LARSEN, A. A. FREITAS E J.C. NIEVOLA. Constructing X-of-N attributes with a genetic algorithm. Em A LOFTI, J. GARIBALDI E R. JOHN, editores, “Procs. of 4th Int. Conf. on Recent Advances in Soft Computing (RASC-2002)”, páginas 326–331. Nottingham Trent University (dezembro 2002).
- [279] R. LEARDI. Genetic algorithms in feature selection. Em J. DEVILLERS, editor, “Genetic Algorithms in Molecular Modeling”, páginas 67–86. Academic, London (1996).
- [280] K. LEE, Y. CHUNG E H. BYUN. SVM-based face verification with feature set of small size. *Electronics Letters* 38(15), 787–789 (2002).
- [281] S.-I. LEE, J.-H. AHN E S.-B. CHO. Exploiting diversity of neural ensembles with speciated evolution. Em “Procs. of International Joint Conference on Neural Networks”, páginas 808–813, Washington, D.C. (2001). IEEE Press.
- [282] F.H.F. LEUNG, H. K. LAM, S. H. LING E P. K. S. TAM. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks* 14(1), 79–88 (2003).
- [283] C. P. LIM E R. F. HARRISON. Online pattern classification with multiple neural networks systems: An experimental study. *IEEE Transactions on Systems, Man, and Cybernetics — Part C* 33(2), 235–247 (2003).
- [284] C. A. M. LIMA, A. L. V. COELHO E F. J. VON ZUBEN. Ensembles of support vector machines for regression problems. Em “Procs. of the International Joint Conference on Neural Networks (IJCNN)”, páginas 2381–2386, Hawaii (2002). IEEE Press.
- [285] C. A. M. LIMA, A. L. V. COELHO E F. J. VON ZUBEN. Fuzzy systems design via ensembles of ANFIS. Em “Procs. of IEEE International Conference on Fuzzy Systems (Fuzz-IEEE)”, páginas 506–511, Hawaii, USA (2002). IEEE Press.

- [286] C. A. M. LIMA, A. L. V. COELHO E F. J. VON ZUBEN. Mixture of experts applied to nonlinear dynamic systems identification: A comparative study. Em "Procs. of VII Brazilian Symposium on Neural Networks (SBRN)", páginas 153–158, Recife, Brazil (2002). IEEE Computer Society Press.
- [287] C. A. M. LIMA, A. L. V. COELHO E F. J. VON ZUBEN. Model selection based on VC-dimension for heterogeneous ensembles of support vector machines. Em A. LOTFI, J. GARIBALDI E R. JOHN, editores, "Procs. of 4th International Conference on Recent Advances on Soft Computing (RASC)", páginas 459–464, Nottingham (dezembro 2002). The Nottingham Trent University.
- [288] C. A. M. LIMA, A. L. V. COELHO E F. J. VON ZUBEN. On localized mixture of support vector machine experts. Em A. LOTFI, J. GARIBALDI E R. JOHN, editores, "Procs. of 4th International Conference on Recent Advances on Soft Computing (RASC)", páginas 465–470, Nottingham (dezembro 2002). The Nottingham Trent University.
- [289] C. A. M. LIMA, A. L. V. COELHO E F. J. VON ZUBEN. Ensembles of support vector machines for classification tasks with reduced training sets. *WSEAS Transactions on Systems* 2(2), 370–375 (abril 2003).
- [290] C. A. M. LIMA, A. L. V. COELHO E F. J. VON ZUBEN. A multiagent-based constructive approach for feedforward neural networks. Em M. R. BERTHOLD, H.-J. LENZ, E. BRADLEY, R. KRUSE E C. BORGELT, editores, "Advances in Intelligent Data Analysis V", tomo 2810 de "LNCS", páginas 462–473. Springer-Verlag (2003).
- [291] C. A. M. LIMA, A. L. V. COELHO E F. J. VON ZUBEN. Alleviating the kernel function choice problem with heterogeneous ensembles of support vector machines. Em preparação. (2004).
- [292] C. A. M. LIMA, A. L. V. COELHO E F. J. VON ZUBEN. Embedding support vector machines into localised mixtures of experts. Em A. LOTFI E J. M. GARIBALDI, editores, "Applications and Science in Soft Computing", Advances in Soft Computing Series, páginas 155–162. Springer-Verlag (2004).
- [293] C.-T. LIN E C. S. GEORGE LEE. "Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems". Prentice Hall (1996).
- [294] R. LIPPMANN. Neural networks, Bayesian *a posteriori* probabilities, and pattern classification. Em Cherkassky et al. [77], páginas 83–104.
- [295] Y. LIU E X. YAO. Evolutionary design of artificial neural networks with different nodes. Em "Procs. of the 1996 IEEE International Conference on Evolutionary Computation", páginas 670–675 (1996).
- [296] Y. LIU E X. YAO. Towards designing neural network ensembles by evolution. Em "Parallel Problem Solving from Nature V", tomo 1498 de "LNCS", páginas 623–632. Springer (1998).
- [297] Y. LIU E X. YAO. Ensemble learning via negative correlation. *Neural Networks* 12(10), 1399–1404 (1999).
- [298] Y. LIU, X. YAO E T. HIGUCHI. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation* 4(4), 380–387 (novembro 2000).
- [299] Y. LIU, X. YAO, Q. ZHAO E T. HIGUCHI. Evolving a cooperative population of neural networks by minimizing mutual information. Em "Procs. of IEEE International Conference on Evolutionary Computation", páginas 384–389, Seoul, Korea (2001).
- [300] S. LUKE, C. HOHN, J. FARRIS, G. JACKSON E J. HENDLER. Co-evolving soccer softbot team coordination with genetic programming. Em "Procs. of the First International Workshop on RoboCup", páginas 115–118, Nagoya (1997).

- [301] S. LUKE E L. SPECTOR. Evolving teamwork and coordination with genetic programming. Em J.R. KOZA ET AL., editor, "Proceedings of the First Annual Conference on Genetic Programming", páginas 150–156. MIT Press (1996).
- [302] R. MACLIN E D. OPITZ. An empirical evaluation of Bagging and Boosting. Em "Procs. of Fourteenth National Conference on Artificial Intelligence (AAAI)", páginas 546–551 (1997).
- [303] S. W. MAHFOUD. Niching methods for genetic algorithms. Relatório técnico IlliGAL Report 95001, University of Illinois at Urbana-Champaign (1995).
- [304] M. MANGEAS, A. S. WEIGEND E C. MULLER. Forecasting electricity demand using nonlinear mixture of experts. Em "Procs. of World Congress on Neural Networks", tomo II, páginas 48–53 (1995).
- [305] V. MANIEZZO. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks* 5(1), 39–53 (janeiro 1994).
- [306] J. MAO. A case study on Bagging, Boosting and basic ensembles of neural networks for OCR. Em "Procs. of IEEE International Joint Conference on Neural Networks", tomo 3, páginas 1828–1833 (1998).
- [307] L. MARGULIS E R. FESTER, editores. "Symbiosis as a Source of Evolutionary Innovation". The MIT Press (1991).
- [308] N. MATOS, C. SIERRA E N. R. JENNINGS. Determining successful negotiation strategies: An evolutionary approach. Em "Procs. of International Conference on Multiagent Systems", páginas 182–189, Paris (1998).
- [309] H. MATSUBARA, I. NODA E K. HIRAKI. Learning of cooperative actions in multi-agent systems: A case study of pass play in soccer. Em "Adaptation, Coevolution and Learning in Multiagent Systems: 1996 AAAI Spring Symposium", páginas 63–67. AAAI Press (1996).
- [310] H. A. MAYER. Symbiotic coevolution of artificial neural networks and training data sets. Em "Parallel Problem Solving from Nature — PPSN V", tomo 1498 de "LNCS", páginas 511–520. Springer (1998).
- [311] H. A. MAYER E R. SCHWAIGER. Differentiation of neuron types by evolving activation function templates for artificial neural networks. Em "Procs. of International Joint Conference on Neural Networks", páginas 1773–1778. IEEE Press (maio 2002).
- [312] E. MAYR. "Systematics and the Origin of Species". Columbia University Press (1942).
- [313] J. MCCARTHY E P. J. HAYES. Some philosophical problems from the standpoint of artificial intelligence. Em "Machine Intelligence", tomo 4. Edimburgh University Press, Edimburgh (1969).
- [314] M. MCCLOSKEY E N. COHEN. Catastrophic interference in connectionist networks: The sequential learning problem. Em G. H. BOWER, editor, "The Psychology of Learning and Motivation", tomo 24, páginas 109–164. Academic Press (1989).
- [315] J. D. McDONNELL E D. E. WAAGEN. Evolving recurrent perceptrons for time-series modeling. *IEEE Transactions on Neural Networks* 5(1), 24–38 (janeiro 1994).
- [316] G. J. McLACHLAN E D. PEEL. "Finite Mixture Models". Wiley (2000).
- [317] L. A. MEDSKAR. "Hybrid Intelligent Systems". Kluwer Academic Publisher (1995).
- [318] K. MEHROTRA, C. K. MOHAN E S. RANKA. "Elements of Artificial Neural Networks". MIT Press (1997).
- [319] J. W. L. MERRILL E R. F. PORT. Fractally configured neural networks. *Neural Networks* 4(1), 53–60 (1991).

- [320] Z. MICHALEWICZ. "Genetic Algorithms + Data Structures = Evolution Programs". Springer-Verlag (1996).
- [321] D. MICHIE, D. J. SPIEGELHALTER E C. C. TAYLOR. "Machine Learning, Neural, and Statistical Classification". Ellis Horwood Limited, London (1994).
- [322] G. F. MILLER, P. M. TODD E S. U. HEDGE. Designing neural networks using genetic algorithms. Em "Procs. of Third International Conference on Genetic Algorithms", páginas 379–384 (1989).
- [323] J. H. MILLER. The coevolution of automata in the repeated prisoners' dilemma. *Journal of Economic Behavior and Organization* 29, 87–112 (1996).
- [324] M. MINSKY. "The Society of Mind". Simon and Schuster (1986).
- [325] M. MITCHELL, J. P. CRUTCHFIELD E P. T. HRABER. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D* 75, 361–391 (1994).
- [326] T. M. MITCHELL. "Machine Learning". McGraw-Hill (1997).
- [327] K. MÜLLER, A. SMOLA, G. TATSH, B. SCHÖLKOPF, J. KOHLMORGEN E V. VAPNIK. Predicting time series with support vector machines. Em "Procs. of International Conference on Artificial Neural Networks", tomo 1327 de "LNCS", páginas 999–1004 (1997).
- [328] P. MOERLAND. Some methods for training mixtures of experts. Relatório técnico 97-05, IDIAP (1997).
- [329] P. MOERLAND. Classification using localized mixtures of experts. Em "Proceedings of the International Conference on Artificial Neural Networks (ICANN'99)", tomo 2, páginas 838–843 (1999).
- [330] M. C. MONARD E J. A. BARANAUSKAS. Conceitos sobre aprendizado de máquina. Em Rezende [389], páginas 89–114.
- [331] D. J. MONTANA E L. DAVIS. Training feedforward neural networks using genetic algorithms. Em "Procs. of 11th International Joint Conference on Artificial Intelligence", páginas 762–767 (1989).
- [332] C. MORAGA E E. MEYER. Knowledge-based genetic algorithms with fuzzy fitness. Em "Procs. of Third International Conference on Fuzzy Logic, Neural Nets, and Soft Computing", páginas 473–474, Japan (1994).
- [333] P. A. MORETTIN E C. M. TOLOI. "Séries Temporais". Métodos Quantitativos. Atual Editora (1986).
- [334] D. MORIARTY E R. MIIKKULAINEN. Efficient reinforcement learning through symbiotic evolution. *Machine Learning* 22, 11–33 (1996).
- [335] D. E. MORIARTY E R. MIIKKULAINEN. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation* 5(4), 373–399 (1997).
- [336] D. E. MORIARTY E R. MIIKKULAINEN. Hierarchical evolution of neural networks. Em "Procs. of the IEEE International Conference on Evolutionary Computation", páginas 428–433, Anchorage, USA (1998).
- [337] J. MORRISON. Co-evolution and genetic algorithms. Dissertação de mestrado, Carleton Institute of Computer Science (1998).
- [338] J. MORRISON E F. OPPACHER. A general model of co-evolution for genetic algorithms. Em A. DOBNIKAR, N. STEELE, D. PEARSON E R. ALBRECHT, editores, "International Conference on Artificial Neural Networks and Genetic Algorithms", páginas 262–268. Springer-verlag (1999).
- [339] P. MOSCATO. On evolution, search, optimization, genetic algorithms and martial arts. Relatório técnico C3P-826, Caltech Concurrent Program (1989).

- [340] P. MOSCATO. "Problemas de otimização NP, aproximabilidade e computação evolutiva: Da prática à teoria". Tese de doutorado, Faculdade de Engenharia Elétrica e de Computação (2001).
- [341] H. MÜHLENBEIN E D. SCHLIERKAMP-VOOSEN. Predictive models for the Breeder genetic algorithm. *Evolutionary Computation* 1(1), 25–49 (1993).
- [342] J. M. J. MURRE. "Learning and Categorization in Modular Neural Networks". Erlbaum (1992).
- [343] U. NAFTALY, N. INTRATOR E D. HORN. Optimal ensemble averaging of neural networks. *Network: Computation in Neural Systems* 8(3), 283–296 (1997).
- [344] N. J. NILSSON. "Learning Machines: Foundations of Trainable Pattern-Classifying Systems". McGraw-Hill (1965).
- [345] M. H. NITECKI, editor. "Coevolution". The University of Chicago Press (1983).
- [346] D. A. NIX E A. S. WEIGEND. Learning local error bars for nonlinear regression. Em G. TESAURO, D. S. TOURETZKY E T. K. LEEN, editores, "Advances in Neural Information Processing Systems (NIPS)", tomo 7, páginas 488–496. MIT Press (1995).
- [347] E. NODA, A. L. V. COELHO, I. L. M. RICARTE, A. YAMAKAMI E A. A. FREITAS. Devising adaptive migration policies for cooperative distributed genetic algorithms. Em "Procs. of IEEE International Conference on Systems, Man and Cybernetics (SMC)", tomo 6, páginas 438–443, Tunis (outubro 2002). IEEE Press.
- [348] STEFANO NOLFI E DARIO FLOREANO. How co-evolution can enhance the adaptive power of artificial evolution: Implications for evolutionary robotics. Em "EvoRobots", páginas 22–38 (1998).
- [349] M. A. NOWAK E R. M. MAY. Evolutionary games and spatial chaos. *Nature* 359, 826–829 (1992).
- [350] G. M. P. O'HARE E N. R. JENNINGS, editores. "Foundations of Distributed Artificial Intelligence". Wiley (1996).
- [351] D. OPITZ. Feature selection for ensembles. Em "Procs. of Sixteenth National Conference on Artificial Intelligence (AAAI)", páginas 379–384 (1999).
- [352] D. OPITZ E R. MACLIN. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research* 11, 169–198 (1999).
- [353] D. OPITZ E J. SHAVLIK. A genetic algorithm approach for creating neural network ensembles. Em Sharkey [426], páginas 79–99.
- [354] D. W. OPITZ E J. W. SHAVLIK. Generating accurate and diverse members of a neural-network ensemble. Em D. S. TOURETZKY, M. C. MOZER E M. E. HASSELMO, editores, "Advances in Neural Information Processing Systems", tomo 8, páginas 535–541. The MIT Press (1996).
- [355] S. OSSOWSKI. "Co-ordination in Artificial Agent Societies", tomo 1535 de "LNAI". Springer-Verlag (1999).
- [356] L. PANAIT, R. P. WIEGAND E S. LUKE. A sensitivity analysis of a cooperative coevolutionary algorithm biased for optimization. Em "Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)", páginas 573–584. LNCS (2004).
- [357] J. PAREDIS. Coevolutionary constraint satisfaction. Em Y. DAVIDOR, H.-P. SCHWEFEL E R. MÄNNER, editores, "Parallel Problem Solving from Nature — PPSN III", tomo 866 de "LNCS", páginas 46–55, Berlin (1994). Springer.
- [358] J. PAREDIS. Coevolutionary computation. *Artificial Life Journal* 2(4), 355–375 (1995).

- [359] J. PAREDIS. The symbiotic evolution of solutions and their representations. Em "Procs. of the Sixth International Conference on Genetic Algorithms", páginas 359–365. Morgan Kaufmann (1995).
- [360] J. PAREDIS. Coevolving cellular automata: Be aware of the red queen! Em "Procs. of the International Conference on Genetic Algorithms", páginas 393–400. Morgan Kaufmann (1997).
- [361] R. PAREKH, J. YANG E V. HONAVAR. Constructive neural network learning algorithms for pattern classification. *IEEE Transactions on Neural Networks* 11(2), 436–451 (2000).
- [362] R. PATON. "Computing with Biological Metaphors". International Thompson Computer Press (1994).
- [363] C. A. PEÑA-REYES. "Coevolutionary Fuzzy Modeling". Tese de doutorado, École Polytechnique Fédérale de Lausanne — EPFL (2002).
- [364] C. A. PEÑA-REYES E M. SIPPER. Fuzzy CoCo: A cooperative-coevolutionary approach to fuzzy modeling. *IEEE Transactions on Fuzzy System* 9(5), 727–737 (outubro 2001).
- [365] W. PEDRYCZ, editor. "Granular Computing: An Emerging Paradigm". Physica-Verlag (2001).
- [366] W. PEDRYCZ E F. GOMIDE. A generalized fuzzy Petri net model. *IEEE Transactions on Fuzzy Systems* 2, 295–301 (1994).
- [367] W. PEDRYCZ E F. GOMIDE. "An Introduction to Fuzzy Sets: Analysis and Design". MIT Press (1998).
- [368] F. PENG, R. A. JACOBS E M. A. TANNER. Bayesian inference in mixtures-of-experts and hierarchical mixtures-of-experts models with an application to speech recognition. *Journal of the American Statistical Association* 91, 953–960 (1996).
- [369] M. P. PERRONE E L.N. COOPER. When networks disagree: Ensemble methods for hybrid neural networks. Em R. J. MAMMONE, editor, "Neural Networks for Speech and Image processing", páginas 126–142. Chapman & Hall (1993).
- [370] M. E. POLLACK E J. F. HORTY. There's more to life than making plans: Plan management in dynamic, multi-agent environments. *AI Magazine* (1999).
- [371] V. W. PORTO, D. B. FOGEL E L. J. FOGEL. Alternative neural network training methods. *IEEE Expert* 10(3), 16–22 (1995).
- [372] M. I. POSNER, S. E. PETERSON, P. T. FOX E M. E. REICHLE. Localization of cognitive operations in the human brain. *Science* 240, 1627–1631 (1988).
- [373] M. POTTER. A genetic cascade-correlation learning algorithm. Em L. D. WHITLEY E J. SCHAFER, editores, "Combinations of Genetic Algorithms and Neural Networks", páginas 123–133. IEEE Press (1992).
- [374] M. POTTER. "The Design and Analysis of a Computational Model of Cooperative Coevolution". Tese de doutorado, George Mason University (1997).
- [375] M. A. POTTER E K. A. DE JONG. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation* 8(1), 1–29 (2000).
- [376] M. A. POTTER, K. A. DE JONG E J. J. GREFENSTETTE. A coevolutionary approach to learning sequential decision rules. Em L. ESHELMAN, editor, "Proceedings of the Sixth International Conference on Genetic Algorithms", páginas 366–372, San Francisco, CA (1995). Morgan Kaufmann.
- [377] M. A. POTTER E K. DE JONG. A cooperative coevolutionary approach to function optimization. Em Y. DAVIDOR, H.-P. SCHWEFEL E R. MÄNNER, editores, "Parallel Problem Solving from Nature — PPSN III", LNCS, páginas 249–257. Springer, Berlin (1994).

- [378] M. V. NAGENDRA PRASAD, V. R. LESSER E S. E. LANDER. Learning organizational roles in a heterogeneous multi-agent system. Em "Procs. of the Second International Conference on Multiagent Systems", páginas 291–298 (1996).
- [379] L. PRECHELT. Proben1—A set of neural benchmarking rules. Relatório técnico 21/94, Universität Karlsruhe (1994).
- [380] N. PUPPALA, S. SEN E M. GORDIN. Shared memory based cooperative coevolution. Em "Procs. of the International Conference on Evolutionary Computation", páginas 570–574, Anchorage, USA (1998). IEEE Press.
- [381] V. RAMAMURTI E J. GHOSH. Structurally adaptive modular networks for nonstationary environments. *IEEE Transactions on Neural Networks* 10(1), 152–160 (janeiro 1999).
- [382] A. B. RAPOSO. "Coordenação em ambientes colaborativos usando Redes de Petri". Tese de doutorado, Faculdade de Engenharia Elétrica e de Computação, Unicamp (2000).
- [383] A. B. RAPOSO, A. L. V. COELHO, L. P. MAGALHÃES E I. L. M. RICARTE. Using fuzzy Petri nets to coordinate collaborative activities. Em "Procs. of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference", páginas 1494–1499, Vancouver, Canada (2001).
- [384] L. A. RASTRIGIN E R. H. ERENSTEIN. "Method of Collective Recognition". Energoizdat (1982). Em russo.
- [385] R. REED. Pruning algorithms—A survey. *IEEE Transactions on Neural Networks* 4(5), 740–747 (maio 1993).
- [386] R. D. REED E R. J. MARKS II. "Neural Smithing". MIT Press (1999).
- [387] A. N. REFENES, editor. "Neural Networks in the Capital Markets". John Wiley (1995).
- [388] J. REGGIA. <http://www.cs.umd.edu/~reggia/bic/diffs.html>.
- [389] S. O. REZENDE, editor. "Sistemas Inteligentes: Fundamentos e Aplicações". Manole (2003).
- [390] J. RIBEIRO. Modelos de redes neurais construtivas para classificação e verificação de padrões. Dissertação de mestrado, Departamento de Informática, Universidade Federal de Pernambuco (UFPE) (1999).
- [391] J. RIBEIRO E G. VASCONCELOS. Redes neurais construtivas para classificação de padrões. Em "Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação", tomo 4, páginas 53–64 (1999).
- [392] E. RICH E K. KNIGHT. "Inteligência Artificial". Makron Books (1993).
- [393] M. D. RICHARD E R. LIPPMANN. Neural network classifiers estimate Bayesian *a posteriori* probabilities. *Neural Computation* 3, 461–483 (1991).
- [394] R. W. RODIECK. "The Vertebrate Retina: Principles of Structure and Function". Freeman, New York (1973).
- [395] G. ROGOVA. Combining the results of several neural network classifiers. *Neural Networks* 7(5), 777–781 (1994).
- [396] F. ROLI, G. GIACINTO E G. VERNAZZA. Methods for designing multiple classifier systems. Em "Procs. of International Workshop on Multi-classifier Systems", tomo 2096 de "LNCS", páginas 78–87. Springer (2001).
- [397] B. ROSEN. Ensemble learning using decorrelated neural networks. *Connection Science* 8(3–4), 373–384 (1996).
- [398] J. ROSENSCHEIN E G. ZLOTKIN. "Rules of Encounter—Designing Conventions for Automated Negotiation among Computers". MIT Press (1994).
- [399] C. D. ROSIN E R. K. BELEW. New methods for competitive coevolution. *Evolutionary Computation* 5(1), 1–29 (1997).

- [400] B. J. ROSS. A Lamarckian evolution strategy for genetic algorithms. Em L. D. CHAMBERS, editor, "Practical Handbook of Genetic Algorithms: Complex Coding Systems", tomo 3, páginas 1–16. CRC Press, Boca Raton, Florida (1999).
- [401] D. E. RUMELHART. The architecture of mind: a connectionist approach. Em M. I. POSNER, editor, "Foundations of Cognitive Science", páginas 133–160. MIT Press (1989).
- [402] S. RUSSEL E P. NORVIG. "Artificial Intelligence: A Modern Approach". Prentice Hall (1995).
- [403] D. RUTA E B. GABRYS. Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting. Em "Procs. of International Workshop on Multi-classifier Systems", tomo 2096 de "LNCS", páginas 399–408. Springer (2001).
- [404] E. SAAVEDRA, I. RENNERS, A. GRAUEL, D. MORTON E H.J. CONVEY. Heuristics for kernels adaptation in support vector machines. Em A. LOTFI E J. M. GARIBALDI, editores, "Applications and Science in Soft Computing", Advances in Soft Computing Series, páginas 249–254. Springer-Verlag (2004).
- [405] M. SAHOTA. Real-time intelligent behavior in dynamic environments: Soccer-playing robots. Dissertação de mestrado, The University of British Columbia (1993).
- [406] S.L. SALZBERG. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery* 1(3), 317–327 (1997).
- [407] T. W. SANDHOLM. Distributed rational decision making. Em Weiß [475], páginas 201–258.
- [408] N. SARAVANAN E D. B. FOGEL. Evolving neural control systems. *IEEE Expert* 10(3), 23–27 (1995).
- [409] B. SARENI E L. KRÄHENBÜHL. Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation* 2(3), 97–106 (1998).
- [410] J. D. SCHAFFER, R. A. CARUANA E L. J. ESHELMAN. Using genetic search to exploit emergent behavior of neural networks. *Physica D* 42, 244–248 (1990).
- [411] R. SCHAPIRE, Y. FREUND, P. BARTLETT E W. LEE. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics* 26(5), 1651–1686 (1998).
- [412] R. E. SCHAPIRE. The strength of weak learnability. *Machine Learning* 5(2), 197–227 (1990).
- [413] T. SCHEURER. "Foundations of Computing". Addison-Wesley (1994).
- [414] B. SCHÖLKOPF, C. J. C. BURGESS E A. J. SMOLA, editores. "Advances in Kernel Methods—Support Vector Learning". MIT Press, Cambridge:MA (1998).
- [415] B. SCHÖLKOPF E A. SMOLA. "Learning with Kernels". MIT Press (2002).
- [416] A. SCHMIDT E Z. BANDAR. Modularity—A concept for new neural network architectures. Em "Procs. of IASTED International Conference on Computer Systems and Applications (CSA)", Jordan (1998).
- [417] O. G. SELFRIDGE. Pandemonium: A paradigm for learning. Em "Procs. of Symposium held at Physical Laboratory: Mechanisation Through Processing", páginas 511–527, London (1958). HMSO.
- [418] R. SEMOLINI. *Support vector machines, inferência transdutiva e o problema de classificação*. Dissertação de mestrado, Faculdade de Engenharia Elétrica e de Computação — Unicamp (2002).
- [419] S. SEN, M. SEKERAN E J. HALE. Learning to coordinate without sharing information. Em Huhns e Singh [214], páginas 509–514.

- [420] Y. G. SEO, S.-B. CHO E X. YAO. Emergence of cooperative coalition in NIPD game with localization of interaction and learning. Em "Procs. of IEEE Congress on Evolutionary Computation", páginas 877–884. IEEE Press (1999).
- [421] F. SEREDYNSKI. Coevolutionary game-theoretic multi-agent systems. Em "Procs. of International Symposium on Methodologies for Intelligent Systems", páginas 356–365 (1996).
- [422] F. SEREDYNSKI. Competitive coevolutionary multi-agent systems: The application to mapping and scheduling problems. *Journal of Parallel and Distributed Computing* 47(1), 39–57 (1997).
- [423] A. SHARKEY E N. SHARKEY. Combining diverse neural nets. *The Knowledge Engineering Review* 12(3), 231–247 (1997).
- [424] A. SHARKEY, N. SHARKEY E G. CHANDROTH. Neural nets and diversity. *Neural Computing and Applications* 4, 218–227 (1996).
- [425] A. J. C. SHARKEY. On combining artificial neural nets. *Connection Science* 8(3–4), 299–313 (1996).
- [426] A. J. C. SHARKEY, editor. "Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems". Springer-Verlag, London (1999).
- [427] A. J. C. SHARKEY. Multi-net systems. Em "Combining Artificial Neural Nets" [426], páginas 1–30.
- [428] A. J. C. SHARKEY. Types of multinet systems. Em "Procs. International Workshop on Multi-classifier Systems", tomo 2364 de "LNCS", páginas 108–117. Springer (2002).
- [429] O. M. SHEHORY, K. SYCARA E S. JHA. Multi-agent coordination through coalition formation. Em "Intelligent Agents IV: Agent Theories, Architectures, and Languages", tomo 1365 de "LNAI", páginas 143–154. Springer (1997).
- [430] M. P. SINGH, A. S. RAO E M. P. GEORGEFF. Formal methods in DAI: Logic-based representations and reasoning. Em Weiß [475], páginas 331–376.
- [431] M. SIPPER. "Machine Nature: The Coming Age of Bio-Inspired Computing". McGraw-Hill (2002).
- [432] K. SIRLANTZIS, M. C. FAIRHURST E M. S. HOQUE. Genetic algorithms for multi-classifier system configuration: A case study in character recognition. Em "Procs. of International Workshop on Multi-classifier Systems", tomo 2096 de "LNCS", páginas 99–108. Springer (2001).
- [433] F. J. ŚMIEJA. The pandemonium system of reflective agents. *IEEE Trans. on Neural Networks* 7(1), 97–106 (janeiro 1996).
- [434] J. MAYNARD SMITH. "Evolution and the Theory of Games". Cambridge University Press (1982).
- [435] R. SMITH. The Contract Net protocol—High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers* 29(12), 1104–1113 (1980).
- [436] A. J. SMOLA. "Learning with Kernels". Tese de doutorado, Technical University Berlin (1998).
- [437] P. SMOLENSKY, M. C. MOZER E D. E. RUMELHART, editores. "Mathematical Perspectives on Neural Networks". Developments in Connectionist Theory. Lawrence Erlbaum Associates, Publishers, New Jersey, USA (1996).
- [438] D. SOFGE, K. DE JONG E A. SCHULTZ. A blended population approach to cooperative coevolution for decomposition of complex problems. Em "Procs. of the IEEE International Conference on Evolutionary Computation", páginas 413–418. IEEE Press (2002).

- [439] E. SONTAG. VC dimension of neural networks. Em C. BISHOP, editor, "Neural Networks and Machine Learning", páginas 69–95. Springer (1998).
- [440] M. SRINIVAS E L. M. PATNAIK. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics* **24**(4), 656–667 (1994).
- [441] A. N. SRIVASTAVA, R. SU E A. S. WEIGEND. Data mining for features using scale-sensitive gated experts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**(12), 1268–1279 (dezembro 1999).
- [442] P. STONE, P. RILEY E M. VELOSO. The CMUnited-98 champion simulator team. Em M. ASADA E H. KITANO, editores, "RoboCup-98: Robot Soccer World Cup II". Springer Verlag (1999).
- [443] P. STONE, P. RILEY E M. VELOSO. The CMUnited-99 champion simulator team. Em M. VELOSO, E. PAGELLO E H. KITANO, editores, "RoboCup-99: Robot Soccer World Cup III". Springer Verlag (2000).
- [444] P. STONE E M. VELOSO. Task decomposition, dynamic role assignment and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence* **110**(2), 241–273 (1999).
- [445] M. SUGENO. Fuzzy measures and fuzzy integrals: A survey. Em "Fuzzy Automata and Decision Processes", páginas 89–102. Springer, Amsterdam:North Holland (1977).
- [446] J. A. K. SUYKENS, T. VAN GESTEL, J. DE BRABANTER, B. DE MOOR E J. VANDEWALLE. "Least Squares Support Vector Machines". World Scientific Pub. (2002).
- [447] I. TAHA E J. GHOSH. Symbolic interpretation of artificial neural networks. *IEEE Transactions on Knowledge and Data Engineering* **11**(3), 448–463 (1999).
- [448] M. TAMBE. Towards flexible teamwork. *Journal of Artificial Intelligence Research* **7**, 83–124 (1997).
- [449] J. N. THOMPSON. "The Coevolutionary Process". The University of Chicago Press (1994).
- [450] K. M. TING E I. H. WITTEN. Issues in stacked generalization. *Journal of Artificial Intelligence Research* **10**, 271–289 (1999).
- [451] D. TITTERINGTON, A. SMITH E U. MAKOV. "Statistical Analysis of Finite Mixture Distributions". Wiley (1985).
- [452] V. TRESP. Committee machines. Em Y. H. HU E J.-N. HWANG, editores, "Handbook for Neural Signal Processing". CRC Press (2001).
- [453] R. R. TRIPPI E E. TURBAN, editores. "Neural networks in finance and investing: Using artificial intelligence to improve real-world performance". Irwin (1996).
- [454] H. TSUKIMOTO. Extracting rules from trained neural networks. *IEEE Transactions on Neural Networks* **11**(2), 377–389 (2000).
- [455] K. TUMER E J. GHOSH. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition* **29**, 341–348 (1996).
- [456] K. TUMER E J. GHOSH. Linear and order statistics combiners for pattern classification. Em Sharkey [426], páginas 127–162.
- [457] K. UNO E A. NAMATAME. Evolutionary behaviors emerged through strategic interactions in the large. Em "Procs. of IEEE International Congress on Evolutionary Computation", páginas 1414–1421 (1999).
- [458] L. M. VAN VALEN. A new evolutionary law. *Evolutionary Theory* **1**, 1–30 (1973).

- [459] L. M. VAN VALEN. How pervasive is coevolution? Em Nitecki [345], páginas 1–19.
- [460] G. VALENTINI E T. DIETTERICH. Bias-variance analysis and ensembles of SVM. Em “Procs. of International Workshop on Multi-classifier Systems”, tomo 2364, páginas 222–231. Springer (2002).
- [461] G. VALENTINI E F. MASULLI. Ensembles of learning machines. Em “Procs. of Neural Nets Wirm Vietri-2002”, tomo 2486, páginas 3–19. Springer (2002).
- [462] V. N. VAPNIK. “The Nature of Statistical Learning Theory”. Springer-Verlag (1995).
- [463] V. N. VAPNIK. “Statistical Learning Theory”. John Willey (1998).
- [464] G. VASCONCELOS E J. RIBEIRO. An experimental evaluation of the Cascade-Correlation network in pattern recognition problems. Em “Procs. of the Fourth International Conference on Neural Information Processing”, páginas 1133–1136. Springer (1997).
- [465] L. VECCI, F. PIAZA E A. UNCINI. Learning and approximation capabilities of adaptive spline activation function neural networks. *Neural Networks* **11**, 259–270 (1998).
- [466] A. VERIKAS, A. LIPNICKAS, K. MALMQVIST, M. BACAUSKIENE E A. GELZINIS. Soft combination of neural classifiers: A comparative study. *Pattern Recognition Letters* **20**, 429–444 (1999).
- [467] V. VIANA. “Meta-heurísticas e programação paralela em otimização combinatória”. Edições UFC (1998).
- [468] E. WAN. Neural network classification: A Bayesian interpretation. *IEEE Transactions on Neural Networks* **1**(4), 303–305 (1990).
- [469] S. R. WATERHOUSE. “Divide and Conquer: Pattern Recognition using Mixtures of Experts”. Tese de doutorado, Cambridge University Engineering Department (1997).
- [470] A. S. WEIGEND E N. A. GERSHENFELD, editores. “Time Series Prediction: Forecasting the Future and Understanding the Past”. Addison-Wesley (1993).
- [471] A. S. WEIGEND, B. A. HUBERMAN E D. E. RUMELHART. Predicting the future: A connectionist approach. *International Journal of Neural Systems* **1**, 193–209 (1990).
- [472] A. S. WEIGEND, M. MANGEAS E A. N. SRIVASTAVA. Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. *International Journal of Neural Systems* **6**, 373–399 (1995).
- [473] D. WEINGAERTNER, V. K. TATAI, R. R. GUDWIN E F. V. ZUBEN. Hierarchical evolution of heterogeneous neural networks. Em “Procs. of the IEEE International Conference on Evolutionary Computation”, tomo 2, páginas 1775–1780, Hawaii (julho 2002).
- [474] G. WEISS. Learning to coordinate actions in multi-agent systems. Em Huhns e Singh [214], páginas 481–486.
- [475] G. WEISS, editor. “Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence”. MIT Press (1999).
- [476] G. WEISS E S. SEN, editores. “Adaptation and Learning in Multiagent Systems”, tomo 1042 de “LNAI”. Springer-Verlag (1996).
- [477] G. WEISS E S. SEN, editores. “Distributed Artificial Intelligence Meets Machine Learning”, tomo 1221 de “LNAI”. Springer-Verlag (1997).
- [478] M. P. WELLMAN. A computational market model for distributed configuration design. Em Huhns e Singh [214], páginas 371–379.

- [479] B. WHITEHEAD E T. CHOATE. Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Transactions on Neural Networks* 7(4), 869–880 (1996).
- [480] D. WHITLEY. Genetic algorithms and neural networks. Em “Genetic Algorithms in Engineering and Computer Science”. John Wiley & Sons (1995).
- [481] D. L. WHITLEY, V. S. GORDON E K. E. MATHIAS. Lamarckian evolution, the Baldwin effect and function optimization. Em Y. DAVIDOR, H.-P. SCHWEFEL E R. MÄNNER, editores, “Parallel Problem Solving from Nature — PPSN III”, tomo 866 de “LNCS”, páginas 6–15, Berlin (1994). Springer.
- [482] R. P. WIEGAND. Applying diffusion to a cooperative coevolutionary model. Em “Parallel Problem Solving from Nature — PPSN V”, LNCS, páginas 560–569. Springer, Berlin (1998).
- [483] R. P. WIEGAND. “An Analysis of Cooperative Coevolutionary Algorithms”. Tese de doutorado, George Mason University (2004).
- [484] R. P. WIEGAND, W. C. LILES E K. A. DE JONG. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. Em “Procs. of the Genetic and Evolutionary Computation Conference (GECCO-2001)”, páginas 1235–1242, San Francisco, California, USA (2001). Morgan Kaufmann.
- [485] R. P. WIEGAND, W. C. LILES E K. A. DE JONG. Analyzing cooperative coevolution with evolutionary game theory. Em “Procs. of the IEEE International Conference on Evolutionary Computation”, páginas 1600–1605. IEEE Press (2002).
- [486] R. P. WIEGAND, W. C. LILES E K. A. DE JONG. The effects of representational bias on collaboration methods in cooperative coevolution. Em “Parallel Problem Solving from Nature - PPSN VII”, tomo 2439 de “LNCS”, páginas 257–268. Springer (2002).
- [487] R. P. WIEGAND, W. C. LILES E K. A. DE JONG. Modeling variation in cooperative coevolution using evolutionary game theory. Em “Foundations of Genetic Algorithms VII”, páginas 203–220. Morgan Kaufman (2002).
- [488] S. W. WILSON. ZCS: A zeroth level classifier system. *Evolutionary Computation* 2(1), 1–18 (1994).
- [489] S. W. WILSON. Classifier fitness based on accuracy. *Evolutionary Computation* 3(2), 149–177 (1995).
- [490] T. WINDEATT E G. ARDESHIR. An empirical comparison of pruning methods for ensemble classifiers. Em “Intelligent Data Analysis IV”, tomo 2189 de “LNCS”, páginas 208–217. Springer (2001).
- [491] I. H. WITTEN E E. FRANK. “Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation”. Morgan Kaufmann Pub. (2000).
- [492] D. H. WOLPERT. Stacked generalization. *Neural Networks* 5(3), 241–260 (1992).
- [493] D. H. WOLPERT E W. G. MACREADY. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1(1), 67–82 (abril 1997).
- [494] K. WOODS, W.P. KEGELMEYER E K. BOWYER. Combination of multiple classifiers using local accuracy estimate. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, 405–410 (1997).
- [495] M. WOOLDRIDGE E N. R. JENNINGS. Intelligent agents: Theory and practice. *The Knowledge Engineering Review* 10(2), 115–152 (1995).
- [496] L. XU, M. I. JORDAN E G. E. HINTON. An alternative model for mixture of experts. Em G. TESAURO, D. S. TOURETZKY E T. K. LEEN, editores, “Advances in Neural Information Processing Systems (NIPS)”, tomo 7, páginas 633–640. MIT Press (1995).

- [497] L. XU, A. KRZYŻAK E C.Y. SUEN. Methods of combining multiple classifiers and their applications to hand-writing recognition. *IEEE Transactions on Systems, Man, and Cybernetics* **22**(3), 418–435 (maio 1992).
- [498] R. YAGER E J. KACPRZYK, editores. “The Ordered Weighted Averaging Operators: Theory and Applications”. Kluwer Academic Publishers (1997).
- [499] R. R. YAGER, J. KACPRZYK E M. FEDRIZZI, editores. “Advances in the Dempster-Shafer Theory of Evidence”. John Wiley & Sons, New York (1994).
- [500] S. YANG, A. BROWNE E P. PICTON. Multistage neural networks: Adaptive combination of ensemble results. Em A. LOTFI E J. M. GARIBALDI, editores, “Applications and Science in Soft Computing”, *Advances in Soft Computing Series*, páginas 47–52. Springer-Verlag (2004).
- [501] X. YAO. Evolving artificial neural networks. *Proceedings of the IEEE* **87**, 1423–1447 (1999).
- [502] X. YAO E Y. LIU. Ensemble structure of evolutionary artificial neural networks. Em “Procs. of IEEE International Conference on Evolutionary Computation”, páginas 659–664 (1996).
- [503] X. YAO E Y. LIU. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks* **8**(3), 694–713 (maio 1997).
- [504] X. YAO E Y. LIU. Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* **28**(3), 417–425 (abril 1998).
- [505] J. YEN E L. WANG. Application of statistical information criteria for optimal fuzzy model construction. *IEEE Transactions on Fuzzy Systems* **6**(3), 362–372 (1998).
- [506] X.-H. YU. Can Backpropagation error surface not have local minima? *IEEE Transactions on Neural Networks* **3**, 1019–1021 (1992).
- [507] L. A. ZADEH. Fuzzy logic. *IEEE Computer* **21**(4), 83–93 (1988).
- [508] B.-T. ZHANG E J.-G. JONG. Time series prediction using committee machines of evolutionary neural trees. Em “Procs. of IEEE International Conference on Evolutionary Computation”, tomo 1, páginas 281–286 (1999).
- [509] G. ZHANG, B. E. PATUWO E M. Y. HU. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* **14**, 35–62 (1998).
- [510] Q. ZHAO. A co-evolutionary algorithm for neural network learning. Em “Procs. of the International Conference on Neural Networks”, tomo 1, páginas 432–437 (1997).
- [511] Q. ZHAO. A general framework for cooperative co-evolutionary algorithms: A society model. Em “Procs. of the IEEE International Conference on Evolutionary Computation”, páginas 57–62, Anchorage, USA (1998).
- [512] Q. F. ZHAO, O. HAMMAMI, K. KURODA E K. SAITO. Cooperative co-evolutionary algorithm—How to evaluate a module? Em “Procs. of the IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks”, páginas 150–157 (2000).
- [513] Z.-H. ZHOU, J. WU E W. TANG. Ensembling neural networks: Many could be better than all. *Artificial Intelligence* **137**(1–2), 239–263 (2002).
- [514] Z.-H. ZHOU, J.-X. WU, Y. JIANG E S.-F. CHEN. Genetic algorithm based selective neural network ensembles. Em “Procs. of International Joint Conference on Artificial Intelligence”, tomo 2, páginas 797–802, Seattle (2001). AAAI Press.

-
- [515] Z.-H. ZHOU, J.-X. WU, W. TANG E Z.-Q. CHEN. Selectively ensembling neural classifiers. Em "Procs. of International Joint Conference on Neural Networks", tomo 2, páginas 1411–1415, Hawaii (2002). IEEE Press.
- [516] J. ZURADA, J. MARKS II E C. ROBINSON, editores. "Computational Intelligence—Imitating Life". IEEE Press (1994).