Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Engenharia de Computação e Automação Industrial

EMPREGO DE TEORIA DE AGENTES NO DESENVOLVIMENTO DE DISPOSITIVOS NEUROCOMPUTACIONAIS HÍBRIDOS E APLICAÇÃO AO CONTROLE E IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS

CLODOALDO APARECIDO DE MORAES LIMA

Orientador: PROF. DR. FERNANDO JOSÉ VON ZUBEN
(DCA – FEEC – UNICAMP)

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação (FEEC–UNICAMP) como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica.

Banca Examinadora: Prof. Dr. Marconi Kolm Madrid (FEEC – UNICAMP)
Prof. Dr. Francisco José Gomes (UFJF – MG)

CAMPINAS
Estado de São Paulo – Brasil
Fevereiro de 2000

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

Lima, Clodoaldo Aparecido de Moraes
L628e Emprego de teoria de agentes no desenvolvimento de dispositivos neurocomputacionais híbridos e aplicação ao controle e identificação de sistemas dinâmicos / Clodoaldo Aparecido de Moraes Lima.--Campinas, SP: [s.n.], 2000.

Orientador: Fernando José Von Zuben.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Otimização matemática. 2. Identificação de sistemas. 3. Redes neurais (Computação). 4. Sistemas inteligentes de controle. 5. Operadores não-lineares. 6. Teoria da aproximação. 7. Representação do conhecimento (Teoria da informação). 8. Modelos não-lineares (Estatística). I. Von Zuben, Fernando José. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

O objetivo deste trabalho foi analisar e sintetizar dispositivos neurocomputacionais híbridos, a serem comparados com estruturas de processamento alternativas e aplicados em identificação e controle de sistemas dinâmicos. Estes dispositivos neurocomputacionais representam uma generalização de modelos conexionistas clássicos, denominados redes neurais artificiais, e podem ser adequadamente descritos recorrendo-se a conceitos básicos de sistemas multi-agentes. Dessa forma, a arquitetura conexionista resultante, embora semelhante a arquiteturas clássicas multicamadas, é construída incrementalmente a partir da informação disponível associada ao problema que está sendo tratado. Com isso, além da definição da intensidade das conexões entre as unidades de processamento que compõem uma rede neural, é possível determinar automaticamente cada função de ativação destas unidades e também a dimensão final da rede neural generalizada. Logo, os dispositivos neurocomputacionais correspondem a: soluções híbridas, no sentido de que cada unidade de processamento apresenta sua própria função de ativação; dedicadas, no sentido de dependerem da natureza do problema em estudo; e parcimoniosas, no tocante ao uso de recursos computacionais em sua implementação. Todas estas propriedades são então exploradas na solução de problemas avançados de identificação e controle de sistemas dinâmicos, para os quais a planta é suposta ser desconhecida e não-linear. Como resultado, temos a superação de uma das principais limitações verificadas na aplicação de modelos conexionistas em tarefas avançadas de identificação e controle: o grau de flexibilidade dos modelos de identificação e controle pode agora ser determinado automaticamente.

Palavras chaves: redes neurais artificiais, dispositivos neucomputacionais, sistemas multi-agentes, identificação e controle de sistemas dinâmicos.

Abstract

The objective of this research was to analyze and synthesize hybrid neurocomputational devices, to be compared with alternative processing devices and applied to dynamic system identification and control. These neurocomputational devices represent a generalization of classic connectionist models, called artificial neural networks, and can be properly described by means of basic foundations of multi-agent systems. In this case, the resulting connectionist architecture, although similar to classical multilayer architectures, is incrementally built from the available information associated with the problem under consideration. As a consequence, besides the definition of the intensity of the connections among the processing units that compose a neural network, each activation function and the dimension of the final generalized neural network are automatically determined. The neurocomputational devices, therefore, correspond to: hybrid solutions, in the sense that each processing unit presents its own activation function; dedicated, in the sense of being a function of the nature of the problem under investigation; and parsimonious, in terms of the computational resources used in their implementation. All these properties are then explored in the solution of advanced dynamic system identification and control problems, where the plant is assumed to be unknown and nonlinear. As a result, we are able to overcome one of the main limitations present in applications of connectionist models to advanced tasks of identification and control: the degree of flexibility of the nonlinear identification and control models can now be automatically determined.

Key words: artificial neural networks, neurocomputational devices, multi-agent systems, dynamic system identification and control.

Dedico a meus pais, Eva e Fabiano, e
irmãos, Rivelino e Fabrício

Agradecimentos

Eu agradeço, primeiramente, ao Professor Fernando José Von Zuben pelo amplo apoio, indispensáveis ensinamentos, profícuas discussões e por proporcionar ambiente altamente propício para o desenvolvimento de pesquisa científica. Foi ele quem me introduziu no fascinante mundo das Redes Neurais Artificiais. Agradeço a todos os amigos que conviveram comigo no Laboratório de Engenharia de Computação e Automação Industrial, pelo clima de companheirismo e respeito mútuo, requisitos importantes na criação de condições de trabalho adequadas. Dentre todos, gostaria de ressaltar, em especial, os seguintes:

Mário Ernesto da Silva - pelas discussões e ensinamentos sobre a teoria de agentes, um dos alicerces centrais desta dissertação.

Eurípedes Pinheiro dos Santos - pelas sugestões e ajuda na implementação dos vários algoritmos utilizados nesta dissertação.

Moisés Vidal Ribeiro - pelas sugestões e questionamentos fundamentais para sedimentação dos vários temas abordados e ajuda na revisão e correção desta dissertação.

Gostaria de agradecer, em especial, a *Jeniffer Costa Nogueira* pelo carinho, amplo apoio e dedicação proporcionados não só nos momentos felizes, mas, principalmente, nas dificuldades enfrentadas no transcorrer desta dissertação. Sua tranqüilidade, incentivo e o ambiente familiar em sua casa foram fundamentais. Além disso, contribuiu decisivamente na revisão e correção desta dissertação.

Agradeço à Faculdade de Engenharia Elétrica e de Computação (FEEC) da Unicamp, pela oportunidade de realizar o curso de Mestrado, e aos responsáveis pela chefia do Departamento de Engenharia de Computação e Automação Industrial durante o período de desenvolvimento deste trabalho, por permitirem acesso às instalações.

Esta dissertação contou com o suporte financeiro da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processo no. 98/02417-4, sem o qual não seria possível o seu desenvolvimento. São iniciativas como estas que garantem ao Estado de São Paulo a liderança nacional em pesquisa científica e tecnológica.

Índice

Capítulo 8 – Conclusão e perspectivas futuras

Apêndice A – Alguns aspectos vinculados à teoria de agentes

Capítulo 1

Introdução

As teorias de identificação e controle de sistemas dinâmicos, como áreas de pesquisa multidisciplinar, têm experimentado avanços significativos nas últimas décadas, dando ênfase tanto a aspectos conceituais quanto técnicos. A parte mais desenvolvida destas teorias diz respeito ao tópico de sistemas lineares, sendo que importantes conceitos ou resultados teóricos foram obtidos, em áreas como sistemas multivariáveis, análise de estabilidade, controle ótimo, controle robusto e controle adaptativo.

Ferramentas muito eficientes de álgebra linear podem ser empregadas na análise de sistemas dinâmicos (planta ou processo), desde que o sistema possa ser adequadamente descrito por uma aproximação linear. Na realidade, muitos sistemas de importância prática são não-lineares em maior ou menor grau, sendo que uma parte não desprezível destes admite uma representação de suas propriedades em termos lineares (SCHWARZENBACH & GILL, 1992).

Projetos de controle de sistemas baseados em linearização são amplamente difundidos em diversos domínios, mas a crescente demanda por qualidade (por exemplo, maior capacidade de manipulação de imprecisão e incerteza) e o contínuo aumento na complexidade (por exemplo, maior intensidade dos fenômenos não-lineares envolvidos) dos problemas de identificação de sistemas e controle de processos têm evidenciado as limitações desta abordagem. Particularmente na presença de incertezas e não-linearidades significativas, as técnicas mais avançadas, desenvolvidas para o tratamento de sistemas lineares, não são capazes de responder ao conjunto de necessidades requeridas para atender aos critérios de desempenho geralmente adotados em identificação e controle de processos

atualmente em demanda (NARENDRA & ANNASWAMY, 1989). Sendo assim, embora a suposição da linearidade de sistemas dinâmicos tenha sido feita para desenvolver a teoria de controle, ela acaba reduzindo o alcance dos resultados, a ponto de dificultar a abordagem dos mais avançados problemas de engenharia atuais.

Os sistemas não-lineares, por sua vez, podem apresentar comportamentos dinâmicos mais complexos, por exemplo, com uma mudança qualitativa de comportamento para diferentes pontos de operação e com muito mais flexibilidade em sua evolução no tempo. Isto, no entanto, dificulta a formulação de uma teoria sistemática e geral, aplicável a projetos de identificação e controle não-linear. Esta impossibilidade atual de recorrer a ferramentas lineares efetivas, ou não-lineares genéricas, cria a necessidade de aplicação de ferramentas dedicadas, inerentemente não-lineares e com grande poder de adaptação, dentre as quais se destacam as redes neurais artificias (HUNT *et al.*, 1992).

Uma rede neural artificial é estudada e generalizada neste trabalho, dentro do contexto de processamento de informação e aproximação de funções. HAPPEL & MURRE (1994) definem um modelo conexionista como consistindo de muitos elementos autônomos básicos interconectados, denominados neurônios artificiais, com processamento paralelo. A função de transferência de cada um destes elementos é definida a partir de uma função de ativação simples. Deste modo, funções de maior complexidade emergem da forte interação estabelecida tomando-se um grande número destas unidades de processamento, nas mais variadas configurações de redes neurais.

Redes neurais artificiais representam, portanto, um direcionamento alternativo para a solução de problemas desafiadores em identificação e controle, principalmente junto àqueles que envolvem não-linearidades. O uso de redes neurais em sistemas de controle pode ser visto como um passo natural na evolução da metodologia de controle no sentido de buscar a superação de suas limitações. Analisando as técnicas existentes, a evolução na área de controle tem sido motivada por três grandes necessidades:

- tratar sistemas de grande complexidade, a qual vem crescendo continuamente;
- acompanhar o aumento de demanda por eficiência e qualidade, presente nos requisitos de projeto;
- tratar estes requisitos sob condições de incerteza e imprecisão.

Hoje, a necessidade de controle no tratamento de sistemas dinâmicos com complexidade crescente e diante de fatores de incerteza tem levado à reavaliação dos métodos convencionais e à proposição de métodos conceitualmente mais elaborados. Estas novas propostas incluem, por exemplo, níveis hierárquicos de decisão, planejamento e aprendizagem, que são necessários quando um alto grau de autonomia do sistema é desejável. Dadas estas observações, é natural que a comunidade da área de controle esteja pesquisando seriamente e ativamente em busca de novas ferramentas capazes de tratar eficientemente os problemas atuais de controle. A idéia de que "a necessidade é a força propulsora da invenção" tem sido literalmente aplicada desde a construção dos primeiros artefatos técnicos. Assim, metodologias baseadas em rede neurais representam um passo natural na evolução da teoria de controle.

1.1 Motivação da pesquisa

Quando comparado com os modelos lineares tradicionais, é certo que qualquer modelo não-linear adequadamente projetado é potencialmente capaz de produzir um melhor desempenho. Esta afirmação, no entanto, não contempla qualquer critério de ordem prática (quanto é possível explorar do potencial existente?), justificando o predomínio histórico de abordagens lineares, inclusive no tratamento de problemas inerentemente não-lineares. Conclui-se, então, que qualquer iniciativa de reverter este quadro, ou seja, de explorar a maior capacidade das abordagens não-lineares, incluindo aquelas baseadas na neurocomputação, deve invariavelmente estar associada a motivações de ordem prática.

É por esta razão que o atual crescimento do interesse pelo estudo e implementação de modelos conexionistas é justificado com base no aumento proporcional de poder de processamento computacional e armazenagem de informação a baixo custo, principalmente em arquiteturas convencionais de computadores, geralmente empregadas para operacionalizar "laboratórios de simulação e validação".

No entanto, para propósito de identificação e controle de processos, os modelos convencionais de redes neurais artificiais apresentam limitações, devidas principalmente a três fatores (RONCO, 1994; SZILAS & RONCO, 1995; RONCO & GAWTHOP, 1995):

1. não há um modo sistemático para determinar a estrutura de modelagem requerida para um dado sistema. Isto torna a capacidade de aproximação universal, associada às redes neurais artificiais, de pouco efeito prático;
2. o algoritmo iterativo de aprendizado (retropropagação), essencialmente baseado no gradiente descendente, nem sempre leva à convergência em direção à solução global, ou a uma solução local de boa qualidade, e geralmente requer um número excessivo de iterações, mesmo quando encontrada uma arquitetura adequada para a rede neural;
3. a característica distributiva e não-linear do processamento dificulta e até pode impedir, em alguns casos, a análise eficiente dos modelos conexionistas resultantes, durante e após o treinamento da rede.

Na tentativa de propor soluções para tais problemas, diferentes arquiteturas de redes neurais foram adotadas durante estes últimos anos. O objetivo era modelar o sistema em questão de maneira mais transparente e permitir um maior domínio sobre a flexibilidade das estruturas resultantes. Deste esforço surgiram, por exemplo, as redes neurais modulares, com computação local de informação (cada componente do vetor de entrada é computado por somente uma parte da arquitetura). A decomposição do espaço de entradas é, para o caso da rede modular, realizada de acordo com um conhecimento prévio acerca do sistema (RONCO & GAWTHOP, 1995; RONCO *et al.*, 1996).

Estes conceitos podem ser ainda mais estendidos, de modo a atribuir cada vez maior significado e importância ao papel individual que cada unidade de processamento, ou neurônio artificial, pode exercer dentro da estrutura conexionista. Seguindo esta tendência, é possível descrever uma rede neural como um sistema multi-agentes, capaz de construir soluções computacionais dedicadas para problemas de engenharia.

A utilização de conceitos computacionais derivados do estudo de agentes permite desenvolver ferramentas não-convencionais, caracterizadas pela cooperação e competição entre neurônios artificiais generalizados, também denominados agentes neurocomputacionais. Estes conceitos serão devidamente abordados ao longo dos próximos capítulos. Os métodos que usam tais conceitos levam a implementações computacionais que convergem para soluções dedicadas, capazes de explorar otimamente os recursos computacionais disponíveis, e que são denominadas aqui dispositivos neurocomputacionais híbridos.

1.2 Objetivos da pesquisa

Este trabalho foi desenvolvido considerando os seguintes objetivos básicos, que foram atendidos de modo seqüencial:

1. familiarização com conceitos e aplicações de modelos convencionais de redes neurais artificiais;

2. estudo e implementação de algoritmos avançados de treinamento supervisionado para modelos convencionais de redes neurais artificiais;

3. revisão bibliográfica da literatura associada à aplicação de modelos convencionais de redes neurais artificiais a identificação e controle de sistemas dinâmicos;

4. familiarização com conceitos de modelos não-convencionais de redes neurais artificiais, denominados dispositivos neurocomputacionais híbridos. Para tal estudo, é necessária uma familiarização com conceitos avançados de otimização não-linear de processos e de teoria de aproximação de funções, principalmente nos tópicos envolvendo modelos não-paramétricos que empregam operadores de projeção.

5. proposição de uma nova formalização para a descrição de dispositivos neurocomputacionais híbridos, recorrendo a conceitos básicos de sistemas computacionais multi-agentes;

6. aplicação de dispositivos neurocomputacionais híbridos a problemas de identificação e controle de sistemas dinâmicos, em que a planta é suposta ser desconhecida e apresenta não-linearidades significativas. Neste contexto, o dispositivo neurocomputacional pode ser interpretado como uma ferramenta não-paramétrica de modelagem, caracterizada pelo grande potencial de aproximação, e por produzir modelos computacionalmente parcimoniosos e com peculiaridades diretamente vinculadas à natureza do problema que está sendo tratado.

7. comparação de desempenho desta técnica não-paramétrica com técnicas alternativas, baseadas em estratégias paramétricas, e outras estratégias convencionais disponíveis.

1.3 Organização do texto

O texto está dividido em capítulos, que descrevem em detalhes todos os resultados obtidos na busca do atendimento dos objetivos enunciados na seção anterior. Sendo assim, iremos descrever a seguir a forma de organização em termos dos objetivos propostos.

Capítulo	Conteúdo
1	introdução, motivação, objetivos e descrição do conteúdo da dissertação
2	descrição dos resultados associados ao atendimento dos objetivos (1) e (2)
3	descrição dos resultados associados ao atendimento do objetivo (3), no tocante a estratégias de identificação de sistemas dinâmicos
4	descrição dos resultados associados ao atendimento do objetivo (3), no tocante a estratégias de controle de sistemas dinâmicos
5	descrição dos resultados associados ao atendimento do objetivo (4)
6	descrição dos resultados associados ao atendimento do objetivo (5)
7	descrição dos resultados associados ao atendimento dos objetivos (6) e (7)
8	comentários conclusivos e perspectivas futuras
Apêndice A	descrição genérica de conceitos básicos vinculados a agentes computacionais
Referências	listagem das referências bibliográficas
Índice de Autores	listagem dos autores citados com as respectivas páginas em que cada citação aparece no texto

Capítulo 2

Redes Neurais Artificiais

2.1 Introdução

Redes neurais artificiais são sistemas computacionais, de implementação em hardware e/ou software, que imitam algumas funcionalidades do sistema nervoso biológico, usando um grande número de elementos básicos interconectados, chamados neurônios artificias. Da interação destes elementos relativamente simples emerge um comportamento global coerente, e que pode ser adaptado a diversos contextos de aplicação.

Na tentativa de justificar como tal comportamento pode emergir a partir de implementações artificiais, em computadores digitais, surgiram na Inteligência Artificial (IA) dois paradigmas. O primeiro, chamado de simbolista, foi fortemente influenciado pelos estudos realizados em psicologia. Estes estudos se iniciaram com os trabalhos pioneiros de MCCARTHY (1963), MINSKY (1961) e NEWELL & SIMON (1972). A abordagem simbolista defende que a solução de problemas pode ser obtida através de um processo essencialmente algorítmico. Estas idéias geraram, por exemplo, linguagens de programação simbólicas voltadas para aplicações em IA, como *Lisp* e *Prolog* (RUSSELL & NORVIG, 1995).

Em contrapartida, o segundo paradigma, chamado conexionista, defende que é impossível transformar em algoritmos, isto é, reduzir a uma seqüência finita de passos lógicos e aritméticos, as diversas tarefas que a mente humana executa com facilidade e rapidez, envolvendo reconhecimento de imagens, visão, fala, compreensão e uso da linguagem, evocação de memória por associação, etc.

O conexionismo defende que a resolução de tarefas complexas está fundamentada no modo de operação de um sistema nervoso, de natureza completamente distinta daquele concebido para as arquiteturas computacionais do tipo Von Neumman, por ser baseado na computação paralela e distribuída de elementos básicos e funcionalmente simples, denominados neurônios. As propriedades provêm do comportamento coletivo destes elementos. Suas características básicas são: capacidade de aprendizado, generalização e adaptação. O princípio básico da aprendizagem está no ajuste dos pesos das conexões sinápticas em resposta a estímulos recebidos.

As motivações da pesquisa em redes neurais artificiais variam de acordo com a área de interesse. No campo biológico, podem ser mencionadas: a compreensão da estrutura do cérebro para explicar funções e comportamentos, a memória, os estudos de disfunções cerebrais, como a epilepsia e os efeitos causados por drogas. Na engenharia: a construção de sistemas de processamento de informação mais eficientes e a aplicação em controle e identificação de processos. Na ciência da computação: compreensão do processo de tratamento da informação empregado pelo cérebro, visando construir computadores e sistemas de representação eficientes e o uso de novas tecnologias para a resolução de problemas de alto grau de complexidade, como reconhecimento de imagens, recuperação de informação a partir de dados contendo ruídos, aprendizado de máquina.

Neste capítulo, apresentamos uma revisão de alguns conceitos importantes sobre redes neurais artificiais, enfatizando os tipos de redes estáticas (*rede perceptron multicamadas* (*MLP*) e *rede com funções de ativação de base radial* (*RBF*)) e dinâmicas (*rede totalmente recorrente, rede localmente e globalmente recorrente* e *rede com recorrência externa*).

2.2 Histórico

A primeira tentativa de construir um sistema conexionista artificial foi realizada por McCulloch & Pitts (1943). Eles desenvolveram um modelo matemático do neurônio presente em organismos vivos. Este modelo foi aprimorado posteriormente por Rosenblatt (1960) e Widrow & Hoff (1960). Eles chamaram os elementos adaptativos resultantes de *perceptron* e *adaline*, respectivamente, cuja maior contribuição diz respeito à implementação de uma regra de aprendizado baseada na minimização de uma aproximação

do erro quadrático médio, definido na saída do neurônio artificial, em relação a algum comportamento desejado de entrada-saída.

Um grande arrefecimento no interesse pelo estudo de redes neurais artificiais ocorreu alguns anos mais tarde, quando MINSKY & PAPERT (1969) demonstraram a capacidade limitada deste elemento adaptativo – o neurônio artificial – considerado isoladamente no mapeamento de operações lógicas, destacando-se o problema do OU-exclusivo. Neste período, a máquina de Turing – entidade computacional baseada em processamento seqüencial e memória independente do processador – era vista como a mais poderosa estrutura de processamento de informação, inclusive na área de inteligência artificial. Isto certamente não ajudou no desenvolvimento de uma teoria conexionista, baseada em processamento paralelo, até então analógico, e em memória distribuída.

Este quadro começou a se reverter a partir do início dos anos 80, quando neurofisiologistas e biofísicos começaram a defender a idéia de que o desenvolvimento mais avançado da inteligência artificial demandaria o uso de estruturas de processamento com propriedades ausentes nos computadores baseados na máquina de Turing: processamento paralelo e distribuído, capacidade de aprendizado e adaptação. Contribuindo para a elaboração desta nova concepção de processamento e manipulação de informação, Grossberg já havia tratado o problema de cooperação e competição entre células (GROSSBERG, 1973; GROSSBERG 1978). Entretanto, o ressurgimento efetivo das redes neurais artificiais é geralmente associado à publicação do trabalho sobre redes neurais totalmente recorrentes, desenvolvido por HOPFIELD (1982), e à apresentação do modelo de mapas auto-organizáveis, desenvolvido por KOHONEN (1982).

Um forte impulso para o desenvolvimento de modelos conexionistas foi dado quando RUMELHART & MCCLELLAND (1986) e LECUN (1987) apresentaram o algoritmo de aprendizado de retropropagação (*back-propagation*), indispensável para a viabilização do treinamento de redes neurais multicamadas, superando assim as limitações anteriormente apontadas por MINSKY & PAPERT (1969). Vale salientar que os resultados fundamentais empregados no desenvolvimento do algoritmo de retropropagação já haviam sido obtidos por WERBOS (1974), em um outro contexto (fato que retardou seu aproveitamento junto à área de redes neurais artificiais).

Até o início dos anos 90, em muitos campos científicos, vários pesquisadores estavam entusiasmados com as possibilidades de aplicação das redes neurais artificiais, especialmente considerando o perceptron multicamadas (*MLP – Multi-Layer Perceptron*). A justificativa para este entusiasmo pode ser atribuída à obtenção de demonstrações da capacidade de aproximação universal do *MLP*. Para um número adequado de neurônios na composição da camada intermediária da rede (aquela que não apresenta conexões diretas nem com as entradas nem com as saídas), com funções de ativação respeitando certas propriedades de suavidade, CYBENKO (1989b) e HORNIK (1993), dentre outros, demonstraram ser possível aproximar qualquer mapeamento contínuo definido em uma região compacta do espaço de aproximação (veja também LAPEDES & FARBER, 1987).

Entretanto, dependendo da complexidade do problema, os algoritmos de treinamento tradicionais para redes neurais artificiais apresentam-se computacionalmente ineficientes. Para contornar este problema, várias ferramentas de engenharia que podem contribuir para um aumento da eficiência dos algoritmos de treinamento destas redes têm sido propostas, como algoritmos genéticos e outras técnicas de computação evolutiva (IYODA, 2000), bem como estratégias não-paramétricas de aproximação (VON ZUBEN, 1996).

2.3 Modelos paramétricos e não-paramétricos

Modelos paramétricos são aqueles cuja forma do relacionamento funcional entre as variáveis dependentes e independentes é conhecida, mas podem existir parâmetros cujos valores são desconhecidos, embora passíveis de serem estimados a partir da informação disponível acerca do problema (geralmente representado por um conjunto de dados de treinamento). Já em relação aos *modelos não-paramétricos*, sua característica distintiva é a ausência (completa ou quase completa) de conhecimento prévio a respeito da forma da função, a qual também deve ser estimada a partir do ajuste de parâmetros livres. Sendo assim, o conjunto de "formas" que o modelo não-paramétrico pode assumir (classe de funções que o modelo do estimador pode prever) é muito amplo. Como conseqüência, existirá um número elevado de parâmetros (por exemplo, quando comparado à dimensionalidade da informação disponível ou ao número de dados de entrada-saída para treinamento), que não admitem uma interpretação física isolada. Portanto, a denominação

de modelos não-paraméticos não provém da ausência de parâmetros em sua definição, mas da impossibilidade de atribuir a cada parâmetro um significado ou uma interpretação individual, como ocorre em modelos paramétricos.

Com base nestes conceitos, podemos colocar os modelos de redes neurais dentro de uma ordem de parametrização, conforme ilustrado na figura 2.1.

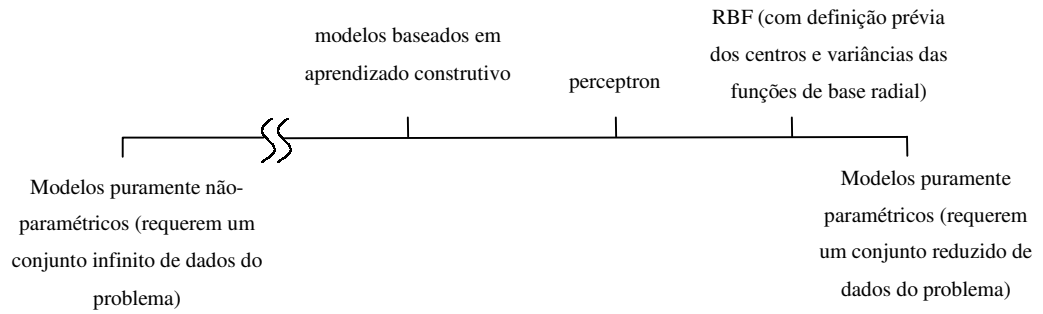


Figura 2.1 - Grau de parametrização de alguns modelos de redes neurais artificiais

2.4 Aspectos de Taxonomia e Arquitetura

Duas classes de redes neurais têm recebido considerável atenção junto a problemas de aplicação:

- redes neurais estáticas ou não-recorrentes;
- redes neurais recorrentes.

Uma típica rede neural estática, com uma camada de entrada, uma camada intermediária e uma camada de saída, é mostrada na figura 2.2. Por conveniência, esta pode ser apresentada na forma de um diagrama de blocos, como mostrado na figura 2.3, com duas matrizes de pesos $\mathbf{W^s}$ e $\mathbf{W^e}$. A rede multicamada representa um mapeamento que pode ser descrito da seguinte forma:

$$\hat{f}(\mathbf{x}) = F_s[\mathbf{W}^s F_e[\mathbf{W}^e \mathbf{x}]] \tag{2.1}$$

onde as matrizes $\mathbf{W^e}$ e $\mathbf{W^s}$ contêm os pesos ajustáveis e F_e e F_s são funções vetoriais e de argumento vetorial, suaves e sigmoidais (por exemplo, construídas a partir da função

tangente hiperbólica). Tal rede tem sido aplicada com sucesso em problemas de aproximação de funções não-lineares e reconhecimento de padrões, onde os pesos são ajustados no sentido de minimizar uma função-custo, que mede o erro ou distância até a solução ótima.

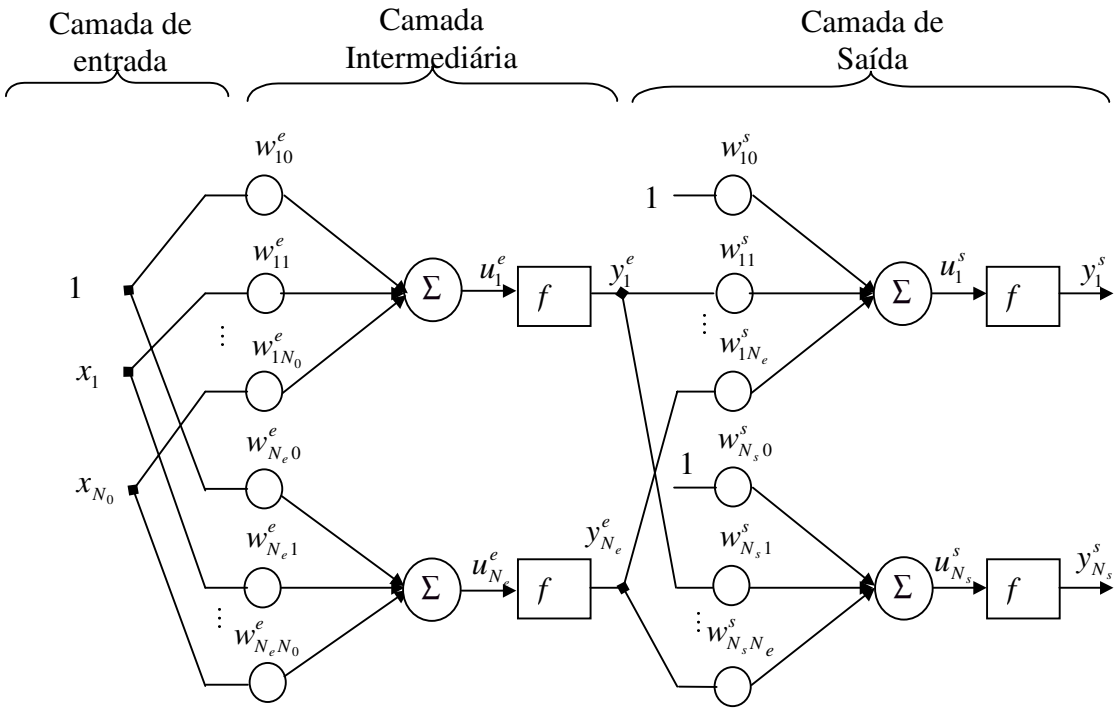


Figura 2.2 - Uma rede neural estática do tipo perceptron multicamadas (*MLP*), com uma camada intermediária. A camada de entrada pode ser interpretada como sendo constituída apenas por neurônios sensoriais, cuja função é distribuir as entradas, sem modificá-las, a todos os neurônios da camada intermediária.

Em contraste com as redes neurais estáticas, as redes dinâmicas ou recorrentes, mais conhecidas a partir do trabalho de HOPFIELD (1982), têm sido utilizadas como memórias endereçáveis por conteúdo, e também na solução de problemas de otimização e identificação de sistemas dinâmicos não-lineares. Uma versão da rede de Hopfield para tempo discreto é mostrada na figura 2.4, a qual consiste de uma rede neural com uma única camada, com atrasos unitários na malha de realimentação.

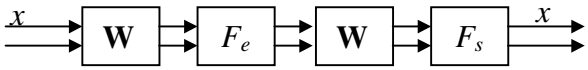


Figura 2.3 – Diagrama de blocos de uma rede neural estática com uma camada intermediária

A escolha dos pesos $\{W_{ij}\}$ determina os estados de equilíbrio para os quais as trajetórias de estado podem convergir. Se existirem múltiplos estados de equilíbrio estáveis, então a convergência para cada um deles dependerá da condição inicial. Quando em operação, uma rede neural de Hopfield corresponde a um sistema dinâmico não-linear e autônomo. É necessário impor restrições na escolha dos pesos sinápticos $\{W_{ij}\}$, de modo a evitar a ocorrência de outros comportamentos dinâmicos em estado estacionário que não sejam pontos de equilíbrio.

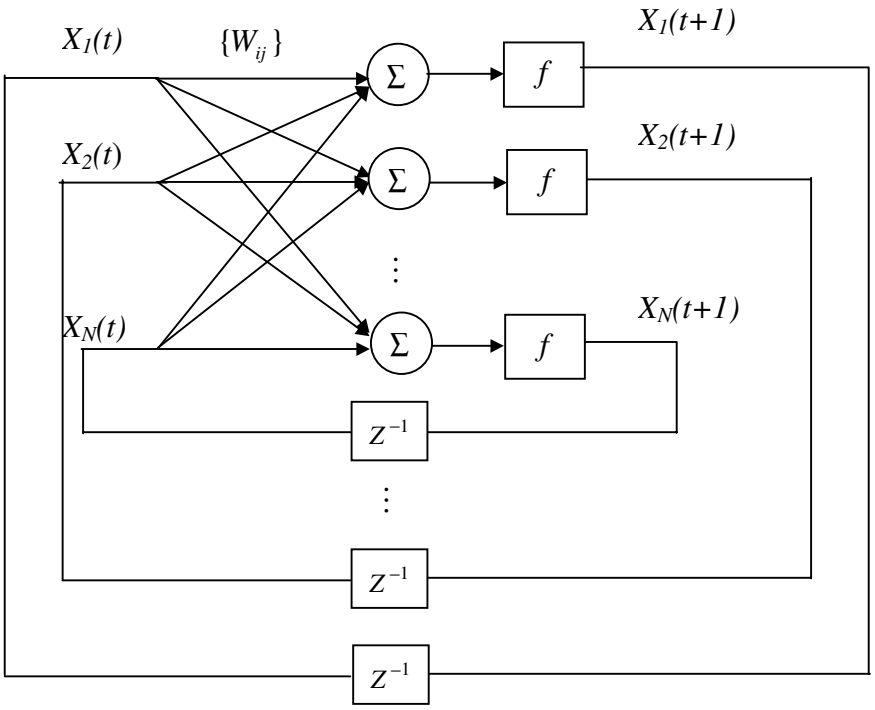


Figura 2.4 - Rede Neural de Hopfield de tempo discreto

De um ponto de vista funcional, a rede estática representa simplesmente uma estrutura versátil para a realização de mapeamentos não-lineares. As redes recorrentes, por outro lado, não necessariamente restrito ao caso da rede de Hopfield, são capazes de

mapear uma ampla gama de sistemas dinâmicos não-lineares. Em análise de sistemas, ambas exercem um importante papel. Na figura 2.5, apresentamos em forma de diagramas de blocos (semelhante à representação adotada na figura 2.3), possíveis composições de tais redes para modelagem de sistemas dinâmicos. NARENDRA & PARTHASARATHY (1988b) usaram o algoritmo de retropropagação para ajustar os pesos de uma rede neural estática e também sugeriram extensões para o mesmo, no sentido de possibilitar o ajuste dos pesos nos casos específicos de redes recorrentes.

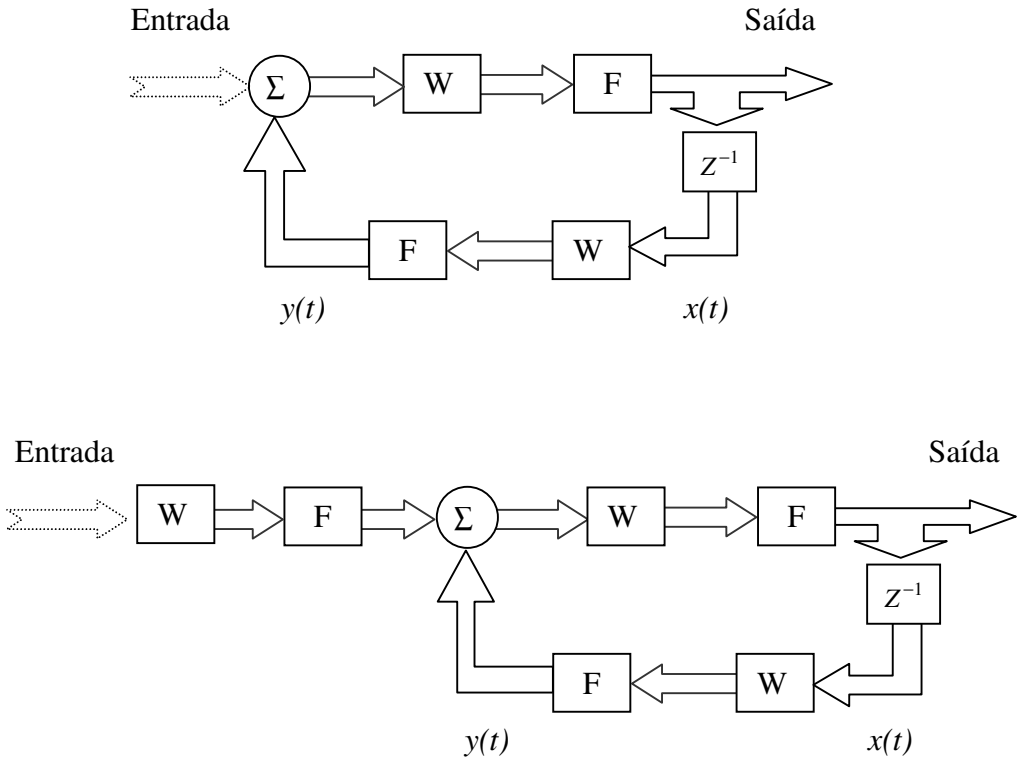


Figura 2.5 - Exemplo de duas configurações de redes recorrentes para identificação de sistemas dinâmicos

HAYKIN (1999) afirma que o aprendizado de uma rede neural é o processo pelo qual os parâmetros livres da rede neural são adaptados através de um mecanismo continuado de estimulação produzida pelo ambiente no qual a rede está inserida. O paradigma de aprendizado é determinado pelo modo através do qual os parâmetros mudam de valor, ou seja, a maneira pela qual o ambiente influência no comportamento da rede, resultando em:

- Aprendizado supervisionado;
- Aprendizado por reforço;
- Aprendizado não-supervisionado.

Aprendizado supervisionado (será detalhado nas próximas seções): é baseado em um conjunto de relações de estímulo-resposta, ou em algum outro tipo de informação, que represente o comportamento que deve ser aprendido pela rede neural. Um ingrediente essencial ao aprendizado supervisionado é a disponibilidade de um "professor" ou supervisor. Em termos conceituais, podemos pensar no professor como tendo o conhecimento do ambiente, que é representado pelo conjunto de relações entrada-saída. O ambiente é, portanto, inicialmente desconhecido para a rede neural. Suponha agora que, ao professor e à rede neural, sejam apresentadas entradas originadas do ambiente. Em virtude de seu conhecimento, ao professor é permitido proporcionar à rede neural uma resposta desejada para aquela entrada específica. Na verdade, a resposta desejada representa a ação ótima a ser adotada pela rede neural. Os parâmetros da rede são ajustados sob a influência combinada do vetor de entrada e do sinal de erro. Em outras palavras, com o aprendizado supervisionado, o conhecimento disponível ao professor é indiretamente transferido à rede neural.

Aprendizado por reforço (SUTTON & BARTO, 1998)*:* o comportamento da rede é avaliado apenas com base em algum critério numérico, fornecido em instantes espaçados de tempo. É um procedimento que procura maximizar um índice escalar de desempenho chamado sinal de reforço. Neste, ocorre a avaliação indireta do comportamento, não indicando explicitamente, se o melhoramento é possível, ou como o sistema deverá mudar seu comportamento para melhor atender os objetivos.

Aprendizado não-supervisionado (KOHONEN, 1997): é baseado apenas nos estímulos recebidos pela rede neural. Basicamente, a rede deve aprender a categorizar os estímulos. Neste caso, não há um professor ou crítico inspecionando o processo de aprendizado. Em outras palavras, não existe um exemplo específico da função a ser aprendida pela rede. Ao contrário, seus parâmetros livres são ajustados no sentido de extrair alguma tendência estatística presente nos dados de entrada. Uma vez que a rede concluiu um processo de auto-organização baseado nos dados de entrada, ela explora a representação interna gerada para discriminar características presentes na entrada.

Neste trabalho, iremos considerar particularmente o paradigma supervisionado. Historicamente, o treinamento supervisionado de uma rede *MLP* usa o algoritmo de retropropagação para o cálculo do vetor gradiente. Uma vez disponível o vetor gradiente, é aplicado um algoritmo iterativo, baseado no gradiente descendente, para minimizar uma medida de erro quadrático médio (*EQM*) entre o vetor de resposta desejada e o vetor de resposta obtida (CHEN & PAO, 1989). Há dois tipos comuns de aprendizado para o algoritmo do gradiente: em *batelada* e *seqüencial*. O método em *batelada* atualiza os pesos depois da apresentação completa do conjunto de dados de treinamento. Assim, uma iteração de treinamento realiza uma varredura através de todos os padrões de treinamento. Por sua vez, o método *seqüencial* ajusta os parâmetros imediatamente após a apresentação de cada padrão de treinamento, e representa uma forma de aproximação de natureza estocástica, já que a seqüência de apresentação de padrões é aleatória (NARENDRA & PARTHASARATHY, 1988a).

Enquanto o algoritmo de retropropagação proporciona uma liberdade para a definição do número de camadas, importante na garantia de uma capacidade de aproximação adequada, o treinamento pode ficar muito lento e ineficiente, já que somente a informação da primeira derivada (ou gradiente) do erro de treinamento é utilizada ao longo do processo iterativo de otimização. Para acelerar o processo de treinamento, alguns algoritmo de otimização de segunda ordem, baseados no gradiente e na informação da segunda derivada (ou Hessiana) têm sido propostos (HWANG & LEWIS, 1990). Por exemplo, o método de Newton (PARKER, 1985), o método quase-Newton (WATROUS, 1987), o método do gradiente conjugado (SHANNON, 1990), o método de Gauss-Newton (KOLLIAS & ANASTASSIOU, 1989), bem como o método do gradiente conjugado híbrido (DOS SANTOS & VON ZUBEN, 2000).

2.4.1 Retropropagação em sistemas estáticos

A retropropagação (*back-propagation*) é um método computacionalmente eficiente para o cálculo da informação de 1ª ordem (derivada parcial do critério de erro em relação aos pesos de uma rede neural multicamada), ou vetor gradiente. Com a disponibilidade desta informação, os pesos são ajustados iterativamente na direção oposta à indicada pelo vetor gradiente, visando minimizar a função de erro no processo de treinamento

supervisionado. NARENDRA & PARTHASARATHY (1988a) apresentam uma representação diagramática para a implementação da retropropagação, a qual tem sido utilizada em estudos de simulação envolvendo tanto sistemas dinâmicos como estáticos. A estrutura da matriz de pesos usada para computar as derivadas (chamada de rede de sensibilidade) é análoga à da rede original, mas com os sinais fluindo em direção oposta, justificando o termo retropropagação. A representação diagramática é também adequada para a implementação algorítmica do processo de retropropagação.

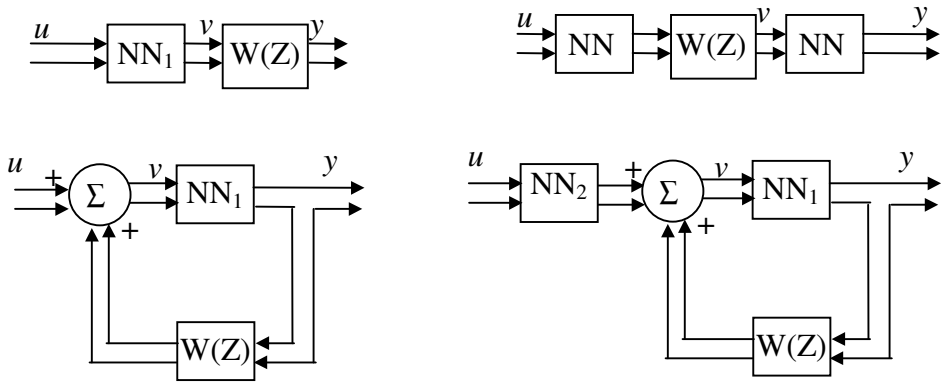
2.4.2 Retropropagação em sistemas dinâmicos

Muitos sistemas físicos são de natureza dinâmica e a identificação de tais sistemas a partir de dados de entrada-saída, expressando seu comportamento no tempo, certamente envolve elementos dinâmicos. Além disso, se tais sistemas devem ser eficientemente controlados, o conceito de realimentação deve ser explorado em toda sua potencialidade. Como os sistemas realimentados devem invariavelmente conter elementos de memória, as redes neurais a serem empregadas em identificação ou controle devem incorporar tais elementos. Por exemplo, em implementações envolvendo tempo discreto, elementos de memória na forma de atrasadores são necessários para impedir que um mesmo sinal apareça simultaneamente na saída e na entrada de algum módulo do sistema.

Já que a retropropagação é empregada na obtenção da informação de 1ª ordem, a ser utilizada para ajustar os pesos de redes neurais realimentadas, é necessário estender seus procedimentos para também manipular o efeito das realimentações. Se atrasos estão presentes em cascata na composição do vetor de entrada da rede neural, o mesmo método (representação diagramática) para redes estáticas pode ser convenientemente empregado na produção das derivadas parciais necessárias. Uma questão prática é como determinar estas derivadas parciais quando a rede neural e elementos dinâmicos estiverem conectados em configurações arbitrárias.

A introdução de elementos dinâmicos no processamento da rede neural requer uma investigação mais refinada do conceito de derivadas parciais. NARENDRA & PARTHASARATHY (1988b) apresentam um tratamento extensivo sobre o assunto. A figura 2.6 mostra algumas estruturas nas quais as redes neurais $NN_{(.)}$ e os elementos dinâmicos $W(z)$ são conectados e realimentados. Estas estruturas representam a construção básica de

blocos de modelos de identificação e controle. Se a retropropagação dinâmica pode ser desenvolvida para determinar as derivadas parciais do erro de saída em relação a algum parâmetro da rede neural em função do tempo, métodos de otimização que empregam o vetor gradiente também podem ser estendidos para o treinamento de redes neurais em tais configurações.



$NN_{(.)}$: bloco representando uma rede neural

Figura 2.6 - Construindo blocos de modelos de identificação e controle

Para todos os esquemas de identificação apresentados na figura 2.6, NARENDRA & PARTHASARATHY (1990) descrevem o procedimento de estimação de $\partial e(k)/\partial w_{ij}$ como saída da rede de sensibilidade. Assim, todas as derivadas parciais podem ser definidas rigorosamente. No entanto, para garantir a validade da estimação, a alteração em w_{ij} deverá ser restrita a uma vizinhança em torno dos valores atuais, tornando elevado o número de iterações necessário para obter o modelo de identificação.

2.5 Redes Neurais Estáticas

2.5.1 Rede perceptron multicamada (MLP)

O *perceptron* multicamada (*MLP – Multilayer Perceptron*) é uma rede estática de processamento direto (não-realimentado) composta de H camadas intermediárias (raramente tem-se $H>2$). A figura 2.2 apresenta um *MLP* com uma camada intermediária ($H = 1$). O vetor $\mathbf{x} = [x_1 \; x_2 \cdots x_{N_0}]^T$ contém N_0 entradas e o vetor $\mathbf{y^s} = [y_1^s \; y_2^s \cdots y_{N_s}^s]^T$

contém as N_s saídas da rede. A camada intermediária é composta de N_e neurônios do tipo perceptron.

Como vamos considerar neste estudo redes neurais com uma única camada intermediária, utilizamos super-índices *e* e *s* para indicar camada intermediária e de saída, respectivamente. Quando for interessante considerar mais camadas intermediárias, torna-se conveniente utilizar rótulos numéricos. Neste caso, usaríamos $e = 0$ e $s = 1$. Sendo assim, cada *perceptron k* da camada *i* recebe as entradas de todas as unidades $j = 1, \cdots, N_{i-1}$ da camada anterior ($i-1$). A saída é baseada em uma ativação interna (soma ponderada) e uma função de ativação (figura 2.7). Primeiro, o *perceptron k* executa a soma de todas as entradas, ponderadas pelos correspondentes pesos. Esta operação produz um potencial de ativação interna u_k^i.

$$u_k^i = \sum_{j=0}^{N_{i-1}} y_j^{i-1} w_{kj}^i \tag{2.2}$$

onde $y_j^{i-1} = x_j$ se $i = 1$, $y_0^{i-1} = 1$, w_{kj} é o *j*-ésimo peso do neurônio *k* e w_{k0}^i é o bias ou nível de polarização (figura 2.7).

O potencial u_k^i é então introduzido como argumento de uma função de ativação, linear ou não-linear, para determinar a saída y_k^i do neurônio *k* da camada *i*. Funções de ativação sigmoidais são geralmente adotadas, destacando-se a função tangente hiperbólica. A saída do neurônio assume então a forma:

$$y_k^i = \tanh(\tau \cdot u_k^i) = \frac{e^{\tau \cdot u_k^i} - e^{-\tau \cdot u_k^i}}{e^{\tau \cdot u_k^i} + e^{-\tau \cdot u_k^i}} \tag{2.3}$$

onde τ é uma constante (freqüentemente igual a 1) que determina a inclinação da função sigmoidal.

Usando a equação (2.3), podemos determinar a saída da rede neural *MLP* ativando-a da esquerda para a direita, camada a camada (ativação direta). Demonstrações da capacidade universal de aproximação apresentada por uma rede neural com uma camada intermediária podem ser encontradas em HORNIK (1993), CYBENKO (1989a) e LAPEDES & FARBER (1987).

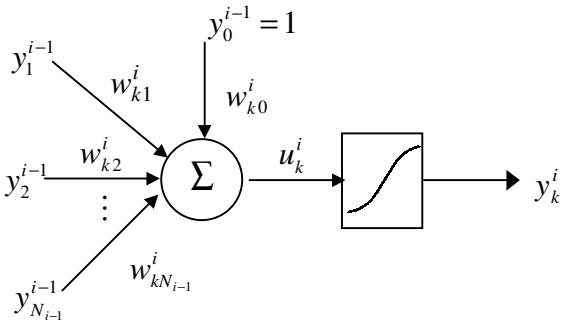


Figura 2.7 Componentes do neurônio *k* da camada *i* de uma rede neural *MLP*

Como já mencionado, um modo simples de introduzir dinâmica no processamento das redes *MLP* consiste no uso de um vetor de entradas **x** composto de valores passados de entradas e saídas do sistema dinâmico (figura 2.8). Desta maneira, a *MLP* pode ser interpretada como um modelo *NARX* (auto-regressivo não-linear com entrada externa) do sistema (CHEN *et al.*, 1990), como será melhor detalhado no capítulo 3. Um modelo *NARX* para um sistema dinâmico com uma entrada e uma saída pode ser expresso como a seguir:

$$\hat{y}(k) = f(y(k-1),\cdots,y(k-n);u(k-q),\cdots,u(k-q-m-1)) = y(k)+e(t) \quad (2.4)$$

onde $u(k)$ e $y(k)$ são exemplos do processo de entrada e saída para o instante de tempo k, q é um tempo de atraso de entrada-saída, n denota o número de saídas passadas usadas no modelo, m denota o número de entradas passadas usadas no modelo, f é uma função não-linear descrevendo o comportamento dinâmico do sistema, e $e(k)$ é o erro de aproximação.

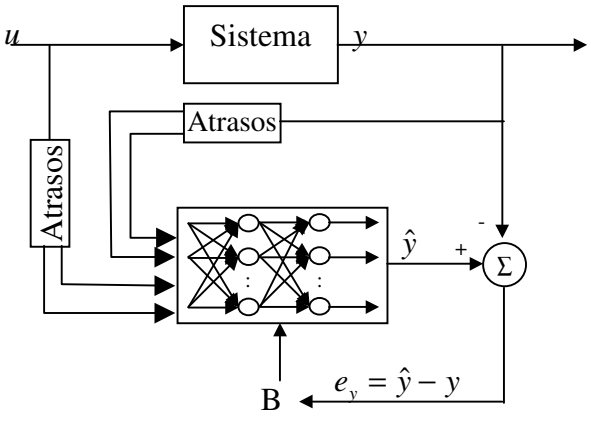


Figura 2.8 - Modelo para identificação de comportamento dinâmico de entrada-saída usando uma rede neural *MLP*

Este modo de introduzir dinâmica junto às redes estáticas tem a vantagem da fácil implementação, mas com propriedades dinâmicas restritas, se comparado às redes recorrentes (GOMM *et al.*, 1997). Por outro lado, as redes recorrentes apresentam maior complexidade de aprendizado (VENUGOPAL *et al.*, 1994).

2.5.2 Rede neural com funções de ativação de base radial (RBF)

As redes neurais com funções de ativação de base radial são redes não-realimentadas com três diferenças principais em relação às redes tipo perceptron multicamadas (*MLP*):

- elas sempre apresentam uma única camada intermediária;
- neurônios de saída são sempre lineares;
- os neurônios da camada intermediária têm funções de base radial como função de ativação, ao invés de uma função sigmoidal.

As funções de base radial $\varphi_i(x)$ usualmente são funções gaussianas do tipo:

$$\varphi_i(x_1, x_2, \cdots, x_n) = e^{\left[-\frac{(x_1-c_{1i})^2}{\sigma_{1i}^2}-\frac{(x_2-c_{2i})^2}{\sigma_{2i}^2}-\cdots-\frac{(x_n-c_{ni})^2}{\sigma_{ni}^2}\right]} \tag{2.5}$$

onde $\mathbf{c}_i = [c_{1i} \quad c_{2i} \quad \cdots \quad c_{ni}]^T$ é um vetor que define o centro da função de base radial do neurônio *i*, e $\sigma_i = [\sigma_{1i} \quad \sigma_{2i} \quad \cdots \quad \sigma_{ni}]^T$ determina a forma de dispersão da base da função em cada eixo de coordenada. Os padrões a serem apresentados na entrada $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^T$ ativam os neurônios de acordo com as distâncias aos centros $\mathbf{c}_i$, $i = 1,...,n$, e também de acordo com a extensão da base da respectiva função gaussiana. Assim, cada neurônio apresenta uma ativação mais forte para entradas que estão mais próximas do seu centro, de acordo com a norma ponderada:

$$\|\mathbf{x} - \mathbf{c}_i\|_{\Sigma_i} = \sqrt{\frac{1}{2}(\mathbf{x} - \mathbf{c}_i)^T \Sigma_i^{-1} (\mathbf{x} - \mathbf{c}_i)} \tag{2.6}$$

onde

$$\Sigma_i = \begin{bmatrix} \sigma_{1i} & 0 & \cdots & 0 \\ 0 & \sigma_{2i} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{ni} \end{bmatrix} \tag{2.7}$$

Outras funções de base radial podem também ser usadas como funções de ativação, sem alterar significativamente o desempenho (HAYKIN, 1999). A arquitetura da rede é apresentada na figura 2.9. As saídas O_j $(j = 1,2,\cdots,m)$ representam uma combinação linear das funções de ativação de base radial, com os pesos w_{ij} desempenhando o papel de ponderadores de cada ativação *i* para a saída *j*.

Várias abordagens para o treinamento de redes com funções de ativação de base radial têm sido propostas. Geralmente, elas podem ser divididas em duas etapas:

- definição dos centros, forma e extensão das funções de base radial, normalmente baseada em treinamento não-supervisionado (quantização vetorial ou treinamento competitivo) ou computação evolutiva (algoritmo genético, programação/estratégia evolutiva);
- aprendizado dos pesos da camada de saída, responsáveis pela combinação linear das ativações da camada intermediária, empregando técnicas como pseudo-inversão e OLS (*Orthogonal Least Squares*).

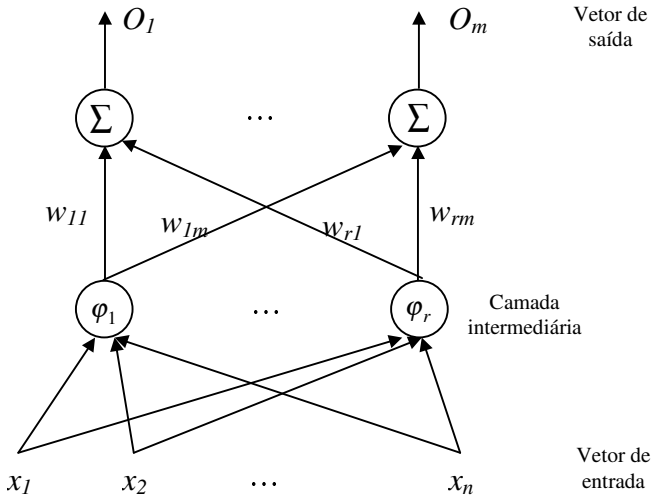


Figura 2.9 Uma rede *RBF* típica com *n* entradas, *r* funções de base radial e *m* saídas

Em BILLINGS & ZHENG (1995), é proposto um algoritmo genético para evoluir somente os centros, sendo que estes assumem valores distintos, devido ao fato de que usando centros idênticos nas redes não há melhoria da capacidade de aproximação. Esta representação para seleção de subconjuntos e operadores genéticos foi proposta por LUCASISUS & KATERMAN (1992) e usada por FONSECA *et al.* (1993) para a seleção de termos em modelos *NARMAX*. Após determinados os centros das funções, os pesos são encontrados utilizando o algoritmo *OLS*. Em MOCLOONE *et al*. (1998), é apresentada uma estratégia híbrida, a qual divide o treinamento da rede em duas etapas: treinamento da parte não-linear (centros e variâncias), utilizando o método do gradiente, e treinamento da parte linear (pesos), utilizando *OLS*.

O algoritmo de aprendizado proposto por HOLCOMB & MORARI (1991) define funções de base radial para a camada intermediária de modo que cada função seja aproximadamente ortogonal às demais. O algoritmo inicia com uma camada intermediária de um neurônio e outros neurônios são adicionados à rede quando necessário. A localização dos neurônios da camada intermediária é otimizada usando técnicas convencionais de otimização. Este algoritmo permite o uso de métodos avançados de otimização para treinar a rede. Entretanto, o critério de término não é robusto e pode ser dependente do problema em questão (BILLINGS & ZHENG, 1995).

Já em relação aos algoritmos propostos por LEE & RHEE (1991) e MUSAVI *et al.* (1992), são utilizados métodos de clusterização hierárquica supervisionada. No trabalho de LEE & RHEE (1991), o aprendizado começa com um neurônio na camada intermediária com uma grande dispersão e introduz-se neurônios adicionais quando necessário. A dispersão associada às funções de base radial pode também ser alterada. No trabalho de MUSAVI *et al.* (1992), o aprendizado começa com um grande número de neurônios, sendo que alguns são combinados (resultando em um único) quando possível. As dispersões e centros associados são devidamente atualizados, a partir daquelas previamente existentes antes da combinação. Estes dois algoritmos foram primeiramente desenvolvidos para reconhecimento de padrões e oferecem um caminho alternativo para definir a estrutura da rede. Uma vez que o número de neurônios, localizações (centros) e dispersões são determinados por clusterização (agrupamento) dos vetores de entrada, eles podem não fornecer soluções ótimas. Isto pode ser especialmente válido para problemas de identificação (BILLINGS & ZHENG, 1995).

Dado um número suficiente de neurônios com função de ativação de base radial, qualquer função contínua definida numa região compacta pode ser devidamente aproximada usando uma rede *RBF* (CHO *et al.*, 1998). As redes *RBF* são redes de aprendizado local, de modo que é possível chegar a uma boa aproximação desde que um número suficiente de dados para treinamento seja fornecido na região de interesse. Em contraste, redes neurais *MLP* são redes de aprendizado não-local, em virtude do modo como é definido o argumento das funções de ativação (elas fazem o papel de *ridge functions*, conforme descrito em VON ZUBEN (1996)).

Redes neurais com capacidade de aproximação local são muito eficientes quando o número de entradas é pequeno. Entretanto, quando o número de entradas não é pequeno, as redes *MLP* apresentam uma maior capacidade de generalização (HAYKIN, 1999). Isto ocorre porque o número de funções de base radial deve aumentar exponencialmente com o aumento da dimensão do espaço de entrada, de modo que as redes *RBF* são mais sensíveis à maldição da dimensionalidade (VON ZUBEN, 1996).

2.6 Redes Neurais Recorrentes

Redes neurais recorrentes são estruturas de processamento capazes de representar uma grande variedade de comportamentos dinâmicos não-lineares. A presença de realimentação de informação permite a criação de representações internas e dispositivos de memória capazes de processar e armazenar informações temporais e sinais seqüenciais.

Por exemplo, vários tipos de redes recorrentes são capazes de inferir gramáticas regulares a partir de exemplos (CLEEREMANS *et al.*, 1989; GILES *et al.*, 1992).

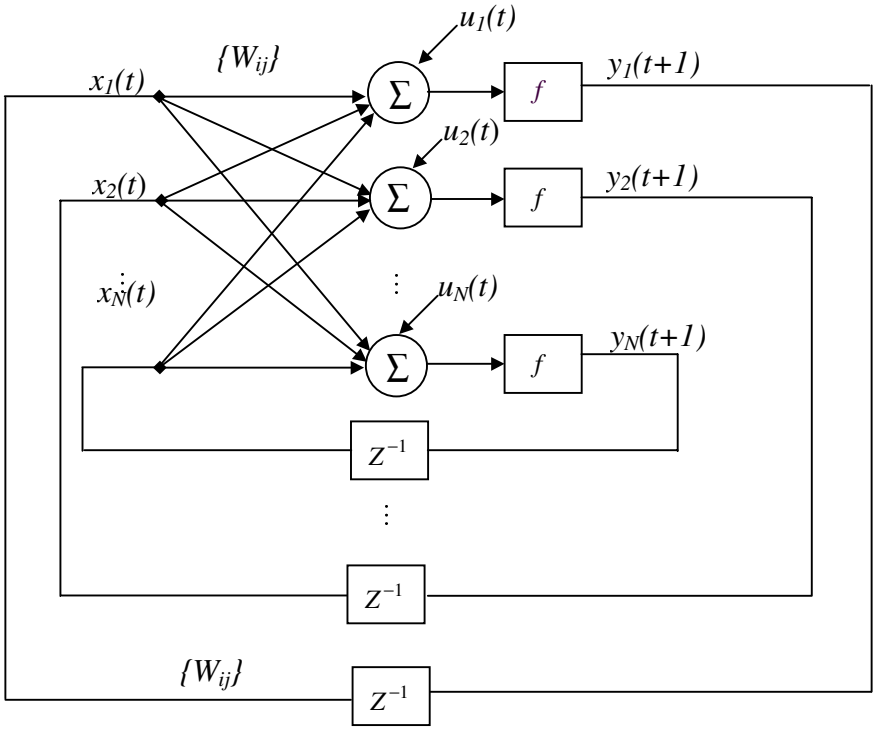


Figura 2.10 – Estrutura de uma rede neural totalmente recorrente (Rede de Hopfield com entradas externas)

2.6.1 Arquiteturas básicas

As redes recorrentes podem ser classificadas como totalmente ou parcialmente recorrentes. Redes totalmente recorrentes (figura 2.10) possuem todo tipo de conexão recorrente, todas ajustáveis. Quando apenas uma parte das possíveis conexões recorrentes é admitida, ou então quando as conexões recorrentes não são ajustáveis, resultam redes parcialmente recorrentes, por exemplo, rede de Elman (figura 2.11), rede com recorrência interna local (figura 2.12), rede com recorrência interna global (figura 2.13) e rede com recorrência externa (figura 2.14). Dentre as arquiteturas de redes neurais recorrentes, a rede de Elman é a mais simples (ELMAN, 1990).

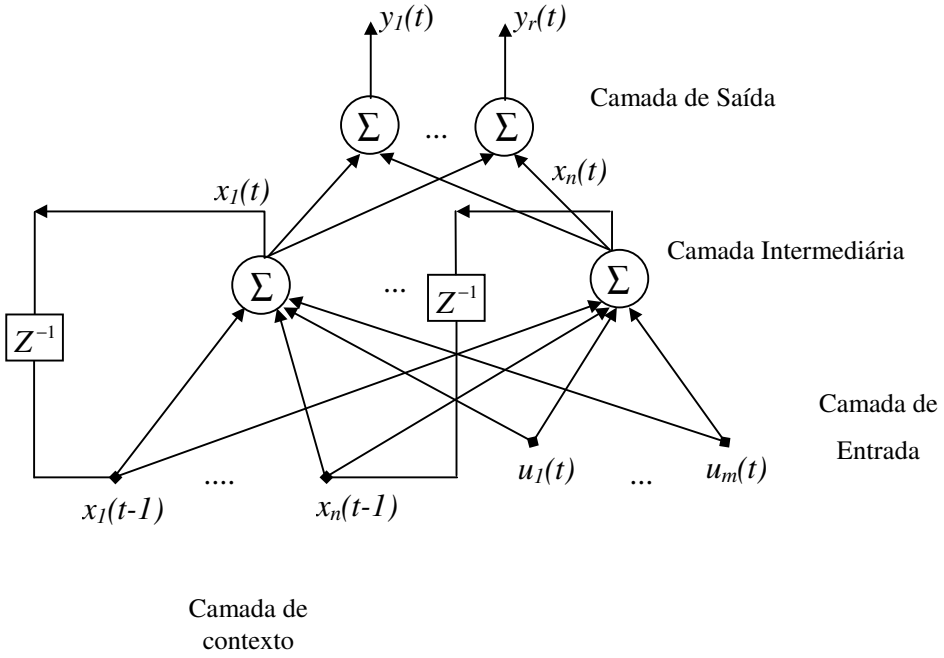


Figura 2.11 - Estrutura de uma Rede de Elman

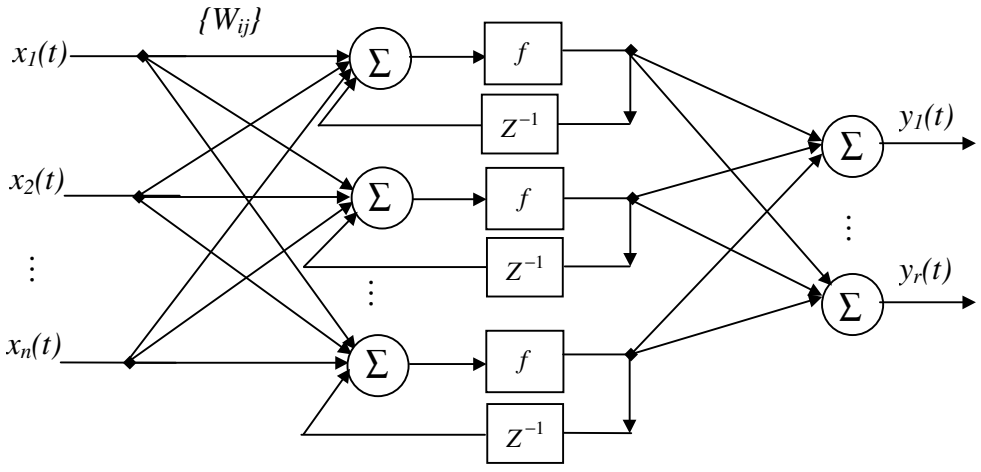


Figura 2.12 - Estrutura de uma rede neural com recorrência interna local

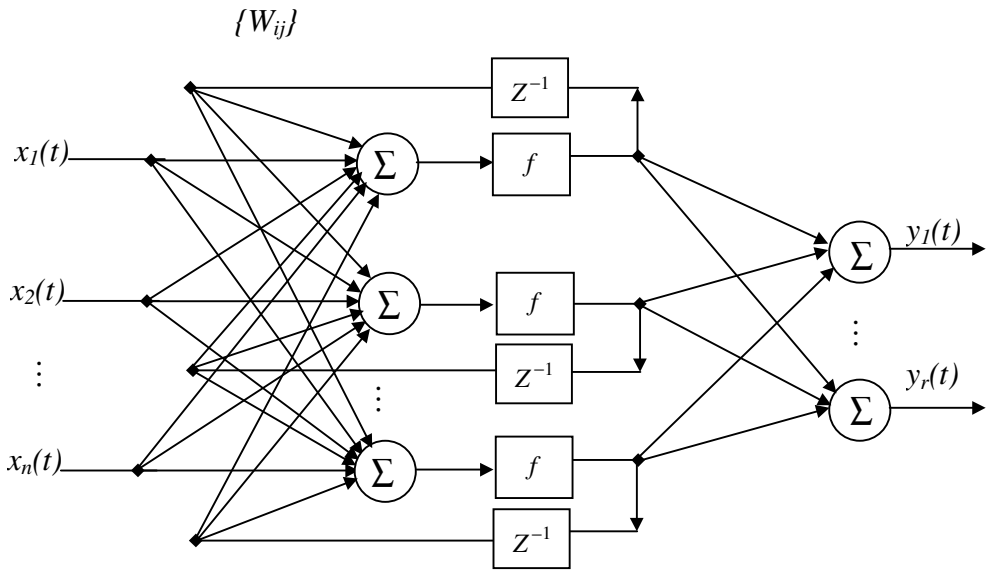


Figura 2.13 - Estrutura de uma rede neural com recorrência interna global

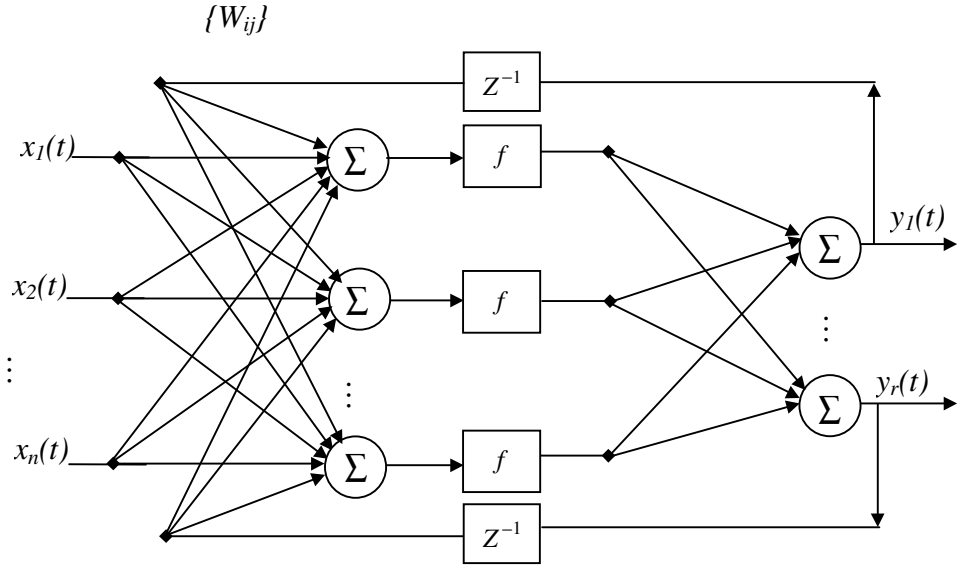


Figura 2.14 - Estrutura de uma rede neural com recorrência externa

2.7 Treinamento supervisionado para redes neurais

A disponibilidade de arquiteturas de rede neurais paramétricas de importância prática está associada à existência de algoritmos de otimização eficientes para realizar os ajuste dos parâmetros no processo de aproximação resultante, denominado treinamento supervisionado.

O algoritmo de aprendizagem possui dois momentos distintos: em primeiro lugar, quando um padrão (ou uma seqüência de padrões de entrada) é apresentado à rede, o fluxo é alimentado para a frente, isto é, propagado adiante até a camada de saída (*forward*). Após este processo, a saída obtida é comparada com a saída desejada, em caso de existir erro, isto é, se a saída desejada não corresponder à obtida, então é feita uma correção nos pesos das conexões sinápticas, ajustando-os na direção oposta à do gradiente do erro instantâneo (este é o instante da aprendizagem). O ajuste é proporcional ao gradiente, segundo um fator de proporcionalidade denominado *taxa de aprendizagem,* que pode ser fixa ou variável. O ajuste dos pesos é feito de trás para frente, isto é, da última camada em direção à camada de entrada (*backward*). Estes dois procedimentos são repetidos até que haja convergência do erro para um valor satisfatório. Neste caso, diz-se que a rede aprendeu o mapeamento, ou que está treinada.

O treinamento de uma rede neural usa, em essência, a regra da cadeia, do cálculo diferencial multivariável, para calcular o modo segundo o qual os peso serão ajustados. Seu objetivo, normalmente, é reduzir o erro quadrático da aproximação feita pela rede, e para tanto emprega o método do passo na direção oposta à do gradiente da superfície de erros, que é função de todos os pesos da rede.

O sinal de erro da saída do neurônio k (da camada de saída) na iteração l (isto é, após a apresentação do l-ésimo exemplo de treinamento) é definido por:

$$e_k(l) = y_k^s(l) - d_k(l) \tag{2.8}$$

onde d_k e y_k^s representam a saída desejada e a saída da rede para o neurônio k, respectivamente. O valor instantâneo do erro quadrático para o neurônio k é definido como:

$$\xi_k(l) = \frac{1}{2}[e_k(l)]^2 \tag{2.9}$$

Da mesma forma, o valor instantâneo do erro quadrático total J é obtido pelo somatório da equação (2.9) para todos os padrões de treinamento e sobre todos os neurônios na camada de saída. Estes são somente os neurônios visíveis para os quais o sinal de erro pode ser calculado diretamente. Podemos então escrever:

$$J(\mathbf{w}) = \sum_{l=1}^{N} J_l(\mathbf{w}) = \sum_{l=1}^{N}\sum_{k=1}^{N_s} \xi_k(l) = \frac{1}{2}\sum_{l=1}^{N}\sum_{k=1}^{N_s}\left[y_k^s(l) - d_k(l)\right]^2 \tag{2.10}$$

onde N_s denota o número de neurônios na camada de saída e N denota o número total de padrões contido no conjunto de treinamento.

O valor do erro quadrático total é função de todos os parâmetros (isto é, pesos sinápticos e entradas de polarização) da rede, denotados aqui por **w**. Para um dado conjunto de treinamento, $J(\mathbf{w})$ representa a função-custo, como medida do desempenho do aprendizado. O objetivo do processo de aprendizado é ajustar os parâmetros livres da rede para minimizar a função-custo, resultando em um problema de otimização irrestrito.

Na próxima seção, são deduzidas as equações para o treinamento supervisionado de uma rede neural estática.

2.7.1 Treinamento supervisionado para redes neurais estáticas

Considere uma rede neural com uma camada intermediária, conforme apresentada na figura 2.2. O vetor de entrada da rede neural é dado por

$$x = [x_1 \cdots x_{N_0}]^T \tag{2.11}$$

o vetor de saída por

$$y = [y_1^s \cdots y_{N_s}^s]^T \tag{2.12}$$

e seu processamento assume a forma:

$$y_k^s(l) = \sum_{j=1}^{N_e} w_{kj}^s y_j^e(l) + w_{k0}^s = \sum_{j=1}^{N_e} w_{kj}^s f\left(u_j^e(l)\right) + w_{k0}^s = \sum_{j=1}^{N_e} w_{kj}^s f\left(\sum_{i=1}^{N_0} w_{ji}^e x_i(l) + w_{j0}^e\right) + w_{k0}^s \tag{2.13}$$

Os resultados apresentados a seguir, embora concentrem-se em redes neurais estáticas com uma camada intermediária, são extensíveis aos casos com múltiplas camadas intermediárias. Caso seja necessário uma rede neural com múltiplas camadas entre a camada de entrada e saída, o procedimento de obtenção das derivadas parciais exigirá um maior número de aplicações da regra da cadeia.

Os pesos serão ajustados de acordo com a regra abaixo

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \alpha \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} \tag{2.14}$$

onde α é taxa de aprendizado.

Para aplicar a equação (2.14) no processo de ajuste dos pesos da rede neural, basta calcular $\partial J/\partial \mathbf{w}$ para todos os peso da rede:

Camada de saída: $w_{kj}^s,\ k = 1,\cdots,N_s;\ j = 0,\cdots,N_e$

$$\begin{aligned}\frac{\partial J}{\partial w_{kj}^s} &= \sum_{l=1}^{N} \frac{\partial J_l}{\partial w_{kj}^s} = \sum_{l=1}^{N} \frac{\partial J_l}{\partial y_k^s(l)} \frac{\partial y_k^s(l)}{\partial w_{kj}^s} = \sum_{l=1}^{N} \frac{\partial J_l}{\partial y_k^s(l)} \frac{\partial y_k^s(l)}{\partial u_k^s(l)} \frac{\partial u_k^s(l)}{\partial w_{kj}^s} = \\ &= \sum_{l=1}^{N} \left(y_k^s(l) - d_k(l)\right) \frac{\partial f}{\partial u_k^s(l)}\, y_j^e(l)\end{aligned} \tag{2.15}$$

onde $y_0^e(l) = 1$, correspondendo à entrada de polarização dos neurônios de saída.

Camada de entrada $w_{ji}^e,\ j = 1,\cdots,N_e;\ i = 0,\cdots,N_0$

Primeiramente, é necessário calcular as derivadas parciais de J_l em relação a $y_j^e(l),\ l = 1,\cdots N;\ j = 1,\cdots,N_e$:

$$\frac{\partial J_l}{\partial y_j^e(l)} = \sum_{k=1}^{N_s} \frac{\partial J_l}{\partial u_k^s(l)} \frac{\partial u_k^s(l)}{\partial y_j^e(l)} = \sum_{k=1}^{N_s} \frac{\partial J_l}{\partial y_k^s(l)} \frac{\partial y_k^s(l)}{\partial u_k^s(l)} \frac{\partial u_k^s(l)}{\partial y_j^e(l)} = \sum_{k=1}^{N_s} \left(y_k^s(l) - d_k(l)\right) \frac{\partial f}{\partial u_k^s(l)}\, w_{kj}^s \tag{2.16}$$

Então obtém-se

$$\frac{\partial J}{\partial w_{ji}^e} = \sum_{l=1}^{N} \frac{\partial J_l}{\partial w_{ji}^e} = \sum_{l=1}^{N} \frac{\partial J_l}{\partial u_j^e(l)} \frac{\partial u_j^e(l)}{\partial w_{ji}^e} = \sum_{l=1}^{N} \frac{\partial J_l}{\partial y_j^e(l)} \frac{\partial y_j^e(l)}{\partial u_j^e(l)} \frac{\partial u_j^e(l)}{\partial w_{ji}^e} = \\ = \sum_{l=1}^{N} \left[\sum_{k=1}^{N_s} \left(y_k^s(l) - d_k(l) \right) \frac{\partial f}{\partial u_k^s(l)} w_{kj}^s \right] \frac{\partial f}{\partial u_j^e(l)} x_i \tag{2.17}$$

onde $x_0 = 1$, correspondendo à entrada de polarização.

2.8 O cálculo do produto da matriz hessiana por um vetor

No desenvolvimento de métodos de treinamento que utilizam informação de 2ª ordem, de acordo com os resultados apresentados por VON ZUBEN (1996), uma propriedade do método do gradiente conjugado é a ausência da necessidade de se determinar e armazenar a matriz hessiana, elemento a elemento. Procedimentos para o cálculo exato da matriz hessiana, elemento a elemento, são apresentados por BISHOP (1992). É suficiente determinar e armazenar o produto da matriz hessiana por um vetor conhecido. Portanto, dado um vetor arbitrário $v \in \Re^p$, o cálculo de $\nabla^2 J(\theta)v$ requer a aplicação do operador diferencial $\Psi_v\{.\}$ (definido em PEARLMUTTER, 1994) a toda operação realizada para obter $\nabla J(\theta)$, pois $\Psi_v\{\nabla J(\theta)\}=\nabla^2 J(\theta)v$.

Assim, com a aplicação do operador $\Psi_v\{.\}$ às equações (2.13) e (2.15), obtém-se exatamente a informação de 2ª ordem necessária para possibilitar a aplicação do algoritmo do gradiente conjugado híbrido (DOS SANTOS & VON ZUBEN, 2000), que será empregado nos problemas de treinamento supervisionado considerados neste trabalho.

2.9 Capacidade de generalização

Existe uma concepção errônea e perigosa relativa ao treinamento iterativo. Ela se apóia na idéia de que deve-se treinar a rede neural para reduzir mais e mais o erro junto ao conjunto de treinamento de modo a aumentar o desempenho da rede neural. Embora, parcialmente verdadeira, esta iniciativa pode conduzir à especialização excessiva junto ao conjunto de treinamento, o qual pode conter amostras sujeitas a ruído, que passariam a ser incorporados à solução, e certamente representa um conjunto finito de informação. A figura

2.15 apresenta um esboço do erro de uma rede neural para o conjunto de treinamento, e para um conjunto de teste (ambos gerados a partir do mesmo conjunto de dados, mas independentes entre si). Respostas com esta conformação geralmente ocorrem em treinamento de redes neurais (GERMAN *et al*., 1992). Como desejado, o erro para o conjunto de treinamento decresce continuamente (com exceção de pequenas oscilações ocasionais). Para o conjunto de teste, ele decresce no princípio, embora o erro se mantenha superior ao do conjunto de treinamento. Continuando o treinamento, o erro referente ao conjunto de teste começa a crescer. Desde que o conjunto de teste (ou validação) é presumivelmente representativo do problema para o qual a rede deverá representar a solução, isto significa que a rede está perdendo capacidade de generalização. Assim, o ideal é acompanhar o erro de treinamento, mas conforme o número de iterações cresce, também deve-se procurar acompanhar o erro de teste, que avalia o desempenho da rede (um tipo de validação cruzada). Deve-se parar o treinamento quando o mínimo do erro frente ao conjunto de teste foi atingido ou esteja dentro de um ε preestabelecido, para garantir melhor capacidade de generalização.

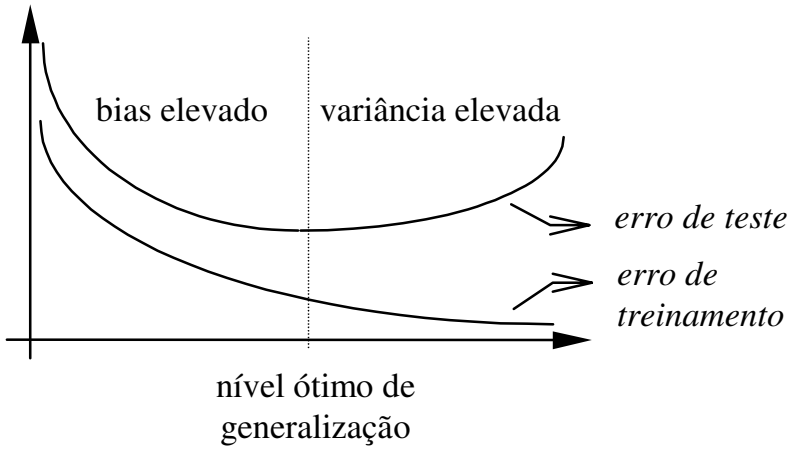


Figura 2.15 – Esboço do comportamento típico de erros de treinamento e teste

2.9.1 Treinamento de uma rede neural com validação cruzada

O objetivo é encontrar uma estimação para os pesos sinápticos (**w**) da rede neural que permita minimizar a função-custo. O conceito de validação cruzada pode ser basicamente descrito na forma: depois da estimação de **w** usando um conjunto de dados de treinamento, a qualidade do mapeamento é avaliada usando um conjunto de dados de teste. O melhor mapeamento é definido como aquele que minimiza o erro de predição junto ao conjunto de dados de teste.

A técnica de validação cruzada pode ser descrita na forma:

1. Separe o conjunto de dados disponíveis para treinamento (N) em dois conjuntos, um conjunto de treinamento $\{x_i,\ y_i,\ i=1,...,N_{tr}\}$ e um conjunto de teste $\{x_i, y_i, i= 1,...,N_{ts}\}$, com $N = N_{tr} + N_{ts}$, de modo que ambos os conjuntos sejam representativos do mapeamento que se quer aproximar;
2. Construa um mapeamento usando o conjunto de treinamento;
3. Avalie o mapeamento usando o conjunto de teste;
4. Repita os passos 2 e 3 até obter um mapeamento que minimiza algum critério aplicado ao conjunto de teste.

A técnica de validação cruzada é usada para modificar o processo de aprendizagem nos algoritmos de treinamento, de tal forma a encontrar uma melhor representação do sistema durante sua identificação na presença de dados com ruído. Em situações reais, os dados disponíveis de um processo, geralmente, contêm ruído. Quando estes dados são usados para construir um modelo do processo a partir de uma rede neural submetida a treinamento supervisionado, a validação cruzada é usada para controlar a quantidade de ruído que é incorporada ao comportamento da rede neural treinada, por exemplo, limitando o número de vezes que o conjunto de treinamento é apresentado à rede neural durante o ciclo de aprendizagem. O treinamento da rede neural é interrompido antes que o mínimo da função-custo seja alcançado junto ao conjunto de treinamento (veja figura 2.15). A motivação para a adoção desta abordagem é a obtenção da melhor representação do modelo atual do sistema na presença de dados ruidosos, ou quando a flexibilidade da rede neural é maior que a flexibilidade do mapeamento a ser aproximado.

2.10 Visão sintética do capítulo

Neste capítulo, apresentamos uma revisão de alguns conceitos importantes e aspectos de implementação relacionados a redes neurais artificiais. Além disso, descrevemos os vários tipos de redes estáticas e dinâmicas encontrados na literatura e os diversos algoritmos de treinamento, dando ênfase àqueles que serão empregados mais adiante nesta dissertação.

Capítulo 3

Identificação de sistemas dinâmicos não-lineares utilizando redes neurais artificiais

3.1 Introdução

A idéia geral no projeto de controladores é encontrar um controlador que modifica um dado comportamento de um sistema dinâmico (planta) com a finalidade de alcançar alguns objetivos. Para tanto, o controlador deverá gerar a entrada $\boldsymbol{u}$ do sistema dinâmico, responsável pela produção do comportamento desejado.

Embora a estratégia de controle de uma dada planta possa ser obtida diretamente, muitos dos mais populares métodos de controle, baseados na idéia de programação dinâmica, requerem que a planta seja identificada, isto é, existe a necessidade de um modelo da planta capaz de predizer as saídas futuras para sinais de teste, permitindo desta forma ajustar os parâmetros do controlador.

Nosso objetivo é encontrar controladores para plantas com dinâmicas desconhecidas e possivelmente não-lineares. Assim, a tarefa de construção de um controlador para uma planta desconhecida pode ser dividida em duas partes:

- modelagem da planta desconhecida, empregando técnicas de identificação de sistemas;
- construção de um controlador usando uma função-custo definida (por exemplo, erro quadrático médio) baseada no modelo da planta.

Identificação de sistemas é um dos problemas básicos em teoria de controle. No caso de plantas lineares, uma abordagem para identificação é construir uma função de transferência representando o comportamento da planta, em tempo discreto ou contínuo, usando o princípio da superposição, sendo que o estado inicial é suposto ser zero. A identificação de plantas não-lineares é mais difícil, pois o princípio da superposição não pode ser usado, e a relação entrada-saída pode depender do estado atual e/ou da história da planta. Além disso, a planta pode ter muitos estados para os quais a saída é constante ou zero.

Apesar destas dificuldades, identificação de sistemas não-lineares pode ser realizada de duas formas básicas: analiticamente e computacionalmente, sendo que estas podem ser usadas separadamente ou combinadas.

Identificação analítica de sistemas envolve a análise da dinâmica do sistema físico e o desenvolvimento de um modelo matemático deste sistema. Um simples exemplo desta técnica pode ser ilustrado através do problema de controle do nível de água em um recipiente cilíndrico. O nível de água e a razão do escoamento de água para fora do recipiente em um instante de tempo determinam o nível de água no próximo instante. Tal sistema é chamado integrador e se $y(k)$ e $u(k)$ são medidas exatas do nível e fluxo de água para um instante k, então este sistema pode ser modelado pela seguinte equação:

$$y(k+1) = f[y(k), u(k)] = y(k) + au(k) \tag{3.1}$$

onde a é uma constante negativa que depende do raio do cilindro e do período de amostragem. Desde que a pode ser determinado por uma inspeção física do sistema, o modelo preciso da planta pode ser identificado por *métodos puramente analíticos ou paramétricos*.

Identificação computacional de sistema envolve coleta estatística das características de entrada-saída da planta e sua utilização na determinação de um modelo que aproxime este comportamento observado. Esta tarefa pode ser representada pelos quatro passos básicos a seguir:

- *Planejamento experimental* – determinação de como os dados serão coletados, isto é, qual o método de amostragem a ser utilizado;
- *Seleção da estrutura do modelo* – seleção da estrutura do modelo e posterior determinação dos parâmetros passíveis de serem ajustados, chamados parâmetros livres;

- *Estimação de parâmetros* – ajuste dos parâmetros livres usando as estatísticas obtidas dos dados;
- *Validação* – avaliação do desempenho do modelo para os dados de teste.

Na prática, este procedimento deve usualmente ser repetido algumas vezes até que o modelo seja validado satisfatoriamente. Todos estes passos, exceto talvez o passo de estimação de parâmetros, envolve alguma análise do sistema. A classificação entre identificação analítica e computacional de sistema é, portanto, uma tarefa difícil de realizar e que requer a participação do projetista em muitas de suas etapas.

Por exemplo, quando um modelo *NARX* (auto-regressivo, não-linear e com entradas externas) é utilizado, a tarefa de identificação – tanto analítica como computacional – é análoga a um problema de aproximação de funções, em que a planta realiza um mapeamento desconhecido de entrada-saída. Redes Neurais Artificiais (RNA), como apresentado em VON ZUBEN (1996), são ferramentas exímias para aproximação de funções, de modo que este paradigma pode ser usado para identificação de sistemas.

Um sistema dinâmico ou um processo pode ser descrito matematicamente como um operador $S \in C$, onde C é uma classe definida sobre um espaço apropriado de entrada e saída. O problema de identificação consiste em escolher uma classe de modelos $\overline{C} \subset C$ e selecionando um elemento $\overline{S} \in \overline{C}$ que aproxime o operador $S \in C$ segundo algum critério. Para esta aproximação ser arbitrária, $\overline{C}$ precisa ser denso em C. Para tratabilidade matemática, $\overline{C}$ é geralmente uma classe convenientemente parametrizada com um número finito de parâmetros arranjados no vetor $p \in \Re^m$, onde m é a dimensão de p. A cada valor do vetor de parâmetros p corresponde um $\overline{S} \in \overline{C}$, e como p se estende sobre todo o $\Re^m$, ele gera a classe $\overline{C}$. O problema de identificação pode, portanto, ser colocado como o problema de determinar um vetor de parâmetros tal que a saída da planta y e a saída do modelo $\hat{y}$ estejam próximas segundo algum critério adotado: isto é, $\|y - \hat{y}\| < \varepsilon$ para toda entrada de uma dada classe, onde $\|\cdot\|$ é uma norma convenientemente definida (NARENDRA & MUKHOPADHYAY, 1996).

Representações de sistemas não-lineares usando séries funcionais são encontradas nos trabalhos clássicos de Volterra e Wiener (SCHETZEN, 1981). Em tais representações, a saída é expressa em termos da entrada como:

$$y(k+1) = F[u(k), u(k-1), \cdots] \tag{3.2}$$

Embora seja uma forma matematicamente elegante, tais representações têm encontrado muito pouca aplicação em identificação. As razões são várias, mas de acordo com NARENDRA & MUKHOPADHYAY (1996) estão relacionadas a dificuldades práticas. Estas dificuldades são devidas principalmente à grande quantidade de parâmetros necessários para representar mesmo sistemas simples e à necessidade de entradas especiais para o sistema, de forma a simplificar o problema de estimação de parâmetros.

Em vista destas dificuldades, é preferível caracterizar o sistema em termos de equações a diferenças não-lineares de dimensão finita. Dependendo das informações a priori disponíveis, diferentes modelos não-lineares podem ser escolhidos se convenientemente parametrizados. Redes neurais artificiais correspondem a uma classe conveniente de tais modelos. A abordagem baseada em redes neurais para identificação tem sido utilizada, principalmente, quando:

- há pouco conhecimento sobre a planta;
- a estrutura da planta apresenta não-linearidades significativas;
- somente dados de entrada-saída estão disponíveis.

No entanto, alguns requisitos são necessários para a aplicação de redes neurais em identificação, como a disponibilidade de dados amostrados de toda a região de operação do sistema, no espaço de estados.

Uma questão que surge na utilização de redes neurais em identificação é a necessidade de se arbitrar a dimensão, ou o número de neurônios na camada escondida. A abordagem construtiva, baseada na teoria de agentes e a ser proposta no capítulo 6, além de conseguir eliminar tal necessidade apresenta uma arquitetura mais compacta e um desempenho superior.

Neste capítulo, apresentamos uma revisão das técnicas de identificação de sistemas utilizadas na literatura, e sua extensão para o emprego de redes neurais artificiais.

3.2 Arquitetura de identificação utilizando rede neurais

A tarefa de identificação utilizando rede neurais pode ser realizada através de redes neurais estáticas ou dinâmicas. Um modelo que descreve o comportamento do sistema é útil para projetar controladores para sistema dinâmicos. Desde que um modelo ótimo para a planta real tenha sido encontrado, simulações computacionais podem ser realizadas e outros aspectos de projeto podem ser contemplados.

Algumas suposições sobre a planta a ser identificada são consideradas:

- a planta pode incluir dinâmica não-linear e/ou linear;
- somente dados de entrada e saída estão disponíveis;
- a planta pode ser um sistema invariante ou variante no tempo.

Uma vez que não será exigido um conhecimento a priori da planta para identificação do sistema, métodos puramente paramétricos ou analíticos não são usados, uma vez que não podemos predizer a ordem do sistema ou a forma de qualquer modelo puramente paramétrico. Somente serão consideradas medidas dos pares de entrada-saída, tomadas da planta original para construir o modelo. Os valores *u(t)* e *y(t)* representam sinais vetoriais correspondendo à entrada e à saída medida da planta, respectivamente. É tarefa do modelo, no caso da rede neural, construir uma representação para a planta usando pares de entrada-saída tomados da planta real. Esta situação é mostrada na figura 3.1, onde desejamos minimizar $\|\hat{y}(k) - y(k)\|$.

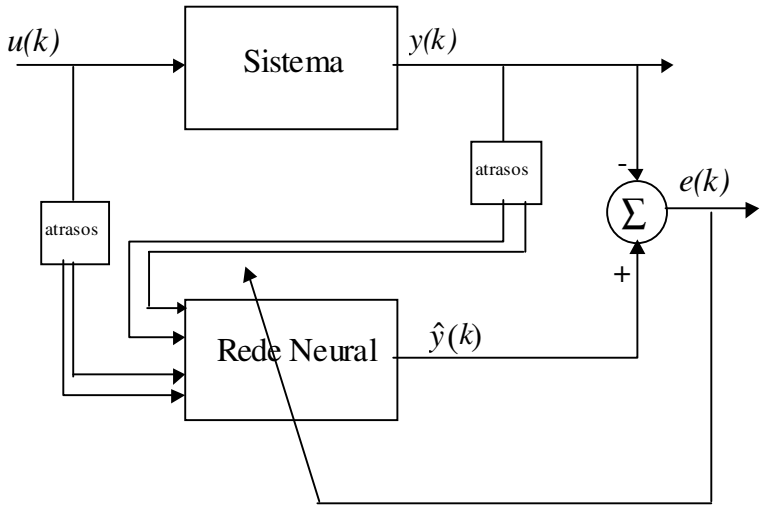


Figura 3.1 – Estrutura de identificação de sistemas utilizando redes neurais

A estrutura da rede neural para identificação de sistemas pode ser uma rede neural multicamadas. O processo de aprendizagem ou ajuste de parâmetros a partir dos dados de entrada-saída disponíveis é feito por minimização da função-custo dada por:

$$J = \frac{1}{N}\sum_{k=1}^{N}\sum_{i=1}^{N_s}(\hat{y}_i(k) - y_i(k))^2 \tag{3.3}$$

onde N é o número de dados e N_s é número de saídas. Observe que estamos considerando aqui o caso de sistemas de tempo discreto.

Os pares de vetores $u(k)$ e $y(k)$ são padrões apresentados para a rede neural. O sinal de erro é calculado e os pesos da rede são ajustados de tal forma que $\hat{y}(k)$ aproxime $y(k)$ para todo $u(k)$. O treinamento é finalizado quando o erro de estimação é significativamente pequeno e a rede neural obtida pode então ser usada como um modelo da planta real.

3.3 Modelagem de tempo discreto

Sinais provenientes de sistemas físicos, cuja dependência de outros sinais não é conhecida, podem somente ser representados por uma função no tempo. Entretanto, se um sinal $y(k)$ depende somente de outro sinal $u(k)$, então é possível representá-lo como uma função $f[.]$, daquele sinal, isto é:

$$y(k) = f[u(k)] \tag{3.4}$$

Um sistema real que pode ser descrito pela expressão (3.4) é chamado um *sistema estático,* visto que não possui nenhuma dependência explícitas do tempo k, ou de valores passados das variáveis envolvidas. Outros sistemas têm memória, e assim têm saídas que dependem de entradas anteriores, além da entrada corrente. Tais sistemas são chamados sistemas dinâmicos. Se tais sistemas satisfazem a condição de observabilidade (veja seção 3.5.2), então eles podem ser modelados por um modelo *NARX* descrito genericamente como segue:

$$y(k+1) = f[y(k),\cdots,y(k-p+1),u(k),\cdots,u(k-q+1)], \tag{3.5}$$

onde p e q são valores inteiros descrevendo o número de exemplos anteriores de sinais u e y que são necessários para predizer a próxima saída.

No caso geral, em que há um atraso de períodos de amostragens $d \geq 1$ entre o tempo em que um sinal $u(k)$ é aplicado na entrada da planta e o tempo em que ele poderá afetar a saída da planta, então a saída da planta pode ser descrita por:

$$y(k+1) = f[y(k),\cdots, y(k-p+1), u(k-d+1),\cdots, u(k-d-q+1)], \tag{3.6}$$

Modelos *NARX* tais como (3.5) e (3.6) podem ser utilizados para modelar uma planta em tempo discreto.

3.4 Tipo de representação de sistema dinâmicos

Ao contrário dos métodos de modelagem linear, nos métodos de modelagem não-linear não há uma equivalência entre modelos de entrada-saída e espaço de estado. Mesmo quando existe um modelo equivalente de entrada-saída para um dado modelo de espaço de estados, sua função de transferência pode necessitar de muito mais argumentos que as funções de transição de estado e de saída para o modelo de espaço de estados correspondente.

3.4.1 Representação usando o espaço de estados

A representação de múltiplas entradas e múltiplas saídas (*MIMO – multi-input multi-output*) de um sistema dinâmico, usando equações de estados, pode ser escrita da seguinte forma:

$$\begin{aligned} x(k+1) &= f[x(k), u(k)] \\ y(k) &= h[x(k)] \end{aligned} \tag{3.7}$$

onde $u(k)$, $x(k)$ e $y(k)$ são vetores de variáveis de tempo discreto, $x(k) = [x_1(k), x_2(k),..., x_n(k)]^T$, $u(k) = [u_1(k), u_2(k),..., u_p(k)]^T$ e $y(k) = [y_1(k), y_2(k),..., y_m(k)]^T$, com $u(k)$ representando a entrada, $x(k)$ o estado e $y(k)$ o vetor de saída para o instante de tempo k. Portanto, a equação representa um sistema de ordem n, com p entradas e m saídas. Se o sistema descrito pela equação (3.7) é suposto linear e invariante no tempo, as equações que governam seu comportamento podem ser expressas por:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{3.8}$$

onde A, B, e C são respectivamente matrizes $(n \times n)$, $(n \times p)$ e $(m \times n)$. O sistema é parametrizado pela tripla $\{A, B, C\}$. A teoria de sistemas lineares invariantes no tempo, quando as matrizes A, B, e C são conhecidas, é bem desenvolvida, e tem sido extensivamente empregada na produção de representação aproximada para sistemas dinâmicos de interesse prático.

Para um sistema linear descrito pela equação (3.8), a saída para o instante de tempo $k+n$ é dada por:

$$y(k+n) = CA^n x(k) + \sum_{l=1}^{n-1} CA^{n-l-1} Bu(l+k) \tag{3.9}$$

Portanto, se o estado do sistema para o instante k é conhecido, é possível determinar o comportamento do sistema para qualquer seqüência de entrada conhecida, e $y(k+n)$ pode ser denotado por:

$$y(k+n) = F\left[x(k), u(k+1), \cdots, u(k+n-2), u(k+n-1)\right] \tag{3.10}$$

onde F é chamado mapa de entrada-saída ou função-resposta do sistema. Se as entradas e as saídas são especificadas e a matriz $[C, CA, ..., CA^{n-1}]$ é de posto n, isto é, tem posto completo, o estado pode ser determinado através da equação (3.9). Então, o sistema é dito ser observável.

3.4.2 Representação usando entrada-saída

A abordagem de entrada-saída para redes neurais foi introduzida e ilustrada por NARENDRA & PARTASARATHY (1990), NARENDRA & PARTASARATHY (1991), NERRAND *et al*. (1994). Desde então, procedimentos de projeto para estimação de modelos usando dados de entrada-saída e procedimentos de síntese para seu controle têm sido realizados com sucesso.

Para sistema lineares controláveis e observáveis de ordem n, segue que o estado inicial do sistema pode ser computado a partir de n valores de entrada e saída e, portanto, qualquer sistema com uma entrada e uma saída (*SISO – single input single output*) pode ser descrito pela seguinte equação:

$$y(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) + \sum_{j=0}^{n-1} \beta_j u(k-j) \tag{3.11}$$

onde $\alpha_i \, e \, \beta_j \; (i, j = 0,1,\cdots,n-1)$ são parâmetros constantes. Sistemas lineares *MIMO*, onde $y(k)$ e $u(k)$ são de dimensão m e p, respectivamente, podem ser descritos por:

$$y(k+1) = \sum_{i=0}^{n-1} A_i y(k-i) + \sum_{j=0}^{n-1} B_j u(k-j) \tag{3.12}$$

onde A e B são matrizes $(m \times m)$ e $(m \times p)$, respectivamente.

As equações (3.11) e (3.12) representam o conhecido modelo *ARMA* para sistemas lineares *SISO* e *MIMO* invariantes no tempo, respectivamente.

Portanto, para sistemas lineares invariante no tempo dados pela representação por espaço de estados da equação (3.8), existe uma representação entrada-saída dada pela equação (3.12). Alternativamente, dada a representação de entrada-saída da equação (3.12), existe uma representação por espaço de estados de dimensão finita na forma de equação (3.8), que tem a mesma resposta entrada-saída dada pela equação (3.12). Isto implica numa função de transferência racional. Portanto, para sistemas lineares invariantes no tempo de dimensão finita, a representação por espaço de estados e uma descrição de entrada-saída que contém um número finito de valores passados de entradas e saídas são formas equivalente de descrição de um sistema.

A relação entre a descrição de entrada-saída e a representação de estados é, entretanto, consideravelmente mais complexa para sistemas não-lineares. Equações de entrada-saída são de importância fundamental na teoria de sistemas dinâmicos, porque elas descrevem o comportamento do sistema sob o ponto de vista de um observador externo. Por outro lado, muitos dos desenvolvimentos recentes da teoria de sistemas dinâmicos usam representação por espaço de estados, já que elas permitem a aplicação de técnicas de equações diferenciais e teoria de otimização. Assim, uma questão básica é decidir quando um dado operador de entrada-saída admite representação na forma de espaço de estados. Este é o domínio da teoria de realização, e se a descrição de espaço de estados existe para um dado operador de entrada-saída, então o operador é dito ser realizável. Uma questão que surge na relação entre equação de estados e equações de entrada-saída pode ser formulada

como segue: dado um sistema *S* representado pela equação (3.7), pode-se produzir uma equação de entrada-saída para o sistema?

Como mencionado anteriormente, em sistemas lineares a existência de equações de entrada-saída equivalentes está relacionada à habilidade de determinar os estados do sistema através de um número finito de medidas de entrada-saída. Para sistemas não-lineares, NARENDRA (1992) propõe procedimentos similares aos discutidos para sistemas lineares.

Considerando o intervalo [*k, k*+*m*-1], a saída do sistema dinâmico dado pela equação (3.7), pode ser expressa como:

$$\begin{aligned}
&y(k) = h[x(k)] \\
&y(k+1) = h[f[x(k), u(k)]] \\
&y(k+2) = h[f[f[(x(k), u(k), u(k+1)]]] \\
&\quad\vdots \quad\vdots \\
&y(k+m-1) = h[f[f[\cdots f[f[x(k), u(k)], u(k+1)]\cdots, u(k+m-3)], u(k+m-2)]
\end{aligned} \tag{3.13}$$

Se a equação (3.13) puder ser solucionada para $x(k)$ em termos de valores de saída e entrada, segue-se que uma equação recursiva de entrada-saída também pode ser derivada para o sistema não-linear. Ou seja, se o sistema for observável para um tempo finito, então existe uma equação recursiva de entrada-saída. Suposições rigorosas sobre as funções *f* e *h* têm sido feitas no caso não-linear para garantir a existência de uma solução. Além disso, para existência de tais soluções, o estado inicial do sistema deve estar restrito a uma dada região do espaço de estados. Para sistemas não-lineares, uma noção de observabilidade hierárquica é possível. Estas incluem observabilidade, observabilidade de simples experimentos, observabilidade genérica de simples experimentos, observabilidade forte e observabilidade genérica global. Observabilidade forte supõe a existência de um inteiro *K* tal que qualquer seqüência de entrada-saída de comprimento maior ou igual a *K* irá determinar unicamente o estado inicial da equação (3.13), enquanto observabilidade genérica assegura o mesmo para quase todas as seqüências de comprimento suficiente. Uma vez que iremos trabalhar com plantas na qual as funções *f* e *h* são supostas serem desconhecidas, todos os sistemas que são considerados no capítulo 7 devem ser observáveis na região que contém o ponto de operação.

3.5 Identificação linear e identificação não-linear usando redes neurais

A identificação de um sistema dinâmico usando paradigmas tradicionais, tais como modelo auto-regressivo com entradas externas (*ARX*), modelo média móvel (*MA*), estruturas de resposta ao impulso finita (*FIR*), modelo autoregressivo com média móvel e entradas externas (*ARMAX*) e modelo de erro de saída (*OE*) é comumente baseada em algumas suposições sobre o sistema verdadeiro. As mais importantes são linearidade e invariância no tempo. A diferença básica entre os modelos de identificação *ARX, MA, ARMAX* e *OE* está na forma de modelagem do ruído.

Os modelos *OE* e *ARMAX* requerem a suposição adicional de que a ordem do sistema possa ser identificada, e o modelo *FIR* supõe que o usuário possa estimar o tempo de atraso do processo na definição do passo da entrada. Estas suposições permitem que um experimento local seja extrapolado globalmente para outros pontos de operação. A validade da extrapolação depende, é claro, da validade das suposições levantadas. A experiência tem mostrado que, para um número significativo de sistemas, estas suposições são razoáveis. Se o sistema não é invariante no tempo, processos de identificação recursiva podem ser necessários. Se o sistema é não-linear com respeito às condições de operação, vários modelos lineares são necessários para cobrir cada ponto de operação do sistema.

As suposições feitas em identificação utilizando redes neurais são consideravelmente menos rigorosas que aquelas feitas em identificação clássica. Para a tecnologia de identificação clássica, o usuário deve especificar a natureza do relacionamento entre as entradas e saídas, enquanto para identificação utilizando redes neurais o usuário deve somente especificar a topologia da rede que é suficiente para descrever o mapeamento de entrada-saída. Esta suposição para redes neurais pode ser suprimida considerando a estratégia construtiva baseada na teoria de agentes, a ser descrita no capítulo 6. A rede neural não requer nenhum conhecimento prévio dos relacionamentos funcionais entre as variáveis de entrada e saída. Ao invés disso, a rede neural simplesmente gera um mapeamento da entrada-saída a partir dos exemplos utilizados no treinamento.

Portanto, a primeira vantagem da abordagem utilizando redes neurais, comparada com a identificação clássica de sistemas dinâmicos, é que evitamos a especificação da

estrutura do modelo, a qual pode ser muito difícil. Isto fica evidente quando a estrutura do sistema real não pertence ao conjunto de modelos que podem ser gerados pelo método selecionado para identificação. Os modelos *ARX, MA, ARMAX, OE* requerem experimentos cuidadosamente controlados sobre o intervalo de operação do processo. Sendo assim, para sistemas não-lineares a serem identificados usando esses modelos lineares, o conjunto de dados deve ser separado em subconjuntos onde o processo é aproximadamente linear. Por outro lado, redes neurais permitem que o mapeamento de entrada-saída seja detectado sem qualquer entendimento do relacionamento funcional entre entrada e saída. A desvantagem desta técnica é a necessidade de especificar dados de entrada para a rede neural que cubra toda a região de interesse: sem conhecimento da estrutura do sistema real, os resultados dos experimentos não podem ser extrapolados para fora da região onde o experimento foi conduzido.

Portanto, identificação utilizando redes neurais requer pouca intervenção do usuário na definição dos relacionamentos estruturais entre as entradas e saídas do processo. Entretanto, experimentos de identificação utilizando redes neurais podem necessitar de um conjunto de dados (pares de entrada-saída) maior - talvez em ordens de magnitude - em relação ao experimento de identificação clássica.

Nas seções seguintes, modelos de resposta ao impulso finita não-linear, média móvel (*MA*), auto-regressivo com entradas externas não-linear (*NARX*), auto regressivo com média móvel não-linear e com entradas externas (*NARMAX*), não-linear de erro de saída (*NOE*), modelos por espaço de estados e sua extensão utilizando redes neurais são apresentados para representar sistemas não-lineares.

3.5.1 Modelo de resposta ao impulso finita não-linear (NFIR)

Alguns dos trabalhos clássicos de Volterra e Weiner (SCHETZEN, 1981) tratam da caracterização de funções não-lineares. Usando o teorema de Stone-Weiestrass, pode ser mostrado que, dentro de certas condições, qualquer funcional f não-linear pode ser representado por uma série de Volterra ou uma série de Weiner. No caso de sistemas dinâmicos lineares invariantes no tempo, é conhecido que a saída $y(k+1)$ pode ser realizada através da entrada u (responsável pela excitação do sistema) pela seguinte equação:

$$y(k+1) = \sum_{i=0}^{\infty} \alpha_i u(k-i) \tag{3.14}$$

onde α_i representa a resposta impulsiva do sistema para o instante de tempo $k–i$ devido a um pulso unitário neste instante.

A série de Volterra e a série de Weiner podem ser consideradas como generalizações de tal representação para o caso não-linear. Em ambos os casos, qualquer sistema dinâmico é descrito por um sistema dinâmico linear invariante no tempo seguido por transformações não-lineares. A saída de um dado sistema não-linear é obtida pela combinação linear de transformações não-lineares da entrada. Por exemplo, a série discreta de Volterra tem a seguinte forma:

$$y(k+1) = \alpha_0 + \sum_{i=0}^{\infty} \alpha_i u(k-i) + \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \alpha_{ij} u(k-i)u(k-j) + \cdots \tag{3.15}$$

onde α_0, α_i (i = 1, 2, ..., ∞), α_{ij} (i = 1, 2, ..., ∞; j = 1, 2, ..., ∞) são parâmetros a determinar. A expressão com duplo somatório representa uma combinação linear dos valores passados da entrada, tomados em produtos dois a dois.

Visto que uma rede neural multicamada pode ser usada para aproximar arbitrariamente bem qualquer função contínua sobre um conjunto compacto, a estrutura de processamento mostrada na figura 3.2 pode ser usada para representar qualquer sistema não-linear possível de ser modelado por uma série de Volterra, desde que sejam fornecidos todos os valores passados de entrada para a rede neural.

Para propósito de identificação, um modelo representado pela equação (3.15) pode ser usado com um número m finito de entradas, onde m é escolhido para atender a um grau de precisão desejado, tal que:

$$\hat{y}(k+1) = f[u(k), u(k-1), \cdots u(k-m)] \tag{3.16}$$

A equação (3.16) é uma versão não-linear do modelo de resposta ao impulso finita usado em sistemas lineares e m corresponde a um valor a partir do qual as entradas $u(k–m–i)$, i = 1, 2, ..., são supostas serem insignificantes na determinação de $\hat{y}(k+1)$.

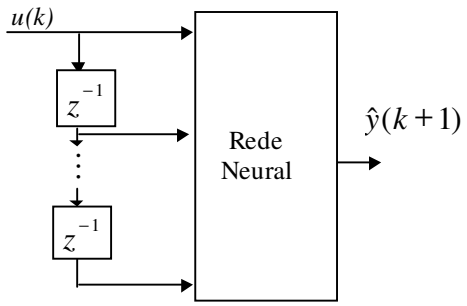


Figura 3.2 – Modelo neural para resposta ao impulso finita não-linear (NFIR)

Alguns pesquisadores da área de controle, tais como NARENDRA & PARTHASARATHY (1990), adotam uma outra nomenclatura para a combinação linear de sinais de controle, chamada de modelo de média móvel, causando assim ausência de homogeneidade de notação e, conseqüentemente, problemas de interpretação.

O modelo *NFIR* apresenta as seguintes características:

- tem a vantagem de produzir um modelo sempre estável, pelo fato de supor que a dinâmica do sistema depende apenas de um número finito de entradas externas passadas;
- tem a desvantagem de requerer uma dimensão n+1 elevada para o vetor de entrada u no caso de dinâmicas com resposta lenta. Ou seja, se o tempo máximo de resposta a qualquer mudança na entrada externa é T e o período de amostragem é δ, então a dimensão de u deve ser da ordem de T/δ, que pode ser um valor elevado (SJÖBERG *et al*., 1995).

3.5.2 Modelo de média móvel (MA)

Este modelo considera que a saída do sistema dinâmico pode ser representada por uma combinação linear finita de valores passados de um processo estocástico do tipo ruído branco $w(k)$, como segue:

$$y(k+1) = \sum_{i=0}^{p-1} \beta_i w(k-i) \tag{3.17}$$

onde p é ordem do modelo e $w(k-i)$, $i = 0, 1,..., p-1$, são saídas de um processo estocástico do tipo ruído branco.

A justificativa conceitual para o modelo *MA* não é tão imediata quanto para o modelo *NFIR*. Porém, uma possível explicação para o mesmo, segundo CANWAY (1998), é aquela que concebe o processo estocástico do tipo ruído branco como sendo constituído por uma superposição contínua e infinita de séries temporais. Neste ponto, é útil pensar na transformada de Fourier de uma série temporal do tipo ruído branco, a qual resulta em um espectro plano, indicando a presença de todo o espectro contínuo de freqüências. De fato, o termo "branco" é utilizado em analogia à luz branca. Neste contexto, o modelo MA pode ser visualizado como procedendo à extração de uma série particular do ruído branco, por intermédio do filtro linear definido pelos coeficientes β_i.

3.5.3 Modelo auto-regressivo não-linear com entrada externa (NARX)

Outro modelo muito utilizado em sistemas lineares para a representação de séries temporais estacionárias é o modelo auto-regressivo (*AR*). Neste caso, a saída do estágio k+1 é suposta ser relacionada linearmente com seus próprios valores para n instantes de tempo anteriores. O modelo *AR* para predição de séries temporais supõe um ruído aditivo, de modo que o modelo de predição assume a forma:

$$y(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) + w(k) \tag{3.18}$$

onde $w(k)$ é um ruído branco. O preditor teórico associado é dado pela seguinte equação:

$$\hat{y}(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) \tag{3.19}$$

Tal estratégia é útil para a parametrização de modelos lineares para predição de série temporais usando um número finito de parâmetros, sendo que nenhuma entrada externa é explicitamente especificada. A versão não-linear deste é descrita pela equação (3.20), onde f é uma função não-linear a ser determinada:

$$y(k+1) = f[y(k), y(k-1), \cdots, y(k-n-1)] \tag{3.20}$$

Dada uma série temporal $\{y(t)\}$, a estrutura apresentada na figura 3.3 pode ser usada para representá-la, sendo que a função f pode ser aproximada por uma rede neural.

Uma rede neural do tipo *RBF* é recomendada somente quando o número de atrasos a ser usado no modelo é pequeno. Portanto, para um número grande de atrasos, uma rede

neural multicamada deve ser utilizada. Redes neurais para predição de séries temporais estacionárias foram consideradas por NARENDRA & PARTHASARATHY (1992). Recentemente, tais modelos foram usados por MUKHOPADHYAY & NARENDRA (1991) para representar distúrbios externos em sistemas dinâmicos.

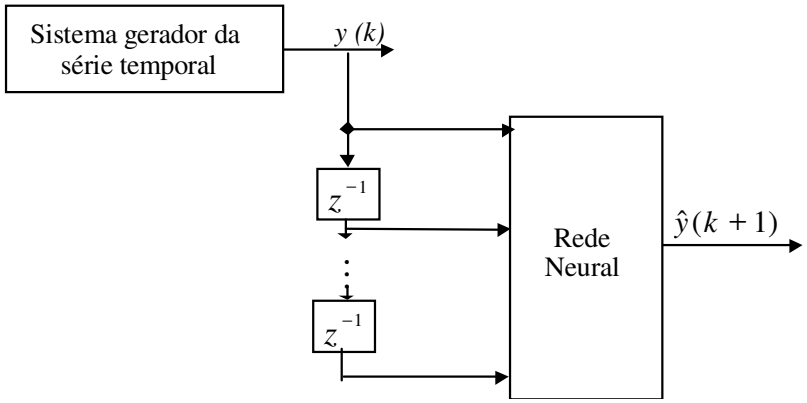


Figura 3.3 – Modelo neural *NAR* para predição de séries temporais

Para o caso de identificação de sistemas dinâmicos, o modelo auto-regressivo não-linear com entrada externa (*NARX*) supõe um ruído aditivo e entrada externa *u* em instantes de tempo anteriores, de modo que o modelo de predição assume a forma:

$$y(k+1) = f[y(k),..., y(k-n+1), u(k),..., u(k-m+1)] + w(k+1) \tag{3.21}$$

onde $w(k)$ é um ruído branco. O preditor teórico associado é dado pela seguinte equação (veja figura 3.4):

$$\hat{y}(k+1) = f[y(k),..., y(k-n+1), u(k),..., u(k-m+1)] \tag{3.22}$$

Supondo que o modelo descrito pela equação (3.22) representa o sistema, então o erro de predição $e(k+1) = \hat{y}(k+1) - y(k+1)$ será igual ao ruído $w(k+1)$, e sua variância é mínima. Assim, o preditor pode ser realizado através de uma rede neural dada pela seguinte equação:

$$\hat{y}(k+1) = \varphi[y(k),..., y(k-n+1), u(k),..., u(k-m+1), \theta] \tag{3.23}$$

onde φ é uma função não-linear implementada por uma rede neural multicamadas com vetor de parâmetros θ.

O modelo *NARX* apresenta as seguintes características:

- tem a vantagem de admitir representações mais adequadas para dinâmicas com resposta lenta, quando comparado com o modelo *NFIR*;
- modelos *NARX* geralmente requerem dimensões elevadas para o vetor de entrada, como condição necessária para uma boa representação das propriedades do ruído (SJÖBERG, 1995).

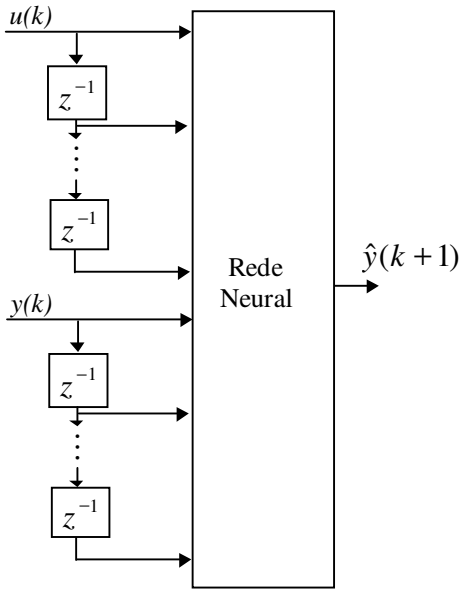


Figura 3.4 – Modelo neural *NARX* para identificação de sistemas dinâmicos

3.5.4 Modelos auto-regressivos com média móvel (ARMA, ARMAX, NARMAX)

É conhecido que qualquer planta dinâmica linear de tempo discreto de ordem p pode ser representada por uma função de transferência $W_p(z)$ tal que:

$$W_p(z) = \frac{\beta_0 z^{-1} + \beta_1 z^{-2} + \cdots + \beta_{p-1} z^{-p}}{z + \alpha_0 z^{-1} + \alpha_1 z^{-2} + \cdots \alpha_{n-1} z^{-n}} \tag{3.24}$$

ou, de modo equivalente, por uma equação a diferenças:

$$\hat{y}(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) + \sum_{j=0}^{p-1} \beta_j w(k-j) \tag{3.25}$$

O modelo descrito pela equação (3.24) combina a parte com média móvel (*MA*) e a parte auto-regressiva (*AR*), resultando no modelo *ARMA* (auto-regressivo com média móvel) usado extensivamente em processamento de sinais e controle de processos.

Como uma extensão do modelo *ARMA*, temos o modelo *ARMAX*, que também considera valores passados da entrada externa. O modelo *NARMAX* é, por sua vez, uma extensão do modelo *ARMAX* para o caso não-linear (LEONTARITIS & BILLINGS, 1985; CHEN & BILLINGS, 1989) e pode ser expresso da seguinte forma (RIVALS & PERSONNAZ, 1996):

$$y(k+1) = h[y(k),\cdots, y(k-n+1), u(k),\cdots, u(k-m+1), w(k),\cdots, w(k-p+1)] + \xi(k+1) \quad (3.26)$$

O modelo descrito pela equação (3.26) apresenta uma dificuldade prática, que é a determinação dos valores $w(k)$, $w(k-1)$,..., $w(k-p+1)$. Estes devem corresponder a um ruído correlacionado com o erro. Uma das alternativas utilizadas é substituir o valor de $w(k)$ pelo erro de predição $e(k) = \hat{y}(k) - y(k)$. O preditor teórico realimentado de ordem p associado ao modelo (suas variáveis de estados são os p valores do erro de predição passado) pode então ser expresso da seguinte forma:

$$y(k+1) = h[y(k),\cdots, y(k-n+1), u(k),\cdots, u(k-m+1), e(k),\cdots e(k-p+1)] \quad (3.27)$$

Supondo que o modelo descrito pela equação (3.27) representa o sistema e que os p primeiros erros de predição são iguais ao ruído, $e(k+1) = w(k+1)\ \forall\ k$; então a variância do erro de predição é mínima. O efeito da condição inicial arbitrária desaparecerá dependendo da função desconhecida h.

Para implementar o preditor associado, deve-se portanto treinar uma rede neural da forma:

$$\hat{y}(k+1) = \varphi[y(k),\cdots, y(k-n+1), u(k),\cdots, u(k-m+1), e(k),\cdots, e(k-p+1); \theta] \quad (3.28)$$

onde φ é uma função não-linear implementada pela rede. Se o modelo suposto é correto e se φ aproxima h com uma precisão arbitrária, então o preditor é assintoticamente próximo do ótimo.

Com o intuito de diminuir a complexidade e simplificar a predição do modelo *NARMAX*, considera-se um modelo com ruído correlacionado. Este modelo é discutido em GOODWIN & SIN (1984), e é dado por:

$$A(q^{-1})\,y(k+1) = h[\,y(k),\cdots,y(k-n+1),u(k),\cdots,u(k-m+1)] + C(q^{-1})w(k) \qquad (3.29)$$

onde A e C são polinômios monotônicos com grau n e p, respectivamente, e h é uma função não-linear. O preditor teórico associado a este modelo é dado por:

$$\hat{y}(k+1) = h\begin{bmatrix} y(k),\cdots,y(k-n+1),u(k),\cdots,u(k-m+1)] + \\ +(1-A(q^{-1}))\,y(k+1) + (C(q^{-1})-1)e(k+1) \end{bmatrix} \qquad (3.30)$$

Se C não tem zeros fora ou no círculo unitário, o efeito da condição inicial arbitrária desvanecerá.

Assim, a rede neural usada para treinamento impõe uma dependência linear entre sua saída e o erro de predição passado:

$$\hat{y}(k+1) = \varphi[\,y(k),\cdots,y(k-n+1),u(k),\cdots,u(k-m+1)] + \Pi(q^{-1})e(k+1) \qquad (3.31)$$

onde Π é um polinômio com coeficientes ajustáveis.

O modelo *NARMAX* apresenta as seguintes características:

- tem a vantagem de permitir uma maior flexibilidade no processo de modelagem do ruído. A presença de entradas diretamente vinculadas ao ruído evita a necessidade de se utilizar um grande número de elementos da entrada ($u(k)$, $y(k)$) para incorporar a descrição do ruído, como ocorre nos modelos *NARX* (SJÖBERG, 1995);
- tem a desvantagem de requerer processos de ajuste de parâmetros mais complexos, pois estes devem considerar também o efeito da recorrência (realimentação da informação da saída) (VON ZUBEN, 1996).

3.5.5 Modelo não-linear de erro de saída (NOE)

O modelo de saída supõe um ruído branco aditivo na saída. Ele é dado por:

$$y(k+1) = h[\,\hat{y}(k),\cdots,.\hat{y}(k-n+1),u(k-1),\cdots,u(k-m+1)] + w(k) \qquad (3.32)$$

Considere o seguinte preditor teórico realimentado de ordem n:

$$\hat{y}(k+1) = h\left[\hat{y}(k),\cdots,\hat{y}(k-n+1),u(k),\cdots u(k-m+1)\right] \qquad (3.33)$$

Se o modelo suposto é correto e se os n primeiros erros de predição $e(k)$ são estatisticamente equivalentes a $w(k)\ \forall\ k$, então o erro de predição é mínimo. Os efeitos da condição inicial arbitrária desaparecerão dependendo da função desconhecida h. Analisando a equação (3.33) do preditor teórico, pode-se observar que este modelo difere do modelo *NARX*, pelo fato de utilizar a saída fornecida pelo sistema de identificação $\hat{y}$ como entrada, ao invés da saída medida do sistema a ser identificado y.

Para implementar o preditor associado à equação (3.33), deve-se treinar uma rede neural realimentada na forma:

$$\hat{y}(k+1) = \varphi(\hat{y}(k), \cdots, \hat{y}(k-n+1), u(k), \cdots, u(k-m+1); \theta) \tag{3.34}$$

O modelo *NOE* apresenta as seguintes características:

- tem a vantagem de dispensar a identificação das propriedades do ruído, por esta etapa estar inerentemente associada à identificação do sistema dinâmico;
- tem a desvantagem de requerer processos de ajustes de parâmetros mais complexos, pois estes devem considerar também o efeito da recorrência (realimentação da informação da saída) (VON ZUBEN, 1996).

3.5.6 Modelo por espaço de estados

A planta desconhecida pode ser descrita por uma equação de estados da seguinte forma:

$$\begin{aligned} x(k+1) &= f[x(k), u(k)] \\ y(k) &= h[x(k)] \end{aligned} \tag{3.35}$$

Se o estado do sistema $x(k)$ para o tempo k é mensurável, o problema de identificação fica substancialmente simplificado. Em tal caso, redes neurais N_f e N_h são usadas para aproximar f e h, respectivamente. O modelo é descrito pela equação:

$$\begin{aligned} \hat{x}(k+1) &= N_f[x(k), u(k)] \\ \hat{y}(k) &= N_h[x(k)] \end{aligned} \tag{3.36}$$

E os parâmetros das redes neurais N_f e N_g são ajustados usando back-propagation estático, descrito no capítulo 2. Os erros $\hat{x}(k) - x(k)$ e $\hat{y}(k) - y(k)$ são usados, respectivamente, para ajustar os parâmetros das duas redes.

O problema é substancialmente mais complexo quando o estado $x(k)$ do sistema não é mensurável em sua totalidade. Em tal caso, o modelo tem a forma

$$\begin{aligned} \hat{x}(k+1) &= N_f[\hat{x}(k), u(k)] \\ y(k+1) &= N_h[\hat{x}(k)] \end{aligned} \tag{3.37}$$

onde N_f e N_h são novamente redes neurais usadas para aproximar a função f e h. A estrutura do identificador é mostrado na figura 3.5. O fato de $x(k)$ não ser acessível torna necessário o uso do back-propagation dinâmico. Os detalhes para ajuste dos parâmetros de N_f e N_h são discutidos em LEVIN & NARENDRA (1992). Repare que no caso da figura 3.5 é possível realizar uma boa identificação sem que as redes neurais N_f e N_h correspondam isoladamente às funções f e h, pois o fundamental passa a ser a composição de ambas, o que pode ser realizado de variadas formas.

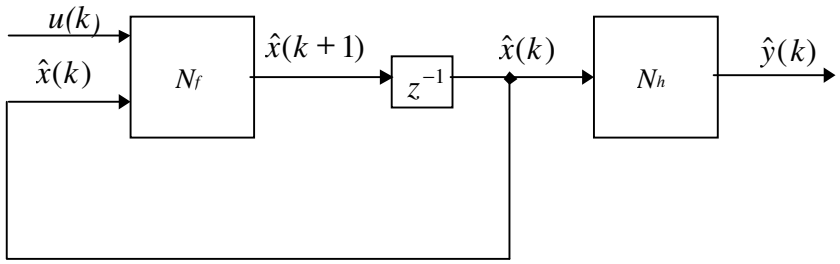


Figura 3.5 – Modelo não-linear por espaço de estados com vetor de estados inacessível

3.6 Algumas técnicas de identificação de sistemas utilizando redes neurais

Modelos lineares têm sido amplamente usados em identificação de sistema por duas razões principais:

- as saídas associadas a sinais de entradas diferentes são facilmente determinadas pelo princípio da superposição de efeitos;
- as ferramentas de identificação para o caso linear encontram-se bem evoluídas.

Entretanto, como já discutido anteriormente, muitos sistemas de controle encontrados na prática não são lineares. Além disso, em muitos casos, aproximações produzidas por modelos lineares não são convenientes para representar estes sistemas, e

modelos não-lineares devem ser considerados diretamente. Efeitos não-lineares como (geração de harmônicas, múltiplos pontos de equilíbrio e caos) fazem com que muitos dos princípios e ferramentas desenvolvidos para o tratamento de sistemas lineares não sejam válidos para sistemas não-lineares. Portanto, identificação de sistemas não-lineares é uma tarefa muito mais difícil que identificação de sistema lineares.

O procedimento principal de identificação de sistemas consiste na seleção da estrutura de um modelo paramétrico e estimação dos valores dos parâmetros envolvidos. O primeiro diz respeito à seleção da classe de operadores matemáticos a ser usado como modelo. O segundo envolve um algoritmo de estimação, e usualmente requer dados de entrada-saída do processo, uma classe de modelos de identificação e critérios de avaliação da qualidade da identificação.

Um grande número de técnicas têm sido desenvolvidas nos últimos anos para seleção de modelos e estimação de parâmetros de sistemas não-lineares. Algoritmos de regressão direta e reversa foram analisados por LEONTARITIS & BILLINGS (1987). Regressão passo-a-passo foi usada por BILLINGS & VOON (1996) e uma classe de estimadores ortogonais foi discutida em KORENBERG *et al.* (1988). Algoritmos desenvolvidos com o objetivo de economia de memória e de modo a permitir uma rápida computação foram propostos por CHEN & WIGGER (1995). Métodos para determinar a estrutura prévia do modelo foram estudados por LJUNG & GLAD (1994). Um panorama de técnicas previamente existentes para identificação de sistemas nos anos 70 foi apresentado por BILLINGS (1980), enquanto HABER & UNBEHAUEN (1990) apresentaram um panorama de detecção de estruturas de entrada-saída de sistemas não-lineares. Um recente panorama de modelagem e identificação de sistema utilizando abordagens neurocomputacionais pode ser encontrado em SJOBERG *et al.* (1995).

Em aplicações de identificação de sistemas, as redes neurais artificiais impõem poucas restrições na definição do modelo, quando comparadas com outras técnicas. A adequação e viabilidade da aplicação de arquiteturas de redes neurais em identificação de sistemas têm sido demonstradas por alguns estudos em tempo discreto (BILLINGS & CHEN, 1992; CHEN *et al.* 1990; KADIRMAKAMANATHAN & LIU, 1995; KUSCHEWSKI *et al*., 1993) e tempo contínuo (LIU *et al.*,1996, 1998a).

Muito já tem sido feito utilizando redes neurais para análise e projeto de controladores de sistemas não-lineares (ANTSAKLIS, 1990; KARAKADOGLU *et al.*, 1993; LEVIN & NARENDRA, 1993; MILLER, SUTTON & WERBOS, 1990; NARENDRA & PARTHASARATHY, 1990; POLICARPOU & IOANNOU, 1991; SADEGH, 1993; SANNER & SLOTINE, 1992; TZIRKEL-HANCOCK, 1992). Neste contexto, diversas metodologias podem ser empregadas:

- aprendizado da ação de controle por sinal de reforço (ANDERSON,1993);
- aprendizado da dinâmica inversa (MILLER, SUTTON & WERBOS, 1990; PSALTIS & SIDERIS, 1988), sem qualquer garantia de estabilidade;
- controle adaptativo indireto (NARENDRA & PARTHASARATHY, 1990);
- controle adaptativo direto, com garantia de estabilidade para o sistema resultante (KARAKADOGLU *et al.*,1993; POLICARPOU & IOANNOU, 1991; SADEGH,1993; SANNER & SLOTINE,1992; TZIRKEL-HANCOCK, 1992).

O tipo mais comum de rede neural usada para controle é o *perceptron* multicamadas (*MLP*) com função de ativação sigmoidal (RUMELHART, HINTON & WILLIAMS,1986; KARAKADOGLU *et al.*, 1993; MILLER *et al.*, 1990; NARENDRA & PARTHASARATHY, 1990; PSALTIS & SIDERIS, 1988; SADEGH, 1993) e redes neurais com funções de ativação de base radial (*RBF*) (BROOMHEAD & LOWE, 1988; POGGIO & GIROSI, 1990; ANDERSON, 1993; POLICARPOU & IOANNOU, 1991; SANNER & SLOTINE, 1992; TZIRKEL-HANCOCK, 1992). Os controladores neurais diretos adaptivos são projetados para uma classe de sistemas não-lineares tendo dinâmica desconhecida. Os sinais de controle são gerados baseados em técnica de linearização por realimentação, usando uma rede neural para aproximação de uma função não-linear do estado *n*-dimensional do sistema (KARAKADOGLU *et al.*, 1993; SANNER & SLOTINE, 1992; TZIRKEL-HANCOCK, 1992).

Recentemente, SADEGH (1993) apresentou um esquema de controle neural direto adaptivo para sistemas não-realimentados linearizáveis. A adaptação nos controladores diretos adaptivos envolve ajustes seqüenciais dos parâmetros da rede neural, baseados em leis derivadas de modo similar ao modelo clássico de projeto de controle adaptativo por modelo de referência (*MRAC*) (ÅSTROM & WITTENMARK, 1994; LANDAU, 1979;

NARENDRA & ANNASWAMY, 1989; SASTRY & BODSON, 1989), o qual recorre aos critérios de Lyapunov, em que a estabilidade do sistema na presença de adaptação é garantida. Estes critérios forçam a trajetória do erro a tender a zero, ou no mínimo permanecer limitada.

Vários outros tipos de estruturas de redes neurais têm sido demonstradas serem ferramentas eficazes também na identificação de sistemas não-lineares. Por exemplo, modelos que realizam um pré-processamento dos sinais de entrada, baseado em série de Volterra, modelos *GMDH* e modelos *SONN* (FU & FARISON, 1973; FARLOW, 1984; PARK & SANDBERG, 1991; TENORIO & LEE, 1989). Entre os pesquisadores da comunidade de controle que têm usado redes neurais artificiais há mais tempo, NARENDRA & PARTHASARATHY (1990), LEVIN & NARENDRA (1992) e NARENDRA & PARTHASARATHY (1989) têm procurado incorporar redes neurais artificiais dinâmicas como componentes do sistema, concentrando-se em identificação de sistemas e controle de plantas não-lineares. Neste contexto, CHEN (1989) introduziu uma rede neural que apresenta um mapeamento não-linear adicional na camada de entrada, para reduzir a complexidade do modelo resultante.

Quando comparadas a técnicas desenvolvidas para o tratamento de problemas de identificação linear, o emprego de redes neurais pode ser considerado como uma abordagem muito flexível, inclusive por não requerer informação prévia do modelo. Mais ainda, técnicas baseadas em redes neurais não requerem a suposição de linearidade. Assim, embora seja verdadeiro que redes neurais podem oferecer pouca, se é que alguma, contribuição no tocante a métodos para identificação de sistema lineares, ela apresenta um potencial que pode ser diretamente explorado no desenvolvimento de novas técnicas de identificação de sistemas não-lineares.

A maioria das técnicas de identificação de sistema são aplicadas anteriormente à operação (*off-line*). Entretanto, se há uma mudança no comportamento do sistema ou em seu ponto de operação, isto exigiria a readaptação dos parâmetros do modelo de identificação (por exemplo, uma rede neural), sob pena de o sistema de identificação sofrer uma degradação de desempenho. Para evitar isto, algumas redes neurais baseadas no esquema de identificação necessitam de leis de ajuste dos parâmetros do modelo para tempo de operação, mantendo-se fixa a dimensão da rede, a qual deve ser suficientemente elevada para contemplar todos os possíveis requisitos de comportamento que podem se

estabelecer. Isto freqüentemente acarreta uma sobre-estimação da estrutura da rede, ou seja, um modelo de identificação que não é ótimo.

Na formulação de tempo discreto, algumas abordagens têm sido desenvolvidas para determinar o número de neurônios nas camadas escondidas, usando teoria de decisão (BAUM & HAUSSLER, 1989) e métodos de comparação, tal como *MDL* (*minimum description length*) (SMYTH, 1991) e métodos bayesianos (MACKAY, 1992). O problema associado a esses métodos é que geralmente eles requerem que todas as observações estejam disponíveis ao mesmo tempo, não sendo convenientes para identificação em tempo de operação (*on-line*). Portanto, a fim de se obter um melhor desempenho, tanto a estrutura quanto os parâmetros do modelo necessitam ser modificados em resposta a variações da características da planta e no ponto de operação. Recentemente, novos algoritmos têm sido desenvolvidos para operar sobre uma 'janela' de dados, usada como base para capturar as variações da estrutura do modelo (FUNG *et al.*, 1996; LUO & BILLINGS, 1995, LUO & BILLINGS, 1998; LUO *et al.*, 1996) e atualizar os parâmetros envolvidos. Como já mencionado, os métodos construtivos (métodos não-paramétricos) não estão sujeitos a este tipo de problema e representam um ferramenta em potencial para aplicação em controle e identificação de sistemas, como será visto no capítulo 7.

Em VANLANDINGHAM *et al*. (1993), duas técnicas de identificação para sistemas *MIMO* (múltiplas entradas e múltiplas saídas) foram apresentadas com o objetivo de proporcionar uma base para comparação e seleção. A primeira técnica é um novo procedimento de identificação, que usa dados de entrada-saída para estabelecer, não somente a ordem do sistema, mas também o número mínimo de parâmetros requeridos para representação (BINGULAC & VANLANDINGHAM, 1993). A segunda técnica usa uma rede neural com aprendizado supervisionado baseado no algoritmo de retropropagação (MCCLELLAND & RUMELHART, 1986) para determinar o modelo de identificação.

Em LIU *et al.* (1998b), é proposta uma rede neural com estrutura polinomial, seguida do processo de estimação de parâmetros do modelo selecionado. No sentido de conduzir à obtenção de uma rede neural apropriada, o algoritmo *OLS* (*Orthogonal Least Squares*) é introduzido para seleção *off-line* de estruturas, e este é então generalizado para produzir técnicas de crescimento da rede para seleção *on-line* da estrutura. No estágio *off-line*, o *OLS* é usado para selecionar um conjunto de funções-base de polinômios de Volterra e para

arranjar a ordem de acordo com sua habilidade de reduzir o erro de aproximação. Na seleção *on-line*, a técnica de crescimento da rede é usada para aproximar gradualmente a complexidade do sistema, com base nas observações recebidas. Para estimação dos parâmetros, um novo algoritmo de aprendizado é desenvolvido, mais uma vez baseado nos critérios de estabilidade de Lyapunov.

Há muitos problemas difíceis de serem superados, principalmente quando os sistemas não-lineares em estudo são ao mesmo tempo complexos e instáveis. Esta última condição cria dificuldades no treinamento da rede neural (BERTRAND, 1992). Uma abordagem é estabilizar localmente o sistema. Tal estabilização para sistemas dinâmicos não-lineares pode ocorrer no caso de sistemas que sejam controláveis próximos de um estado de equilíbrio, permitindo estabilizar o modelo linearizado próximo de um ponto de equilíbrio com realimentação linear (LEVIN & NARENDRA, 1992; CHOI & VANLANDINGHAM, 1992).

Utilizando a técnica descrita acima, em CHOI *et al.* (1996) é proposto um modelo *MLP*, com suas entradas consistindo de sinais de entradas e saídas atrasadas, chamado de *MLP* dinâmica modificada (figura 3.6). A rede *MLP* dinâmica modificada é restrita a ter somente uma única camada escondida. Os primeiros passos são para determinar um ponto de equilíbrio e identificar o sistema linearizado em torno deste ponto de equilíbrio pela combinação de uma aproximação conexionista (aprendizado supervisionado com retropropagação) e um método de identificação convencional para sistemas multivariáveis. Utilizando a abordagem clássica de identificação, pode-se examinar todas as possíveis estruturas do sistema de modo a produzir um modelo linear que generalize otimamente sobre os dados de entrada-saída, em torno do ponto de equilíbrio. Este modelo linear é novamente incorporado ao modelo *MLP* como parte do sistema (CHOI & VANLANDINGHAM, 1993; VANLANDINGHAM *et al.*, 1993). Pelo menos conceitualmente, e para sistemas tratáveis, a parte remanescente não-linear do sistema pode ser capturada com uma simples camada tendo um número (a ser determinado) de elementos de processamento (neurônios) com funções de ativação não-linear. O número requerido de neurônios na camada escondida, e que proporciona resultados satisfatórios, pode ser determinado sistematicamente (FAHLMAN & LEBIERE, 1990). Com esta abordagem, uma rede de complexidade mínima pode ser obtida, com a poda na rede não sendo requerida. Em

princípio, este é um método construtivo, à medida que após determinada a estrutura inicial (número de neurônios e tempo de atraso), ele incrementa o número de neurônios até obter uma aproximação desejável, embora necessite reiniciar o treinamento toda vez que um neurônio é acrescentado.

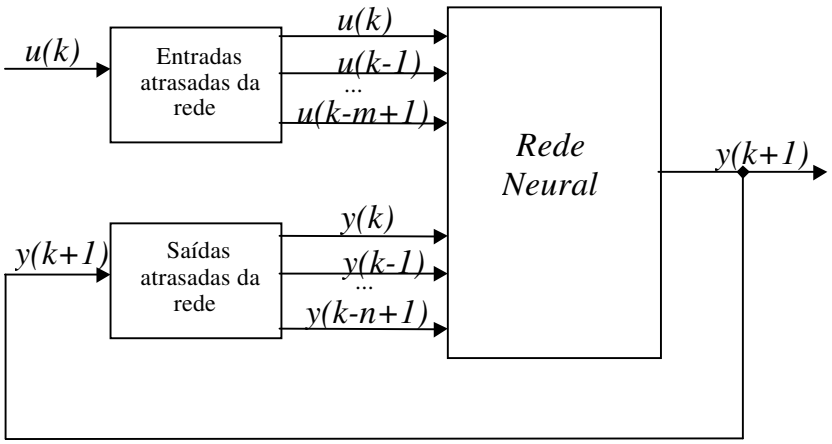


Figura 3.6 - Rede neural dinâmica modificada

Em SADEGH (1993), é apresentado um esquema no qual a rede aprende diretamente a entrada do controle como função do estado desejado da planta. Isto requer, em muitos casos, uma rede com poucas interconexões, o que leva a um menor tempo de treinamento. Uma desvantagem, entretanto, é a possibilidade de perda de estabilidade na malha fechada, a qual pode resultar da interação da planta com a dinâmica da rede. Um análise mais cuidadosa é, portanto, necessária para garantir a estabilidade em malha fechada.

Em FABRI & KADIRKAMANATHAN (1996) é apresentada uma técnica de controle adaptativo, usando rede neural com funções de ativação de base radial (gaussiana) e estrutura dinâmica, que cresce de acordo com o tempo. Este crescimento visa abranger a nova localização do estado do sistema. Esta foi aplicada a uma classe de sistemas não-lineares tendo dinâmica desconhecida ou parcialmente desconhecida. O método resulta em uma rede que é econômica em termo de tamanho, principalmente se o estado se estende somente por um pequeno subconjunto do espaço de estados. Adicionalmente, o sistema é aumentado a partir do emprego de técnicas de controle por modos deslizantes, para garantir estabilidade global, impedindo que o estado se mova para fora da região do espaço de estados estendido. Isto permite também inserir robustez de ação contra distúrbios que

surgem devido ao erro de aproximação da rede e para o fato de que, limitando o tamanho da rede, um número mínimo de funções de base radial possa ser utilizado. Leis de adaptação e ganho de controle deslizantes que garantem estabilidade no sentido de Lyapunov são apresentadas, junto com técnicas para determinar quais funções-base são necessárias para formar a estrutura da rede.

3.7 Caracterização das plantas

As quatros classes de modelos de planta introduzidas aqui serão utilizadas no capítulo 7, e permitem o uso de redes neurais e/ou abordagem construtiva baseada na teoria de agentes para identificação e controle de sistema dinâmicos não-lineares. Estas classes de modelos podem ser descritas pelas seguintes equações a diferenças não-lineares:

$$Modelo\ I: y(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) + f[u(k), u(k-1), \cdots, u(k-m+1)] \tag{3.38}$$

$$Modelo\ II: (k+1) = f[y(k), y(k-1), \cdots, y(k-n+1)] + \sum_{i=0}^{n-1} \beta_i y(k-i) \tag{3.39}$$

$$\begin{aligned} Modelo\ III: y(k+1) = {} & f[y(k), y(k-1), \cdots, y(k-n+1)] + \\ & g[u(k), u(k-1), \cdots, u(k-m+1)] \end{aligned} \tag{3.40}$$

$$Modelo\ IV: y(k+1) = f[y(k), y(k-1), \cdots, y(k-n+1); u(k), u(k-1), \cdots, u(k-m+1)] \tag{3.41}$$

onde $[u(k), y(k)]$ representa o par de entrada-saída da planta SISO no instante de tempo k.

As funções f e g são supostas serem funções diferenciáveis em relação a seus argumentos. É evidente que o Modelo IV engloba os Modelos I a III. Entretanto, o Modelo IV é analiticamente o menos tratável e, sendo assim, em aplicações práticas, o uso de algum dos outros modelos pode ser mais atrativo. A seguir, cada um dos modelos é brevemente descrito.

Modelo I: A saída da planta não-linear desconhecida é suposta depender linearmente de seus valores passados e não-linearmente dos valores passados da entrada (figura 3.7).

Modelo II: Este modelo é realizado como mostrado na figura 3.8. Neste caso, a saída depende linearmente dos valores atuais e passados da entrada *u(k)*, mas não-linearmente em relação a seus próprios valores passados.

Modelo III: A planta desconhecida neste caso depende não-linearmente dos valores passados da entrada e da saída. Entretanto, os efeitos dos valores de entrada e saída se manifestam em uma composição aditiva. A representação do modelo é mostrada na figura 3.9.

Modelo IV: Como mencionado previamente, este é o mais geral de todos os modelos introduzidos aqui e engloba os modelos anteriores. A saída para qualquer instante, neste caso, é uma função não-linear dos valores passados das entradas e saídas. A representação do modelo, usando linhas de atraso, é mostrado na figura 3.10.

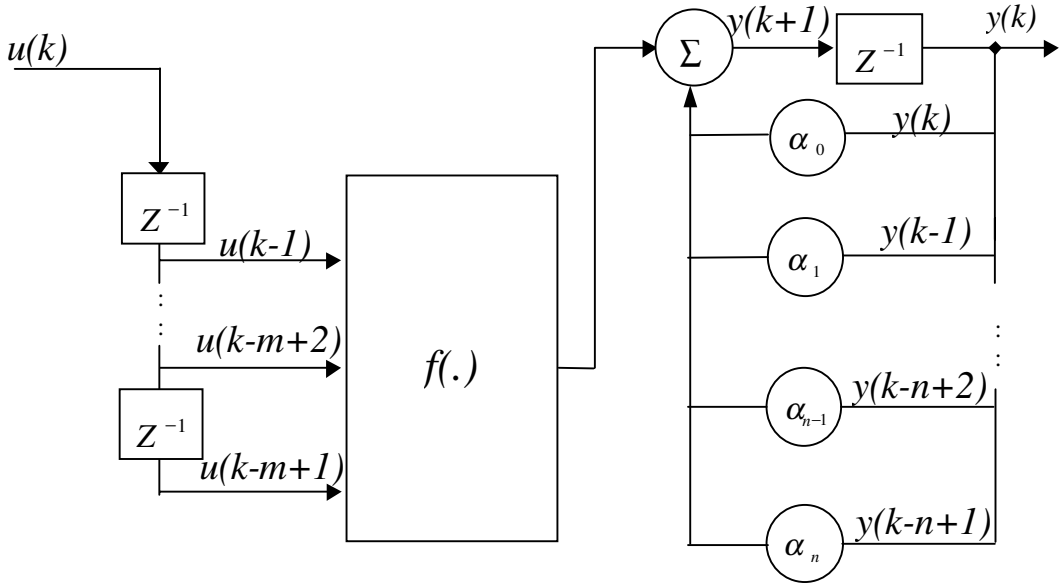


Figura 3.7 – Estrutura do Modelo I

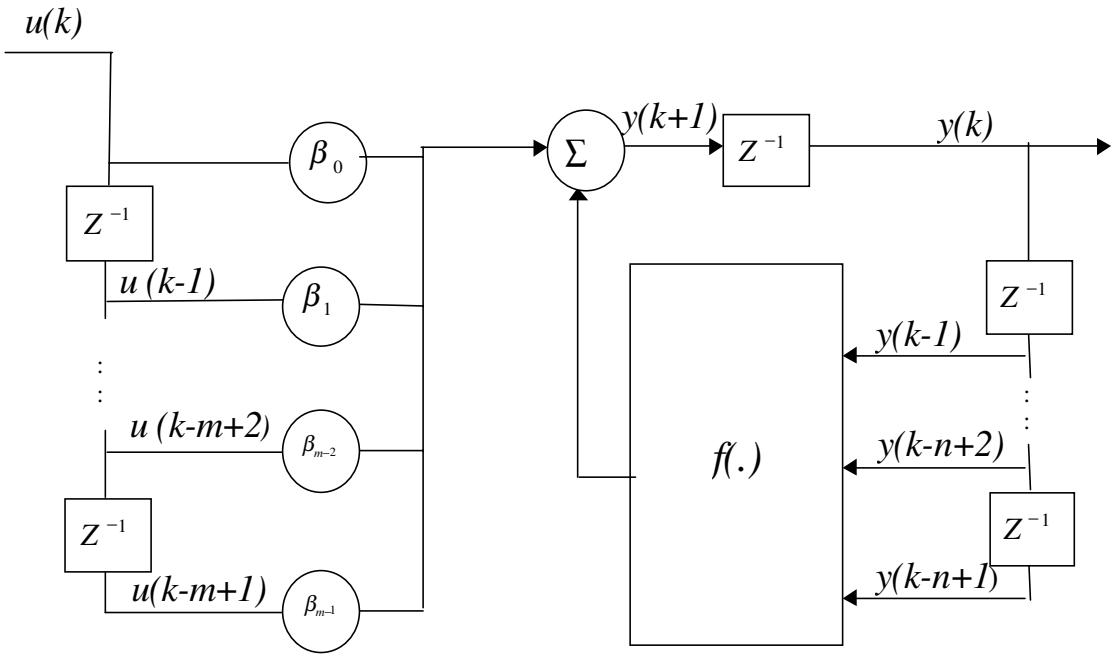


Figura 3.8 – Estrutura do Modelo II

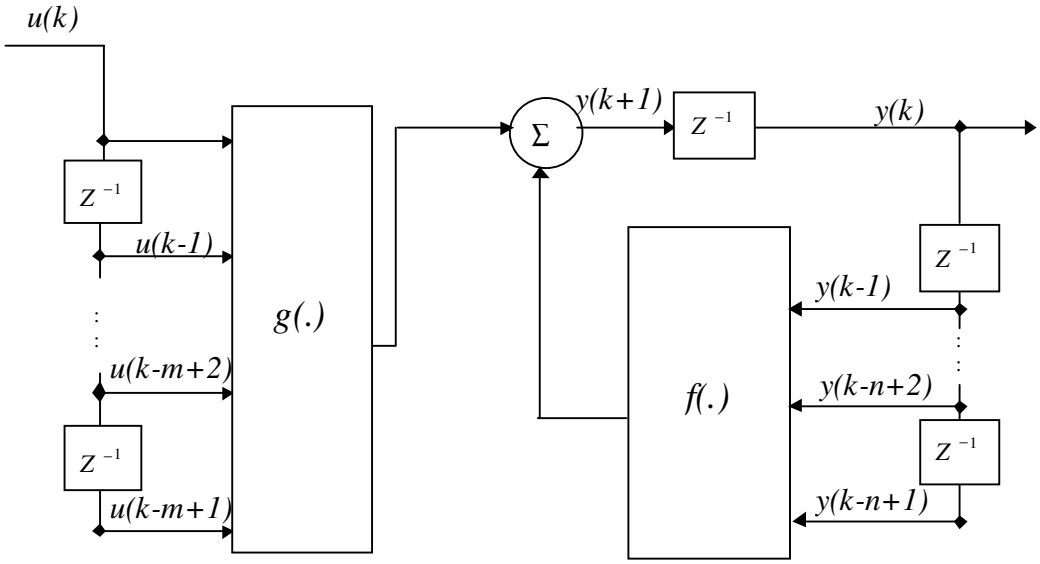


Figura 3.9 – Estrutura do Modelo III

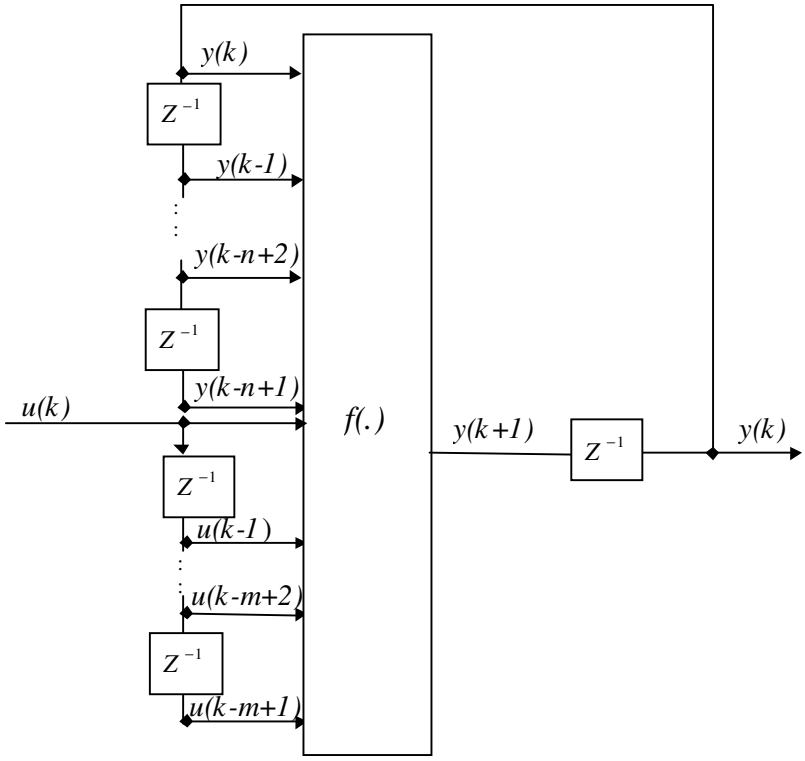


Figura 3.10 – Estrutura do Modelo IV

3.8 Procedimentos de identificação utilizando redes neurais

A identificação do modelo da planta é feita através de uma rede neural e de linhas de atraso. Em cada caso, a rede neural é suposta conter um número suficiente de camadas e de neurônios em cada camada, de modo a permitir a representação das características de entrada-saída do correspondente mapeamento não-linear da planta. Do ponto de vista da análise matemática, isto implica que as funções não-lineares nas equações a diferenças descrevendo a planta podem ser substituídas por redes neurais com parâmetros (pesos) a determinar. Por isso, a solução teórica do problema de identificação adaptativa é suposta existir, por princípio.

Para identificar a planta, um modelo de identificação é escolhido baseado na informação prévia sobre a classe à qual ele pertence. Por exemplo, supondo que a planta tem estrutura descrita pelo Modelo III, o modelo é escolhido para ter a forma mostrada na figura 3.11. O objetivo então é determinar os pesos das duas redes neurais N_1 e N_2 de modo

que o mapeamento N_1 seja igual a $g(.)$ e o mapeamento N_2, igual a $f(.)$. Se $y(k+1)$ e $\hat{y}(k+1)$ são respectivamente a saída para o estágio $k+1$ da planta e do modelo de identificação, o erro é usado para atualizar os pesos de N_1 e N_2. Como descrito anteriormente, retropropagação estática ou dinâmica podem ser utilizadas, dependendo da estrutura do modelo identificado.

Modelo Paralelo: Neste caso, a estrutura do identificador é idêntica àquela da planta, com f e g substituídos por N_2 e N_1, respectivamente. Isto implica que o modelo é descrito pela equação a seguir:

$$\hat{y}(k+1) = N_2[\hat{y}(k), \hat{y}(k-1), \cdots, \hat{y}(k-n+1)] + N_1[u(k), u(k-1), \cdots, u(k-m+1)] \qquad (3.42)$$

Como N_2 é uma malha de realimentação dinâmica, os parâmetros de N_1 e N_2 devem ser ajustados usando retropropagação dinâmica.

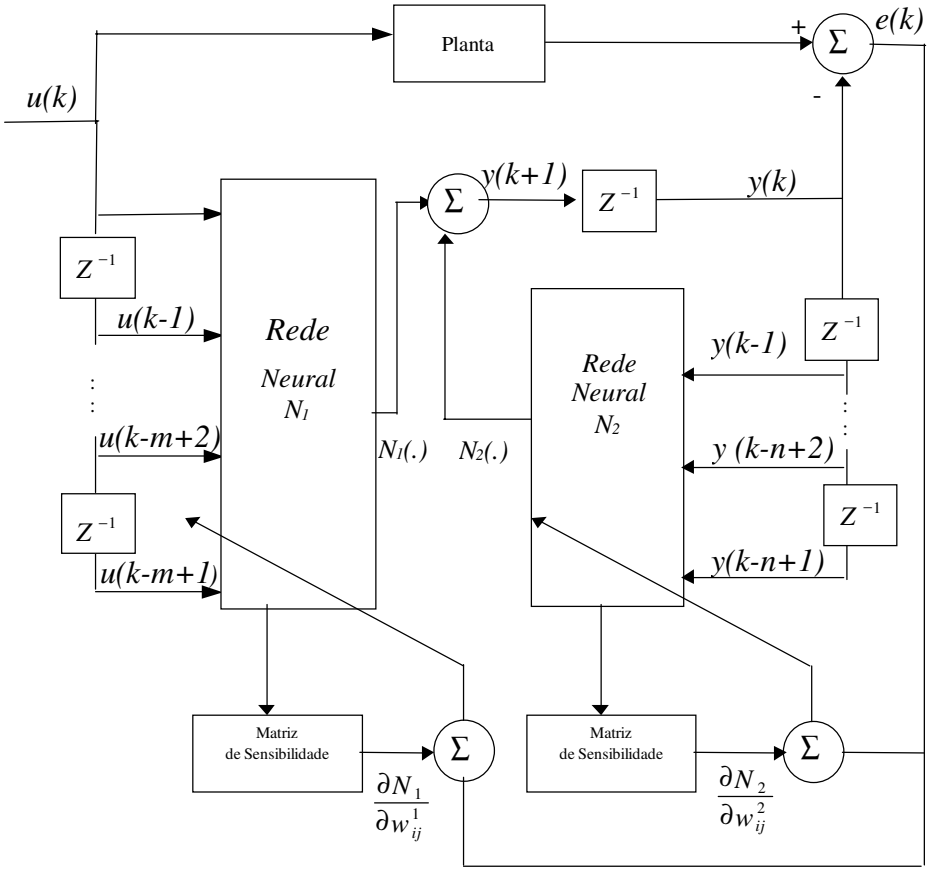


Figura 3.11 - Estrutura de um modelo de identificação

Modelo Série–Paralelo: Neste caso $y(k+1)$, ao invés de $\hat{y}(k+1)$, é usado para gerar a saída do modelo. Isto implica que o modelo fica descrito pela equação:

$$y(k+1) = N_2[y(k), y(k-1), \cdots, y(k-n+1)] + N_1[u(k), u(k-1), \cdots, u(k-m+1)] \qquad (3.43)$$

Como o modelo não inclui uma malha de realimentação contendo um elemento não-linear, a retropropagação estática, ao invés da retropropagação dinâmica, pode ser usada para atualizar os pesos da rede neural.

Os dois métodos descritos acima têm sido discutidos amplamente no contexto de identificação de sistemas lineares invariante no tempo com parâmetros desconhecidos (NARENDRA & ANNASWAMY, 1989). Enquanto o modelo série-paralelo tem sido mostrado ser globalmente estável, resultados similares não estão disponíveis para o modelo paralelo.

Para evitar muitas das dificuldades analíticas encontradas, bem como para assegurar estabilidade e simplificar o procedimento de identificação, o modelo série-paralelo foi usado por NARENDRA & PARTHASARATHY (1990). Simulações computacionais têm revelado que uma ampla gama de plantas não-lineares pode ser identificada usando o procedimento acima. Entretanto, estudos teóricos sobre estabilidade e convergência estão ainda em estágio inicial, e muitas questões devem ainda ser respondidas (NG, 1997).

3.9 Visão sintética do capítulo

Neste capítulo, apresentamos os vários tipos de representação de um sistema dinâmico, uma comparação de modelos lineares e não-lineares usando redes neurais e algumas técnicas e procedimentos de identificação. Parte dos modelos aqui apresentados serão utilizados no capítulo 7 para efeito de comparação com modelos a serem propostos nos próximos capítulos, em termos de desempenho junto a problemas de aplicação.

Capítulo 4

Abordagens para controle de sistemas dinâmicos não-lineares

4.1 Introdução

A aplicação de redes neurais em controle tem sido uma área de pesquisa ativa desde os anos 80, em virtude da dificuldade de controlar sistemas dinâmicos com alto grau de não-linearidade, incerteza e imprecisão, empregando técnicas convencionais de controle. Redes neurais apresentam algumas propriedades desejáveis para modelar sistema não-lineares e, com isto, podem superar algumas dificuldades encontradas por outras técnicas.

A habilidade das redes neurais em representar mapeamentos não-lineares, e assim modelar sistemas não-lineares, é a característica mais explorada na síntese de controladores. Recentemente, foram propostos diversos métodos de projeto de controle não-linear, incluindo principalmente técnicas de otimização, que têm um importante papel no ajuste dos parâmetros dos modelos envolvidos. Estes métodos serão explorados no capítulo 7.

Uma área estabelecida em teoria de controle moderno, com grande relevância aqui, é a teoria de sistemas adaptativos (ÅSTROM & WITTENMARK, 1994). Esta área tem produzido muitos resultados teóricos interessantes nos últimos anos. Embora a teoria de sistemas adaptativos seja fundamentada na suposição de sistemas lineares e invariantes no tempo, os conceitos e resultados encontrados na literatura estão sendo estendidos para uso no campo de redes neurais. No entanto, muitos resultados acabam sendo verificados apenas através de simulações, necessitando de uma maior fundamentação teórica.

Duas abordagens distintas têm sido empregadas para controle adaptativo de uma planta (GOODWIN & SIN, 1984; LANDAU, 1979):

- Controle Direto;
- Controle Indireto.

No controle direto, os parâmetros do controlador são ajustados para reduzir alguma norma do erro de saída. O sucesso desta abordagem depende do conhecimento a priori do jacobiano da planta, ou pelo menos do sinal de seus elementos. Na tentativa de eliminar a necessidade do conhecimento do jacobiano, PSALTIS *et al.* (1987) propôs uma aproximação *on-line*, perturbando a entrada da planta ligeiramente em torno de um ponto de operação e medindo a mudança na saída, ou comparando as mudanças em iterações anteriores. NG & COOK (1998), mostraram que tal aproximação pode, algumas vezes, resultar em propriedades indesejáveis para seguir uma referência. Ao invés disso, eles propuseram um conveniente algoritmo *on-line* para plantas com dinâmica não muito rápida.

Por outro lado, no controle indireto, os parâmetros da planta são estimados como elementos de um vetor $\varphi(k)$, para algum instante k, e os parâmetros $\theta(k)$ do controlador são escolhidos supondo que $\varphi(k)$ representa o valor verdadeiro do vetor de parâmetros p da planta. Mesmo quando a planta é suposta ser linear e invariante no tempo, os controles adaptativos direto e indireto conduzem a problemas de otimização não-lineares.

Conforme já mencionado, a aplicação de redes neurais em controle de processos representa uma alternativa às técnicas tradicionais. Várias arquiteturas, incluindo redes independentes para identificação e controle, são apresentadas abaixo. Além disso, métodos tradicionais de controle não-linear são revistos.

4.2 Abordagens tradicionais para controle de sistemas dinâmicos não-lineares

Os sistemas dinâmicos de tempo contínuo são tipicamente modelados por equações diferenciais ordinárias. Geralmente, recorre-se a um sistema acoplado de equações de primeira ordem, $\dot{\mathbf{x}} = F(\mathbf{x}, \mathbf{u}, t)$, onde os vetores $\mathbf{x}$ e $\mathbf{u}$ representam respectivamente o estado e a entrada do sistema e F é um campo vetorial não-linear (ISIDORI, 1989). O objetivo de controle é encontrar a entrada $\mathbf{u}$, a qual conduz o sistema a apresentar características

desejáveis, por exemplo, estabilidade e alto desempenho ao seguir uma referência. No caso do controle por realimentação, a entrada será uma função de algumas variáveis do próprio sistema, por exemplo, $\mathbf{u} = g(\mathbf{x})$.

Embora exista uma grande variedade de ferramentas para projeto de sistemas de controle lineares, este não é o caso para sistemas de controle não-lineares, e mesmo quando existe alguma técnica para solução de problemas de controle não-linear, ela não é aplicável genericamente na prática devido à sua complexidade e especificidade. A maior dificuldade é que nenhuma formalização genérica está disponível para o tratamento de todos os problemas não-lineares. Muitos métodos empregam aproximações, visto que as soluções exatas para as equações diferenciais não-lineares são raramente conseguidas. Entre as técnicas mais freqüentemente usadas temos *simulação*, *linearização local* e *ganho de escalonamento*, *linearização por realimentação global*, *funções descritivas*, *ganhos equivalentes* e *estratégias baseadas no segundo método de Lyapunov* (PHILLIPS & MÜLLER-DOTT, 1992).

O comportamento dinâmico acoplado de equações de 1ª ordem, utilizadas na geração do modelo, pode ser revelado através da solução numérica obtida por simulação em computadores digitais. Os resultados de simulação não conduzem a conclusões fortes sobre a natureza qualitativa, como por exemplo a garantia ou não de estabilidade do sistema, mas permitem avaliar o desempenho sob certas condições.

Linearização local é uma das técnicas mais comuns para análise e projeto de sistemas não-lineares, desde que ela aproxime localmente o sistema não-linear por um sistema linear. A validade da aproximação linear se restringe a uma região limitada em torno de um ponto de operação específico. Uma vez que existe um grande número de ferramentas para projeto e análise de sistemas lineares, a abordagem por linearização parte do conjunto de equações diferenciais não-lineares e produz um conjunto de equações diferenciais lineares para um ponto de operação específico. Esta técnica de linearização local torna-se ineficiente para sistemas altamente não-lineares, uma vez que um número não razoável de pontos de operação discretos pode ser requerido para modelar o sistema não-linear com um grau de precisão aceitável. Em se tratando de controle, esta tendência é altamente prejudicial, uma vez que tal projeto deve ser repetido para cada modelo linear.

Linearização por realimentação é uma abordagem de natureza mais abrangente, a qual requer um controlador não-linear com a função específica de remover um comportamento não-linear indesejado da planta. Esta técnica exige um amplo conhecimento da planta e essencialmente requer a construção de um controlador não-linear por realimentação para cancelar as não-linearidades da planta. O projeto de tal controlador não é trivial e os efeitos da modelagem incompleta e de distúrbios da planta ainda não foram plenamente investigados (ISIDORI, 1989).

Existem outras técnicas de aproximação menos freqüentemente usadas, mas que merecem ser mencionadas. O método da função descritiva considera a resposta do sistema para entradas senoidais (KOCHENBURGER, 1950). A função descritiva consiste no ganho e fase da freqüência fundamental da resposta do sistema (aproximação de 1ª harmônica) e ignora todos os outros componentes da resposta. Isto permite uma análise de estabilidade aproximada e a obtenção de medidas de desempenho para o sistema não-linear (OGATA, 1996).

O segundo método de Lyapunov (VIDYASAGAR, 1993; WANG & YEH, 1990) pode ser empregado na análise de estabilidade de sistemas não-lineares sob bases teóricas. O método foi desenvolvido para examinar a estabilidade de equações diferenciais não-lineares. Um importante aspecto do segundo método de Lyapunov é que ele não requer a solução do sistema de equações diferenciais. Embora o método nos poupe a tarefa de solucionar o sistema de equações diferenciais, ele requer a escolha de uma função de Lyapunov, para a qual não existem procedimentos sistemáticos de obtenção. A idéia básica do segundo método de Lyapunov é que um sistema físico pode somente armazenar uma quantidade finita de energia. Se puder ser mostrado que a energia está sempre sendo dissipada, exceto para pontos de equilíbrio, então o sistema deve se aproximar do ponto de equilíbrio, que representa um estado de energia mínima. A função de energia para o sistema, que pode não ser única, é chamada de *função de Lyapunov*.

Cada uma destas técnicas convencionais de projeto de controle não-linear tem todos os inconvenientes associados às abordagens lineares. É por causa destas dificuldades que têm sido propostas alternativas para controle não-linear, como por exemplo o uso de redes neurais artificiais.

4.3 Técnicas de neuro-controle baseadas em dinâmica inversa

Muitas das aplicações de controle utilizando redes neurais, também denominadas neuro-controle, foram projetadas para conduzir à determinação da dinâmica inversa do sistema (KAWATO *et al.*, 1987; GOMI & KAWATO, 1993; ORTEGA & CAMACHO, 1996; STECK *et al.*, 1996). Esta abordagem é especialmente útil para plantas que apresentam dinâmica rápida.

4.3.1 Controle neural baseado em aprendizado por mímica

Aprendizado por mímica é um aspecto essencial de muitos sistemas biológicos. Trata-se da mesma abordagem adotada para treinamento supervisionado de redes neurais multicamadas utilizadas para imitar o comportamento de outros sistemas. O método a ser considerado aqui para desenvolver um controlador neural envolve a tentativa de copiar um controlador humano. WIDROW & SMITH (1963) aplicaram esta técnica para controlar um pêndulo invertido. Este método pode ser útil para plantas passíveis de serem controladas por operadores humanos e para as quais é difícil obter um controlador automático com base em técnicas tradicionais de projeto (HUNT *et al.*, 1992). O treinamento da rede neural consiste simplesmente em aprender a relação existente entre a informação sensorial recebida pelo operador humano e a entrada u de controle, conforme ilustrado na figura 4.1. O problema com este método ocorre quando é difícil determinar a informação usada pelo operador humano, considerado especialista no controle do sistema (AGARWAL, 1997).

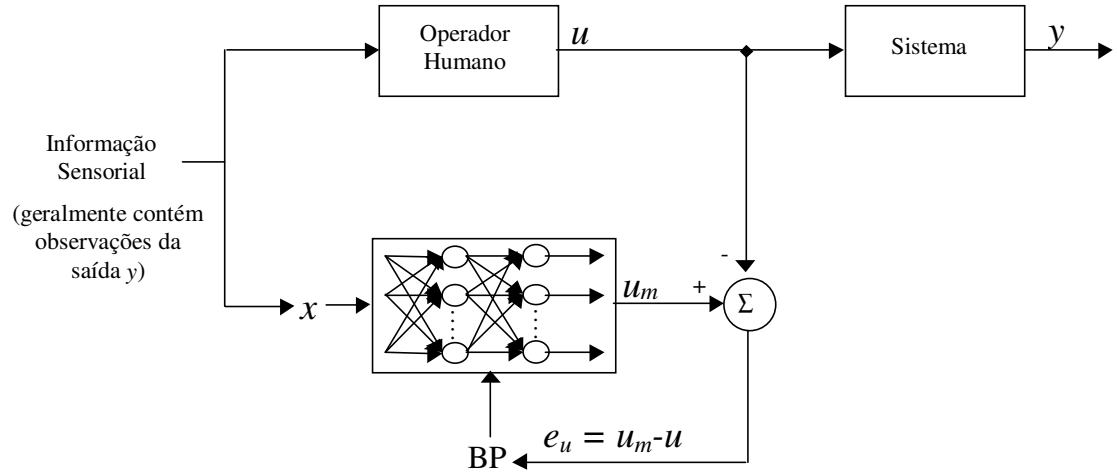


Figura 4.1 - Aprendizado de um controlador neural imitando um operador humano

Uma abordagem similar exige um controlador neural para imitar o comportamento de um outro controlador automático projetado com métodos convencionais, conforme ilustrado na figura 4.2. Esta técnica pode ser útil quando a implementação do controle automático requer muita computação, ou quando se deseja refinar o controlador posteriormente, através de aprendizado. Este refinamento, na grande maioria das vezes, é necessário pelo fato de que a entrada do controlador neural não leva em consideração a saída da planta.

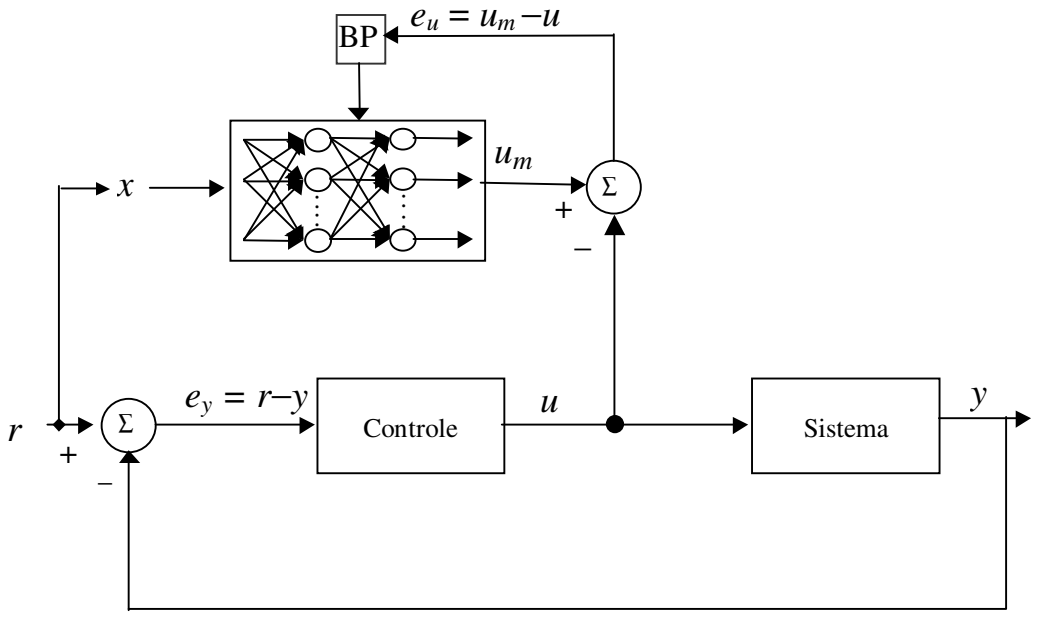


Figura 4.2 - Aprendizado de um controlador neural imitando um controlador realimentado

4.3.2 Controle neural direto-inverso

O objetivo do controle direto-inverso é controlar diretamente um sistema por uso da dinâmica inversa, isto é, o controlador neural de dinâmica inversa deve ser projetado de modo a produzir uma função de transferência controlador + sistema que se aproxime da função identidade, obtendo na saída a resposta desejada, a qual foi apresentada na entrada do controlador. Durante o aprendizado, a rede recebe a saída do sistema como entrada (figura 4.3). Alguns exemplos podem ser encontrado em WINDROW & STEARNS (1985), PSALTIS *et al.* (1988) e SANNER & AKIN (1990). Note que, embora na fase de treinamento haja caminho de realimentação entre a saída do sistema *y* e o vetor de entrada *x*, este sistema de controle neural vai operar em malha aberta, no sentido de que não vai levar em

consideração o erro entre a saída do sistema y e a saída desejada r ($e_y = r - y$) para decidir uma ação de controle conveniente. É evidente que não será tomada, pela rede neural, nenhuma ação para compensar qualquer desvio entre a saída do sistema e a saída desejada, o que pode ser desastroso caso este desvio seja cumulativo.

A principal dificuldade em aplicar este método de controle neural é a escolha do sinal de treinamento u. O sistema deve ser trazido para a região de operação onde o controlador irá atuar, o que exige alguns conhecimentos prévios a respeito do sistema a ser controlado. Este esquema será adotado nesta dissertação na fase de geração da estrutura de um controlador baseado em aprendizado construtivo, conforme será descrito no capítulo 7.

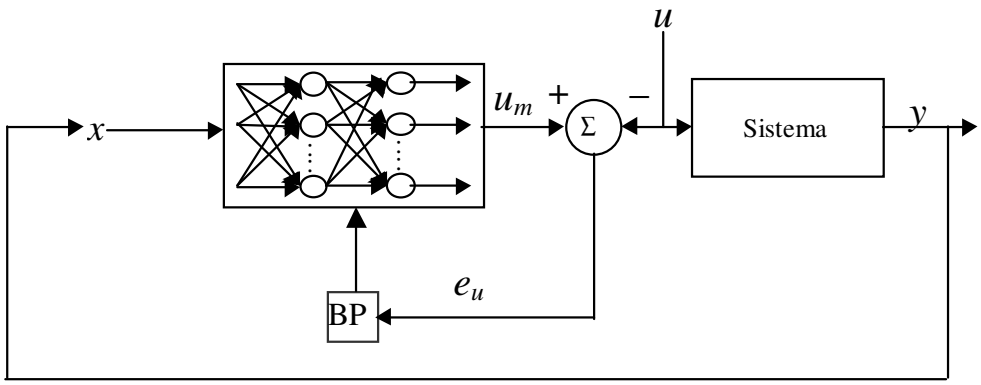


Figura 4.3 - Aprendizado de um controlador neural direto-inverso

4.3.3 Controle neural de inversa especializada

PSALTIS *et al.* (1988) propuseram um aprendizado de inversa especializada. Este é o objetivo da abordagem do controle neural direto. O principal aspecto é a diferença fundamental entre o aprendizado do controle de inversa especializada e o aprendizado do controle direto-inverso. A rede é treinada em tempo de atuação (*on-line*) para maximizar o desempenho de controle, conforme ilustrado na figura 4.4.

Este método pode resultar em uma solução inversa ótima se o comportamento de controle não conduzir o sistema para regiões de instabilidade na etapa de treinamento. Isto ocorre porque e_y não está diretamente relacionado com a saída do controlador, como deveria ser. Portanto, e_y pode ser totalmente descorrelacionado do erro do controlador neural $e_u = u_m - u$, sendo que o erro e_y pode ser de sinal inverso ou de grandeza completamente diferente do erro e_u. PSALTIS *et al.* (1988) argumenta que para aplicar este

método deve-se recorrer ao jacobiano do processo, devido à necessidade de trazer o sinal *y* para a saída da rede de modo a viabilizar o cálculo das derivadas do processo de treinamento. Entretanto, é visto que mesmo uma aproximação grosseira do erro e_u pode levar à convergência do aprendizado do controlador neural (ZHANG *et al.*, 1995). Na verdade, este método não pode ser generalizado para muitos sistemas, por não se ter controle sobre a qualidade da aproximação do erro e_u ou então devido à necessidade de se conhecer o jacobiano do processo.

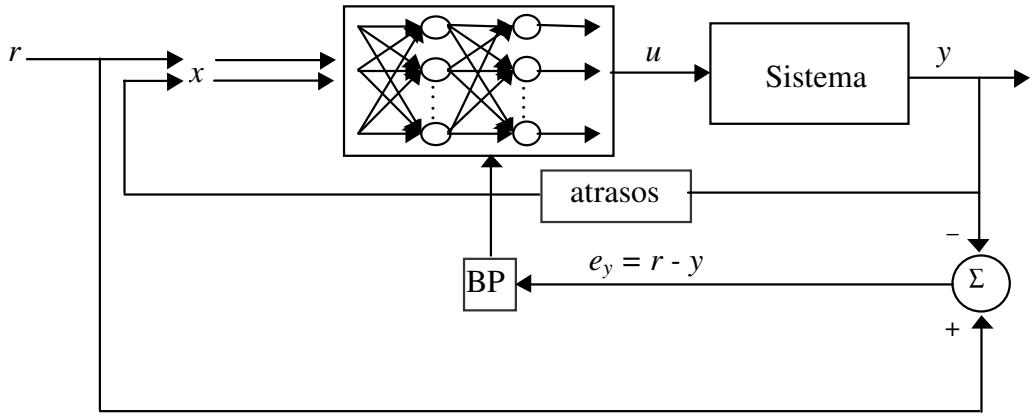


Figura 4.4 - Aprendizado de um controlador neural de inversa especializada

4.3.4 Controle neural através da retropropagação no tempo

Para evitar o problema apresentado acima para controle de inversa especializada, vários pesquisadores (NGUYEN & WIDROW, 1990; NARENDRA & PARTHASARATHY, 1990; JORDAN & RUMELHART, 1992) propuseram independentemente a retropropagação no tempo como método de aprendizado para controle neural. O problema do aprendizado do controle de inversa especializada é o fato da medida de desempenho $e_y = r - y$ não estar diretamente associada à saída do controle neural, já que o erro efetivo para o aprendizado do controle neural é $e_u = u - u_m$. Portanto NGUYEN & WIDROW (1990), NARENDRA & PATHARSARATHY (1990) e JORDAN & RUMELHART (1992) propuseram uma medida de erro baseada na representação do erro e_y propagado de volta através de um modelo direto do sistema (figura 4.5). Este caminho de retropropagação do erro e_y, em direção à saída do modelo do controle neural, representa uma boa estimativa do erro $e_u = u - u_m$, caso o modelo direto do sistema

tenha sido obtido com sucesso. Isto faz o aprendizado do controle possível em muitos casos.

Note que o modelo neural direto é treinado antes do aprendizado do controlador neural. Embora o erro e_y seja propagado de volta através do modelo direto, o último não é adaptado durante o aprendizado do controlador neural. Note que o erro e_y pode ser determinado usando a saída de um modelo direto, ao invés da saída do sistema. Este pode ser útil em situações em que existem dificuldades em se utilizar o próprio sistema dinâmico durante o treinamento do controle neural. Contudo, sempre que for viável deve-se utilizar a saída do próprio sistema, pois assim preserva-se a possibilidade de obter um modelo inverso de alta qualidade, mesmo que o modelo direto não reflita precisamente o comportamento do sistema dinâmico (JORDAN & RUMELHART, 1992).

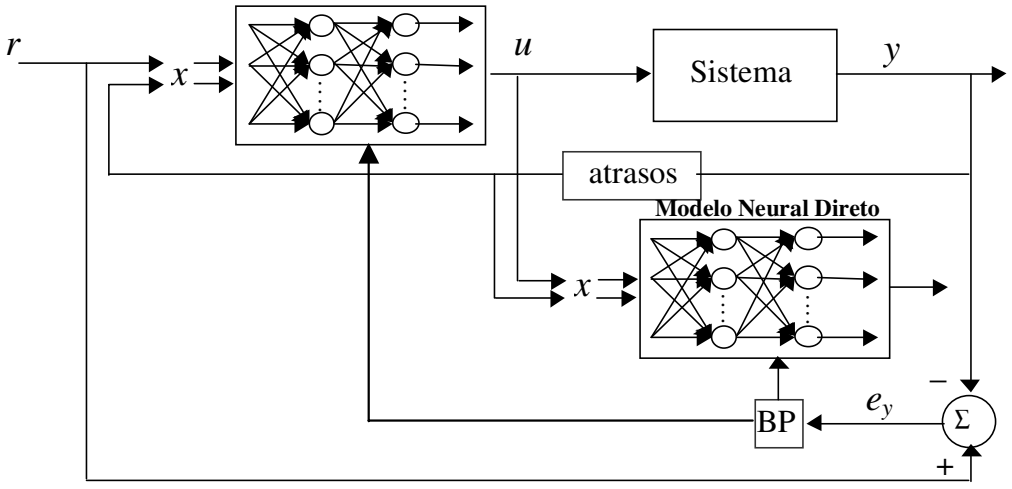


Figura 4.5 - Aprendizado de um controlador neural via retropropagação ao longo do tempo

A aproximação estudada por NARENDRA & PARTHASARATHY (1990), e mais tarde por GOMI & KAWATO (1993), é um pouco diferente daquela discutida há pouco. Um modelo de referência foi incorporado à estrutura para produzir um transiente na saída desejada do sistema y_d em relação a uma saída desejada r invariante no tempo, de tal forma a excitar todos os modos do sistema e, conseqüentemente, produzir um controlador com melhor desempenho (figura 4.6).

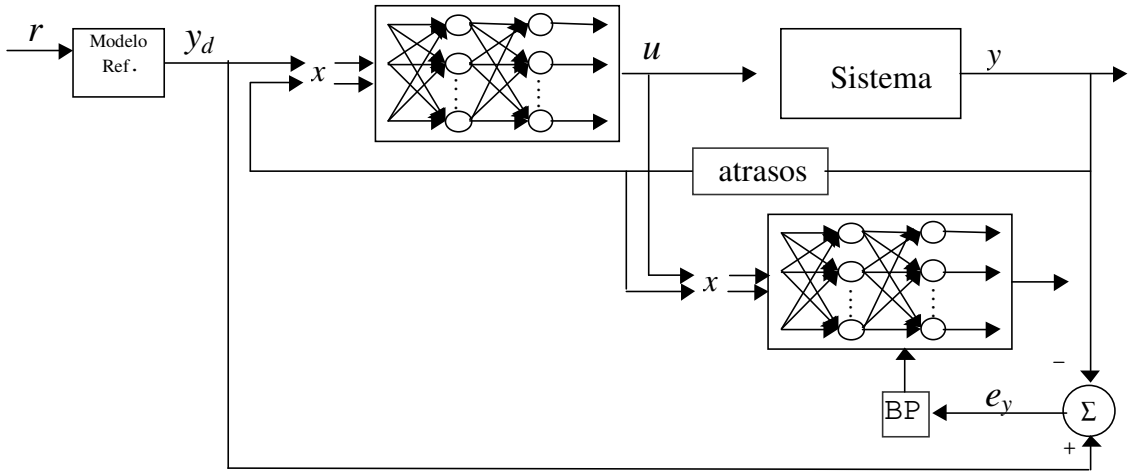


Figura 4.6 - Aprendizado de um controlador neural via retropropagação ao longo do tempo com a presença de modelo de referência

4.3.5 Controle realimentado e controle neural baseado no sistema inverso

Métodos de controle neural baseados no sistema inverso, embora já tenham apresentado resultados práticos interessantes, principalmente em robótica (MILLER *et al.*, 1990), continuam a manifestar sérios inconvenientes. O mais comum diz respeito à falta de realimentação. Se o controlador não é um modelo inverso preciso do sistema, este pode levar a ações de controle incoerentes, que conduzem a erros que não podem ser detectados e corrigidos sem realimentação. No entanto, não é trivial aprender completamente a dinâmica inversa de um sistema de ordem elevada e sob efeito de ruído, como ocorre em boa parte dos sistemas do mundo real (GORINEVSKY, 1993). Mesmo se obtivermos o verdadeiro modelo inverso do sistema, o controlador ainda assim pode ser muito sensível a distúrbios e atrasos no sistema. SLOTINE (1985) mostrou que, com um modelo preciso de um manipulador robótico, a trajetória média do erro do alvo desejado se degrada rapidamente quando a incerteza (isto é, o distúrbio) aumenta, com o sistema eventualmente tornando-se instável. Um último problema do controle inverso a ser levantado aqui, mas nem por isso menos importante, é que nem todo sistema é inversível. Estes problemas restringem fortemente a amplitude de aplicação do método de controle neural baseado no sistema inverso.

Fica então devidamente justificado o emprego de realimentação junto à malha de controle, quando isto for possível. No entanto, quando a realimentação é introduzida como

entrada do controlador o risco de instabilidade cresce. Isto implica que resultados mais abrangentes ainda são necessários para desenvolver esquemas de aprendizado de controle realimentado.

Em breve, os sistemas de malha fechada deverão ser empregados para realizar ações de controle mais precisas. Sistema de malha fechada são mais interessantes quando o processo é lento e a regulação é importante (SCHMIT, 1982). Entretanto, sistemas de malha aberta, sendo essencialmente servo-mecanismos, deverão ser aplicados pelo menos para controle de sistema de movimento rápido, tal como movimento de braços em robótica (SCHMIT, 1982; CARMON, 1986; TOASTES, 1975; GORINEVSKY, 1993; BERTHIER *et al.*, 1993; KAWATO, 1989).

4.3.6 Controle preditivo baseado em modelo neural

Controle preditivo tem sido principalmente desenvolvido para vencer o problema do atraso presente em muitas das plantas do mundo real. Estas plantas nunca irão reagir imediatamente à ação da entrada.

O propósito do controle de realimentação convencional é agir sobre a planta de acordo com o erro de controle (equação 4.1), no sentido de fazer o sistema y mover-se seguramente em direção à resposta desejada r.

$$e = y - r \tag{4.1}$$

Agora, se supusermos que há um tempo de atraso de 1 segundo no sistema, é claro que o erro de controle (equação 4.1) não irá ser afetado no primeiro segundo da ação do controle. Assim, o controlador irá agir do mesmo modo na segunda ação de controle. Para evitar este comportamento indesejado, uma solução seria esperar um segundo e então ativar o sistema com o sinal de controle. Isto não pode ser realizado no caso de requisitos de controle rápido. Então, uma outra solução é usar um modelo do sistema para predizer qual será o estado do sistema um segundo à frente, e então determinar o erro de controle de acordo com a saída do modelo, e não em relação à saída atual do sistema. Esta é a idéia básica do controle preditivo: o controle é realizado de acordo com a saída predita da planta, dada por seu modelo direto de predição.

Esta idéia foi generalizada por CLARKE *et al.* (1987) para lidar com plantas com dinâmica complexa (isto é, sistemas com inversa instável, tempo de atraso variante no tempo, etc.), e modelos de plantas com com erros (RUSNAK *et al.*, 1996). CLARKE *et al.* (1987) chamaram este modelo de controlador como controle preditivo generalizado (*GPC - Generalized Preditive Control*). O *GPC* é o principal método de projeto de controlador preditivo baseado em modelo (*MBPC*). Um *MBPC* possui três principais componentes: o sistema, seu modelo e uma função de otimização (figura 4.7). O modelo é usado para predizer comportamentos futuros da planta. De acordo com o comportamento da planta, a função de otimização define a seqüência requerida de ações u para fazer o sistema comportar-se como desejado. Esta função toma a forma de uma função de custo quadrática, tal como:

$$J(N_1, N_2, N_3, t) = \sum_{i=1+N_1}^{t+N_2} \mu(i)\left[y_d(i) - \hat{y}(i)\right]^2 + \sum_{i=t}^{t+N_u} \lambda(i)\Delta u(i-1)^2 \qquad (4.2)$$

onde y_d é a saída desejada que pode ou não variar no tempo, $\hat{y}$ é a saída do modelo, $\Delta u(i) = u(i) - u(i-1)$ é o incremento da entrada do processo, N_1 e N_2 definem o horizonte de predição, N_u é o horizonte de controle e λ, usualmente uma constante no intervalo [0,+1], dá o peso do comportamento futuro. Uma escolha conveniente para N_1 é fazê-lo igual ao tempo de atraso do processo. Este esquema de controle pode ser pensado como sendo um modelo de malha aberta, já que a saída da planta não é requerida. Entretanto, a saída da planta é usada para reiniciar a dinâmica do modelo. Isto é importante, pois deve ser considerado que, depois de um tempo, devido à discrepância no modelo ou a perturbações externas, o comportamento da planta irá divergir do comportamento do modelo.

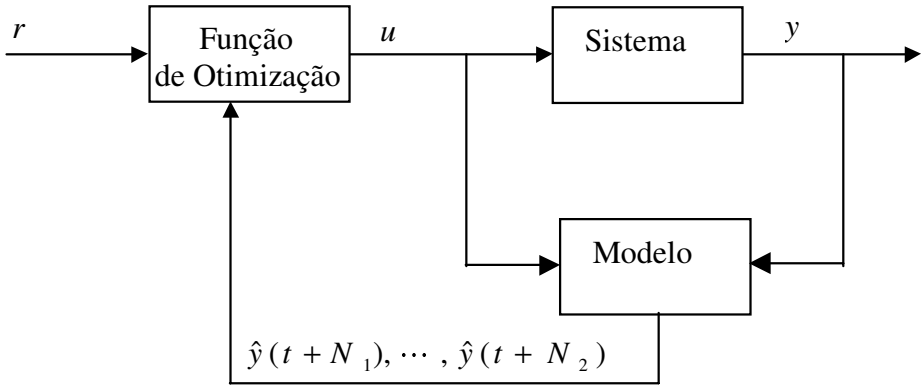


Figura 4.7 - Arquitetura geral de um modelo baseado em controlador preditivo

Uma importante característica deste esquema é que nenhum projeto de controlador é requerido, somente o modelo direto da planta irá afetar a eficiência do *MBPC*. Assim, a aplicação de redes neurais neste contexto é direta. Atualmente, o problema de não ser possível estabelecer uma interpretação física para o modelo (quando este é construído com base em redes neurais) tem um impacto reduzido neste esquema, já que é mais importante aqui a precisão e robustez do modelo.

No contexto de *MBPC*, redes neurais têm sido aplicadas geralmente para modelagem do sistema (RUSNAK *et al.*,1996; TEMENG *et al.*, 1995; GOMM *et al.*, 1997; PARK & CHO, 1995; MILLS *et al.*, 1994). Resulta então um controlador preditivo baseado num modelo neural (figura 4.8). Como um outro exemplo, em ORTEGA (1996) uma rede neural é usada para implementar a função de custo e assim aumenta a velocidade de computação para o problema de acompanhamento de trajetória em robótica.

4.4 Controle adaptativo de sistemas não-lineares

Supondo que a classe de modelos descritos no capítulo 4 é suficientemente eficaz, isto é, produz uma aproximação do sistema em questão, tais modelos podem ser usados para projetar controladores para plantas não-lineares desconhecidas. Se o processo de identificação é realizado *on-line*, o procedimento é geralmente denominado controle adaptativo indireto.

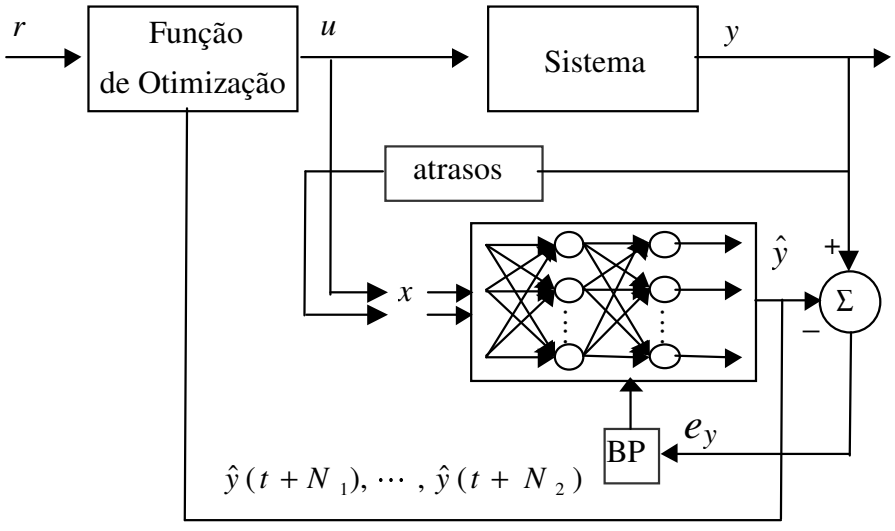


Figura 4.8 – Aprendizado de uma rede neural em uma arquitetura para controle preditivo

Como em problemas de identificação discutidos no capítulo 3, ambos os modelos por espaço de estados e entrada-saída têm sido usados para controle. O modelo por espaço de estados é uma escolha natural quando o vetor de estados da planta é acessível, mas torna-se computacionalmente intensivo quando somente medidas de entrada-saída podem ser feitas. Por isso, no último caso, a ênfase será sobre modelos de entrada-saída.

4.4.1 Controle usando representação por espaço de estados

Seja um sistema dinâmico representado pela equação

$$x(k+1) = f[x(k), u(k)] \quad x(0) = 0 \tag{4.3}$$

onde $x(k) \in \Re^n, u(k) \in \Re^m$, e $f : \Re^n \times \Re^m \rightarrow \Re^n$ é uma função suave.

Definição 4.1

A equação de estados (4.3) é dita ser controlável no instante t_0, se há um $t_1 > t_0$ tal que para quaisquer x_0 e x_1 no espaço de estados Ξ, existe uma entrada $u_{[t_0,t_1]}$ que irá transferir o estado x_0 em $t = t_0$ para o estado x_1 em $t = t_1$. Caso contrário, a equação de estados é dita ser não controlável no instante t_0.

Comentários: Esta definição requer somente que a entrada $\underline{u}$ seja capaz de mover qualquer estado no espaço de estados para qualquer outro estado em um tempo finito: qual a trajetória que o estado deverá tomar não é especificada. Além disso, não há restrição imposta sobre a entrada. Sua magnitude pode ser tão grande quanto desejável.

Definição 4.2

A equação de estados (4.3) é dita ser (completamente) observável em t_0 se existe um $t_1 > t_0$ finito tal que para qualquer estado x_0 em $t = t_0$, o conhecimento da entrada $u_{[t_0,t_1]}$ e da saída $y_{[t_0,t_1]}$ basta para determinar o estado x_0. Caso contrário, a equação (4.3) é dita ser não observável em t_0.

Comentários: O conceito de observabilidade é dual àquele de controlabilidade. Grosseiramente falando, controlabilidade estuda a possibilidade de dirigir o estado a partir da entrada, enquanto que observabilidade estuda a possibilidade de estimar o estado a partir da entrada. Se uma equação dinâmica é controlável, todos os modos da equação podem ser

excitados a partir da entrada; se uma equação dinâmica é observável, todos os modos da equação podem ser observados a partir da saída.

Após revermos estes conceitos, vamos tentar resolver o primeiro problema, que neste caso é estabilizar o sistema em torno da origem (a qual é um estado de equilíbrio) usando realimentação. Isto implica na determinação de $u(k) = g\left[x(k)\right]$ tal que o estado de equilíbrio do sistema seja assintoticamente estável. Este problema foi considerado em detalhes em LEVIN & NARENDRA (1993). Se um sistema linear é controlável, então pela definição 4.1 qualquer estado do sistema pode ser transferido para qualquer outro estado em um número finito de passos (NARENDRA & MUKHOPADHYAY, 1996). LEVIN & NARENDRA (1993) usaram este fato para demonstrar que se a linearização da equação (4.3) em torno da origem é controlável, existe uma região contendo a origem na qual o sistema não-linear pode ser estabilizado usando um controlador realimentado na forma $u(k) = g\left[x(k)\right]$. Assim, a existência de tal controlador é admitida e ele pode ser realizado usando redes neurais, como descrito em LEVIN & NARENDRA (1993).

Condições necessárias e suficientes para transformar um sistema não-linear descrito pela equação (4.3) em um sistema linear controlável invariante no tempo $z(k+1) = Az(k) + Bu(k)$, usando uma transformação de coordenadas $z(k)=T[x(k)]$ e uma transformação da entrada $u(k)=g[x(k)]$, são comumente conhecidas. Esta transformação é referida como linearização por realimentação. Se há informação disponível tal que as transformações T e g existem, então estas transformações podem ser aproximadas usando redes neurais. As redes neurais N_T (responsável pela aproximação da transformação T) e N_g (responsável pela aproximação da função g) podem ser treinadas usando dados obtidos da linearização por realimentação. Este problema foi considerado em LEVIN & NARENDRA (1993), onde exemplos na forma de estudos de simulação são apresentados. Uma discussão mais extensiva da linearização por realimentação e sua obtenção utilizando redes neurais pode ser encontrado em (CABRERA & NARENDRA, 1993).

4.4.2 Controle usando representação de entrada-saída

Uma conseqüência da definição 4.2 é que um sistema é observável se o estado do sistema para um instante k pode ser calculado a partir do conhecimento das saídas

$y(k-l), \quad l = 1,\ldots$. Devido ao fato de que um sistema que é controlável na origem pode ser estabilizado usando realimentação de estados, segue que ele pode ser estabilizado usando valores passados da entrada e saída desde que seja observável. Dado um modelo de entrada-saída da planta, isto é,

$$y(k+1) = f[y(k), \cdots, y(k-n+1), u(k), \cdots, u(k-m+1)] \tag{4.4}$$

a entrada de controle estabilizante tem a forma

$$u(k) = g[y(k), \cdots, y(k-n+1), u(k-1), \cdots, u(k-m+1)] \tag{4.5}$$

Alternativamente, se o sinal da saída desejada é especificado como $y_d(k+1) = r(k)$, a entrada de controle pode ser generalizada para

$$u(k) = \overline{g}[y(k), \cdots, y(k-n+1), u(k-1), \cdots, u(k-m+1), r(k)] \tag{4.6}$$

tal que $\lim_{k\to\infty} |y(k) - y_d(k)| = 0$. As funções g e $\overline{g}$, por sua vez, podem ser aproximadas usando redes neurais.

A estrutura global do sistema usando redes neurais para identificação e controle de uma planta, dada em LEVIN & NARENDRA (1993) e utilizada nesta dissertação, é estudada no capítulo 7, e está apresentada na figura 4.9. Na figura 4.9, N_i é uma rede neural usada no modelo de identificação. Ela tem como suas entradas $y(k), \cdots, y(k-n+1)$ e $u(k), \cdots, u(k-m+1)$. Seus parâmetros são ajustados, tanto na abordagem clássica quanto na construtiva baseada na teoria de agentes (ver capítulo 6), com base no erro de identificação $e_i(k+1) = \hat{y}(k+1) - y(k+1)$. O controlador é também realizado por uma rede neural (clássica ou construtiva baseada na teoria de agentes), denotado por N_c, cujas entradas são $y(k), \cdots, y(k-n+1), u(k-1), \cdots u(k-m+1)$ e a entrada de referência *r(k)*. A saída desejada é obtida de um modelo de referência assintoticamente estável cuja entrada é *r(k)*. Numa forma simples, o modelo de referência pode ser somente um atraso, isto é, $y_d(k+1) = r(k)$. O treinamento do controlador N_c é realizado utilizando dois estágios, conforme será descrito no capítulo 7, de tal forma a minimizar o erro de controle $e_c(k+1) = y(k+1) - y_d(k+1)$. Por causa da dinâmica da planta ser considerada desconhecida, o modelo de identificação é usado no lugar da planta para propósito de cálculo do gradiente com respeito aos parâmetros do controlador N_c.

Na discussão acima, é suposto que *u(k)* afeta as saídas da planta para o instante de tempo *k*+1 (isto é, o atraso do sistema é unitário). Se o atraso do sistema é *d*, a saída desejada para o instante de tempo *k*+*d* deve ser conhecida para o instante de tempo *k*. Neste caso, $y_d(k+d) = r(k)$ e a função de controle será a mesma dada pela equação (4.6).

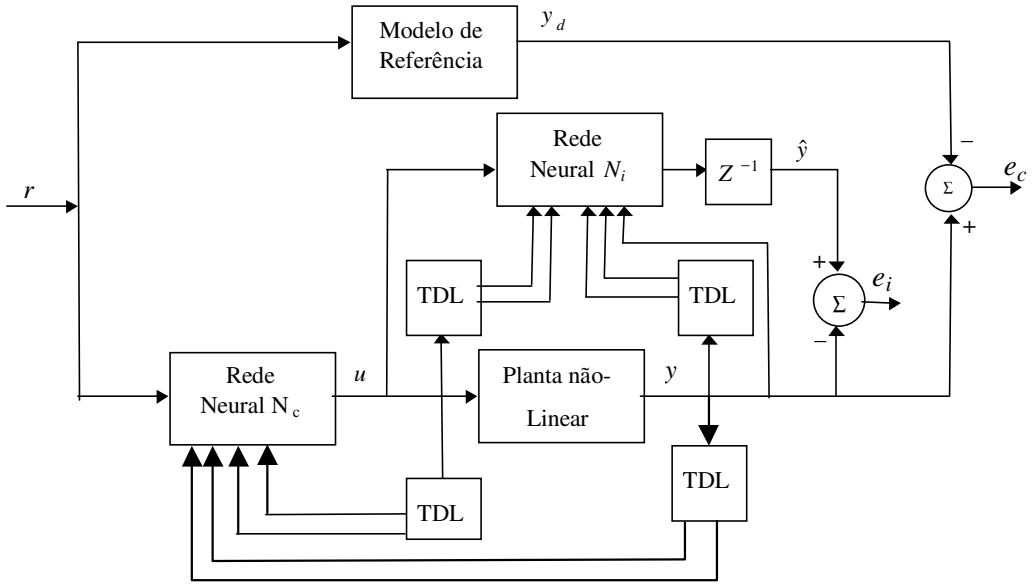


Figura 4.9 Controle adaptativo indireto usando Redes Neurais em malha fechada . *TDL* - linhas de retardo (*tapped delay line*)

Os valores de erro e_i e e_c serão empregados, respectivamente para o ajuste de pesos das redes neurais N_i e N_c.

4.5 Aprendizado de um controle neural baseado em computação evolutiva

Neuro-controle envolve o uso de redes neurais, recorrentes ou não-recorrentes, para o propósito de controle. Sob certas circunstâncias, pode não ser possível o uso de abordagens clássicas de otimização e ajuste de parâmetros para treinamento de neuro-controladores. Por exemplo, controlar sistemas não-lineares instáveis usando redes neurais recorrentes freqüentemente leva o processo de ajuste à instabilidade, pois não vai haver oportunidade do controlador aprender inicialmente a estabilizar a planta. Em alguns casos, pode ser possível apenas calcular o erro para propagar de volta pelas camadas da rede, de

modo que uma busca tradicional no espaço dos pesos, baseada por exemplo no gradiente descendente, torna-se impraticável. Computação evolutiva desponta, então, como uma técnica alternativa para a busca potencial de um conjunto ótimo de valores para todos os parâmetros a determinar envolvidos no processo. Em FOGEL (1995), foi realizado o estudo de dois sistemas de controle empregando computação evolutiva, e os resultados indicam que a evolução de neuro-controladores a partir de técnicas de computação evolutiva é bastante promissora. Embora este resultado tenha sido baseado apenas em arquiteturas fixas de redes não-recorrentes, tal restrição não necessita ser mantida junto ao método proposto. Em relação a uma dada topologia que soluciona um dado problema, a computação evolutiva pode ajustar simultaneamente os pesos e a arquitetura da rede. Esta abordagem proporciona um método para projeto de redes de dimensões adequadas para solucionar o problema, evitando assim a sobre-parametrização do modelo neural. Além disso, a mesma técnica pode ser aplicada sem variações significativas para redes recorrentes. Em ANGELINE & FOGEL (1998), é descrito um algoritmo evolutivo que produz um conjunto de expressões simbólicas como modelo para plantas desconhecidas. O sistema de expressões é avaliado como um conjunto de equações a diferenças que especifica o comportamento da planta no tempo. Cada expressão individual é uma função arbitrária da entrada e de cálculos resultantes de outras equações do sistema. Como tipicamente é o caso em aplicações envolvendo técnicas de computação evolutiva, uma população de soluções é evoluída em paralelo, usando uma metodologia inspirada em processos de seleção natural (HOLLAND, 1992).

4.6 Visão sintética do capítulo

Neste capítulo, revemos os métodos tradicionais de controle não-linear, e apresentamos as várias arquiteturas de controle utilizando redes neurais. Estas arquiteturas serão utilizadas no capítulo 7, para aplicação em controle.

Capítulo 5

Modelos baseados em projeção e modelos não-paramétricos

5.1 Introdução

Alguns dos métodos mais populares de modelagem no campo da engenharia realizam uma transformação nos dados de entrada, projetando-os sobre um hiperplano, antes da aplicação de funções básicas de transformação não-linear (por exemplo, tangente hiperbólica no caso de redes neurais). Dentre estes métodos, destacam-se os de projeção linear:

- regressão ordinária dos quadrados mínimos (*OLS – ordinary least-square regression*);
- regressão parcial dos quadrados mínimos (*PLS – partial least-square regression*);
- regressão de componentes principiais (*PCR – principal component regression*);
- regressão de expansão ortogonal (*RR - ridge regression*);
- redes neurais artificiais não-recorrentes com uma camada intermediária(*ANN – Artificial Neural Network*);
- regressão por busca de projeção (*PPR - projection pursuit regression*);
- aprendizado por busca de projeção (*PPL - projection pursuit learning*);

e projeção não-linear, tais como:

- regressão dos quadrados mínimos parciais não-lineares (*NLPLS – nonlinear partial least-square regression*);
- regressão de componentes principiais não-lineares (*NLPCR –nonlinear principal component regression*).

Estes métodos têm sido utilizados no desenvolvimento de modelos para uma grande variedade de problemas, tais como identificação e controle de processos. Todos os métodos baseados em projeção linear relacionam a entrada à saída na forma:

$$\hat{y}_k(x) = \sum_{m=1}^{p} c_{mk} f_m \left(\sum_{j=1}^{N_0} v_{jm} x_j \right) \tag{5.1}$$

onde x_j $(j=1,...,N_0)$ são as entradas ou variáveis do preditor, v_{jm} são os coeficientes da combinação linear ($v_m \in R^{N_0}$ é a direção de projeção), f_m são as funções de ativação ou funções básicas, c_{mk} são os coeficientes de regressão, $\hat{y}_k$ $(k=1,...,N_s)$ são as saídas aproximadas ou variáveis de resposta, e p é o número de coeficientes de regressão e funções básicas (para maiores detalhes da notação utilizada, veja capítulo 2)

A grande diversidade de métodos baseados em projeção linear é devida aos diferentes tipos de funções básicas e critérios de otimização usados para determinar os parâmetros do modelo. As decisões sobre os tipos de função básica e critérios de otimização determinam o desempenho de cada método para diferentes problemas de modelagem. Por exemplo, considerando funções básicas lineares para os métodos *OLS, PLS e PCR* resultam modelos lineares com ajuste de parâmetros eficiente e de fácil interpretação. Já os métodos *OLS, ANN e PPR* fornecem uma melhor representação quando uma grande quantidade de dados de treinamento está disponível, ao passo que os métodos *PLS,* e *PCR* fornecem uma melhor representação quando a razão dos dados de treinamento para o número de entradas é grande (baixa dimensionalidade dos dados) e quando as entradas estão correlacionadas.

Na tentativa de eliminar a seleção arbitrária dos métodos de modelagem para uma dada aplicação e combinar as propriedades dos métodos existentes, alguns pesquisadores propuseram uma metodologia para unificar os métodos baseados em projeção. Esta unificação tem como objetivo obter uma melhor representação dos vários tipos de relacionamentos existentes nos dados de entrada e saída, levando em consideração a quantidade de dados de treinamento e a natureza da correlação entre as entradas.

Os critérios de modelagem para *OLS, PLS e PCR* diferem principalmente na forma de capturar os relacionamentos entre as entradas, o qual apresenta uma maior sofisticação no caso do *OLS* quando comparado com *PLS* e *PCR*. Baseado nisto, STONE & BROOKS

(1990) unificaram estes métodos de modelagem desenvolvendo uma função-objetivo comum, em que a seleção de um valor apropriado do parâmetro de ajuste possibilita a obtenção dos métodos *OLS, PLS* ou *PCR*. Uma vez que este novo parâmetro ajustável pode assumir qualquer valor contínuo numa faixa determinada, o método unificado é chamado de *regressão contínua (CR – continuum regression*). Técnicas similares foram também desenvolvidas por LOBER *et al*. (1987), os quais forneceram uma estrutura formal para *PLS*. O parâmetro ajustável do método *CR* é semelhante ao efeito do número de funções básicas sobre o grau de sobreajuste ou bias do modelo, proporcionando controle adicional sobre sua qualidade. A técnica *CR* tem sido estendida para incluir *RR* (*ridge regression*) (SUNDBERG, 1993; DE JONG & FAREBROTHER, 1994) e seu desempenho para modelagem de sistemas dinâmicos foi estudado por WISE & RICKER (1993).

Apesar destas iniciativas, as pesquisas sobre a unificação de métodos de modelagem baseados em projeção têm levado a resultados muito limitados. Uma abordagem geralmente sugerida para combinar os aspectos de métodos baseados em redes neurais e métodos estatísticos é a determinação dos valores iniciais dos pesos da camada de entrada (que fazem o papel da direção de projeção) da rede neural pelo *PCA* ou *PLS* e, a partir daí, adaptando os parâmetros e os pesos para minimizar o erro de predição da saída (PIOVOSO & OWENS 1986; MARTIN *et al*., 1995). Esta abordagem objetiva combinar a habilidade do *PCA* ou *PLS*, para proporcionar melhores modelos com dados limitados, com a propriedade de aproximação universal da rede neural. Várias abordagens de regressão *NLPLS* combinam *PLS* com redes neurais, empregando *splines* e outros suavizadores estatísticos (HOLCOMB & MORARI, 1992; WORLD, 1992; FRANK, 1990). Um outro tipo de método *CR*, que define uma função-objetivo para combinar os benefícios da modelagem *PLS* com *PPR*, é descrito por HAARIO & TAAVITSAINEN (1994). Este método não enfatiza a unificação dos métodos baseados em projeção linear, já que as funções básicas possuem forma fixa e, além disso, redes neurais artificiais não fazem parte desta abordagem. BARRON & BARRON (1988), SJÖBERG *et al*. (1995) e FRANK (1995) propuseram alguns métodos de modelagem empírica não-linear, não enfatizando a unificação destes.

BASKHI & UTOJO (1998) propuseram uma técnica de modelagem que unifica todos os métodos lineares e não-lineares baseados em projeção. Esta unificação é baseada na compreensão fornecida por uma estrutura comum para todos os métodos de modelagem. O

método resultante unifica *OLS, PLS, PCR, RNA* com uma camada intermediária*, PPR, NLPLS* e *NLPCR*. Este método estende o princípio do método *CR* para modelagem não-linear e é denominado *regressão contínua não-linear* (*NLCR*). Funções de ativação com forma variável no espaço de projeção de entrada-saída são determinadas por técnicas suavizantes univariáveis tais como: variável de extensão suave, *splines*, polinômios de Hermite e outras expansões funcionais.

Na modelagem *NLCR*, existe um novo parâmetro ajustável γ, que determina o melhor modelo de acordo com os dados. Conseqüentemente, técnicas eficientes para encontrar o melhor valor de γ são essenciais para aplicação do método *NLCR* a problemas práticos de modelagem. Este valor de γ pode ser encontrado determinando alguns modelos para diferentes valores de γ entre 0 e 1 e selecionando o valor de γ e número de funções básicas que resultam em menor erro de aproximação para dados de teste. Infelizmente, a natureza não-linear do modelo pode fazer com que esta abordagem seja computacionalmente cara, principalmente quando a quantidade de dados de treinamento, dimensão do espaço de entradas e o número de funções básicas aumentam. Além disso, a não-convexidade da superfície de erro pode tornar necessária a utilização de diferentes valores iniciais dos parâmetros para evitar um mínimo local insatisfatório.

Redes neurais artificiais empregam uma técnica de projeção linear, e recentemente têm atraído a atenção de muitos pesquisadores. Normalmente, o método do gradiente (baseado na retropropagação) executa o gradiente descendente somente nos espaços de pesos da rede com arquitetura fixa. Esta abordagem é análoga àquelas adotadas por técnicas de regressão paramétrica não-linear em estatística. Em geral, estes procedimentos paramétricos são úteis somente quando a estrutura (isto é, o modelo) da rede é escolhido adequadamente. Uma rede neural de dimensão muito reduzida não pode representar bem o problema, mas com uma rede de dimensão muito elevada, tem-se problemas de generalização e sobre-ajuste, implicando em uma perda de desempenho. Por isso, estudos recentes têm sido realizados, validados e implementados no sentido de otimizar também a dimensão da rede neural para cada aplicação, em função da complexidade inerente do problema. Esta complexidade é estimada apenas a partir das informações disponíveis, geralmente restritas a dados de entrada-saída.

Há duas abordagens gerais para este problema de otimização. Uma define o início do processo com base em uma rede neural de dimensão maior que o necessário, seguida por etapas de poda de neurônios que não estejam contribuindo de modo significativo para a solução. Métodos usando esta abordagem são chamados ***métodos de poda*** (HASSIBI & STORK,1993; KARNIN,1990; LECUN *et al.*,1990; MOZER & SMOLENSKY, 1989; REED, 1993). A outra abordagem corresponde aos ***métodos construtivos*** (ASH, 1989; FAHLMAN & LEBIERE, 1990; HIROSE *et al.*, 1991; HWANG *et al.*, 1994; SJOGAARD, 1991; YEUNG, 1993), que partem de uma pequena rede, à qual são adicionadas unidades ocultas e pesos até uma solução satisfatória ser encontrada.

Nas seções seguintes, será realizada uma revisão dos métodos de modelagem baseados em projeção linear, enfatizando suas principais diferenças sobre três aspectos: natureza da transformação da entrada, tipo de função de ativação e critério de otimização para determinação dos parâmetros do modelo. Em seguida, serão apresentados os *métodos construtivos* e *os métodos de poda* para redes neurais, ressaltando as vantagens e desvantagens de cada um.

5.2 Estrutura dos métodos de modelagem usando projeção linear

Os modelos determinados pelos métodos baseados em projeção linear podem ser representados como uma soma ponderada de funções básicas

$$\hat{y}_k = \sum_{m=1}^{p} c_{mk} f_m(\phi_m(v; x_1, x_2, \cdots, x_{N_0})) \tag{5.2}$$

onde v é uma matriz de parâmetros das funções básicas e ϕ_m *(m = 1,...,p)* representam as projeções lineares da entrada. As entradas transformadas são também chamadas variáveis latentes e representadas como:

$$z_m = \phi_m(v; x_1, x_2, \cdots, x_{N_0}), m = 1,..., p \tag{5.3}$$

O espaço de entrada refere-se ao espaço a que pertencem as variáveis de entrada, enquanto que o espaço transformado de entrada ou espaço de projeção refere-se ao espaço das variáveis latentes. O modelo dado pela equação (5.2) pode também ser representado por

uma rede neural, onde *v* representa um conjunto de pesos da camada de entrada, $C = \{c_{mk}\}$ representa um conjunto de pesos da camada de saída e $f_m, (m = 1,..., p)$ representam as funções de ativação. A especificação de métodos de modelagem pode ser derivada da equação (5.2) a partir da decisão sobre a natureza da transformação, tipos de função de ativação, ou função básica, e critério de otimização. A equação (5.2) representa um modelo que envolve múltiplas entradas e múltiplas saídas. Se modelos separados são desenvolvidos para cada saída, as funções básicas e as direções de projeção podem ser diferentes para cada saída. Caso contrário, as funções básicas e direções de projeção serão as mesmas para todas as saídas, mudando apenas os coeficientes da combinação linear associada a cada saída.

5.2.1 Natureza da transformação da entrada

A complexidade da tarefa de modelagem e a quantidade de dados de treinamento requerida para uma aceitável qualidade do modelo aumentam significantemente com o aumento do número de variáveis de entrada. Técnicas de modelagem combatem a *maldição da dimensionalidade* transformando as entradas em variáveis latentes que capturam o relacionamento entre entrada e saída, sendo que as variáveis latentes são em menor número que as variáveis de entrada. Tal redução de dimensionalidade é, usualmente, acompanhada pela exploração dos relacionamentos entre as entradas que sejam relevantes na predição da saída. Assim, os métodos de transformação podem ser divididos dentro de três categorias, dependendo das transformações da entrada:

- *Métodos baseados em projeção linear:* exploram o relacionamento linear entre as entradas, projetando-as em um hiperplano antes da aplicação das funções básicas. Serão os únicos métodos a serem tratados em detalhes neste capítulo.
- *Métodos baseados em projeção não-linear:* exploram o relacionamento não-linear entre as entradas projetando-as sobre uma hipersuperfície, resultando em variáveis latentes que são funções não-lineares da entrada. Se as entradas são projetadas sobre uma hipersuperfície na forma de uma hiperesfera ou hiperelipse, então as funções básicas são locais;
- *Métodos baseados em partição:* combatem a *maldição da dimensionalidade* pela seleção de variáveis de entrada que são mais relevantes para uma modelagem

eficiente. O espaço de entradas é particionado por hiperplanos que são perpendiculares pelo menos a um dos eixos associados a variáveis de entrada.

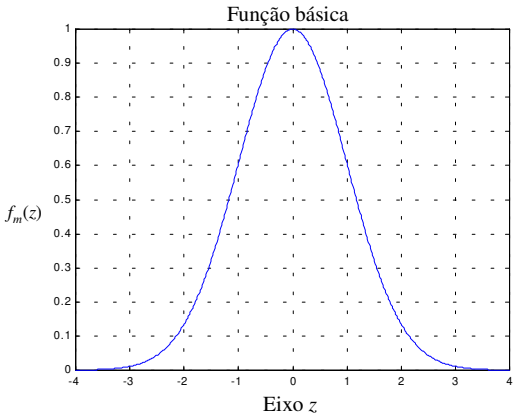
5.2.2 Tipo de função de ativação

A grande variedade de funções de ativação que podem ser empregadas em métodos de modelagem são em geral divididas em duas categorias, dependendo da forma ser fixa ou adaptativa. As funções de ativação, $f_m(z_m)$ ($m = 1,...,p$) relaciona a entrada transformada $z_m = \phi_m(v; x_1,...,x_{N_0})$ à saída e são normalmente unidimensionais. Se a forma for variável, os parâmetros da função de ativação e o tipo de transformação da entrada determinam a natureza das funções básicas $f_m(z_m)$ ($m = 1,...,p$), responsáveis pelo relacionamento entre entrada e saída.

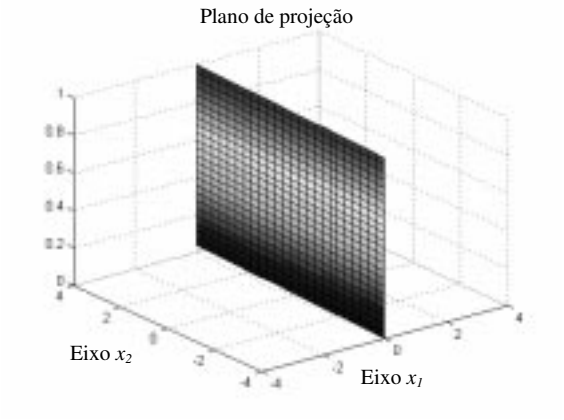
Quando as funções $f_m(z_m)$ ($m = 1,...,p$) são constantes para os valores de x em hiperplanos do $\Re^{N_0}$, elas são denominadas funções de expansão ortogonal a uma determinada direção v_m – *ridge function* (DAHMEN & MICCHELLI, 1987). Considerando $N_0 = 2$ e $f_m(z)$ arbitrária, as figuras 5.1 (a) e 5.1(b) permitem verificar esta propriedade. A expansão é ortogonal à direção de projeção $v_m = [0\ 1]^T$.

As funções de ativação $f_m(z_m)$ ($m = 1,...,p$) de forma fixa, tais como, linear, sigmóide, gaussiana, wavelet ou senoidal, são geralmente usadas em alguns métodos de modelagem. Ajustando-se os parâmetros c_{mk} e ϕ_m na equação (5.2), muda-se a localização, tamanho e orientação de f_m, no entanto sua forma permanece fixa. Alguns métodos de modelagem relaxam a necessidade de forma fixa e permitem as funções básicas adaptarem suas formas aos dados de treinamento, além da localização, tamanho e orientação.

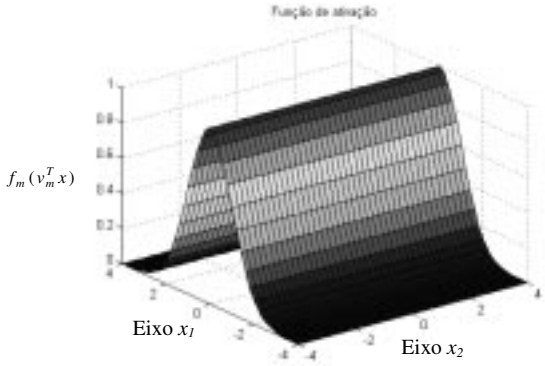
Este grau adicional de liberdade proporciona maior flexibilidade na determinação da superfície desconhecida de entrada-saída e freqüentemente resulta em modelos mais compactos. As funções básicas de forma adaptativa são obtidas através da aplicação de técnicas suavizantes tais como: *splines*, variáveis de extensão suave e polinômios de Hermite para aproximar a superfície transformada de entrada-saída.



(a) $f_m(z) = e^{-0,5z^2}$



(b) $f_m(v_m^T x) = e^{-0.5\cdot\left([0 \quad 1]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right)^2}$



(c) $v_m = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Figura 5.1 - Relacionamento entre função básica e função de ativação. (a) Relacionamento entre entrada transformada e saída. (b) Plano de projeção. (c) Relacionamento entre entradas originais e saída.

5.2.3 Critério de otimização

A transformação da entrada é determinada pelas funções ϕ_m (m = *1*,...,p) e matriz de parâmetros v, ao passo que o modelo de relacionamento da entrada transformada para saída é determinada pelos parâmetros c_{mk} e funções de ativação f_m (m = *1*,...,p; k=1,...,N_s).

Métodos de modelagem freqüentemente usam diferentes funções-objetivo ou critérios de otimização para estimar os parâmetros e as relações funcionais que determinam a transformação das entradas (ϕ_m, v) e aqueles que determinam a relação entre as entradas transformadas e as saídas (c_{mk}, f_m).

Tabela 5.1 - Tipos de métodos de projeção utilizados em modelagem

Método	***Transformação da entrada***	***Função básica***	***Critério de otimização Equação (5.1) ou (5.2)***
OLS	Projeção linear	Forma fixa, linear	*v* - maximizar a correlação quadrada entre entrada projetada e saída *c* - minimizar o erro de predição da saída
PLS	Projeção linear	Forma fixa, linear	*v* - maximizar a correlação entre entrada projetada e saída *c* - minimizar o erro de predição da saída
PCR	Projeção linear	Forma fixa, linear	*v* - maximizar a variância da entrada projetada *c* - minimizar o erro de predição da saída
PPR	Projeção linear	Forma adaptativa, super-suave	[*v, c, f*] – minimizar o erro de predição da saída
PPL	Projeção linear	Forma adaptativa, splines, Hermite	[*v, c, f*] – minimizar o erro de predição da saída
NLPCR	Projeção não-linear, não-local	Forma adaptativa	[v,ϕ]- minimizar o erro de predição da entrada
NLPLS	Projeção não-linear	Forma fixada, radial	[centro,variância] – Minimizar a distância entre as entradas e os centros dos grupos de dados *c* - minimizar o erro de predição da saída

Esta separação de critérios de otimização para modelagem permite controle sobre a redução da dimensionalidade para a entrada transformada e pode resultar em maior precisão

nos modelos empíricos, como demonstrado por BAKSHI & UTOJO (1998). Estes métodos de modelagem podem ser divididos em duas categorias, dependendo do critério de otimização: para a entrada transformada contendo informações somente da entrada, como por exemplo, maximização da variância dos dados capturados pela entrada transformada, usada no método *PCR*; ou contendo informações de entrada e de saída, como por exemplo, maximização da covariância entre a entrada transformada e a saída, usada no método *PLS*. O critério para determinação dos parâmetros da saída e função de ativação é minimizar o erro de predição, sendo comum para todos os métodos de modelagem baseados em dados de entrada-saída.

A natureza da entrada transformada, tipo de função básica, e critérios de otimização discutidos acima proporcionam uma estrutura comum para comparar a grande variedade de técnicas para a entrada transformada e modelagem entrada-saída, como descrita na tabela 5.1. Esta estrutura de comparação é útil para o entendimento das similaridades e diferenças entre os vários métodos e pode ser usada para selecionar o melhor método para uma dada aplicação e identificar os efeitos produzidos pela combinação das propriedades das várias técnicas.

5.3 Técnicas para determinação da função de ativação em modelos de transformação da entrada por projeção linear

Uma vez definida a direção de projeção $v_m \in \Re^{N_0}$, (veja equação (5.1)) as funções de ativação podem ser obtidas usando técnicas suavizantes univariáveis, aproximando os dados de treinamento no espaço transformado de entrada-saída. Uma variedade de técnicas está disponível para determinação da função de ativação, incluindo variáveis de extensão suave (FRIEDMAN, 1984), funções de Hermite (HWANG *et al*., 1994) e splines suavizantes automáticos (ROOSEN & HASTIE, 1994). Qualquer técnica para determinação de uma função de ativação necessita possuir as seguintes características:

- habilidade para suavizar automaticamente o espaço de dados contendo diferentes quantidades e tipos de ruídos e curvaturas;
- rápida implementação com uso reduzido de memória;
- fácil computação dos valores da função de ativação obtida e suas derivadas.

No entanto, problemas de aproximação baseados em pares de vetores de entrada e saída, gerados a partir do mapeamento definido pela função a ser aproximada, são problemas mal-comportados (TIKHONOV & ARSERNIN, 1977), pois a informação contida nos dados de entrada-saída não é suficiente para reconstruir um mapeamento em regiões onde os dados não estão disponíveis e, a princípio, não há como assegurar que a solução ótima dependa continuamente dos dados disponíveis.

Algum tipo de restrição deve ser imposta à função a ser aproximada, de forma a produzir um problema de aproximação bem-comportado. Uma das restrições mais gerais e fracas é assumir que o mapeamento é suave. Esta restrição faz com que a aproximação seja possível, pois impõe aos dados um nível de redundância. Deste modo, a suavidade de uma função reflete o efeito da não-localidade, isto é, o valor de uma função em um ponto depende do valor da função em uma região que envolve este ponto (VON ZUBEN, 1996).

O nível de suavidade do mapeamento a ser aproximado geralmente não é conhecido *a priori*, mas normalmente pode ser obtido a partir dos dados amostrados. Todas as técnicas vão requerer a definição de um grau apropriado de suavidade, ou seja, aquele que proporciona o melhor ajuste. Caso contrário, o problema de aproximação continuará mal-condicionado.

5.3.1 Variáveis de extensão suave

É uma técnica de suavização muito simples, que consiste em ajustar uma reta aos dados, definidos numa janela de extensão ajustável. Se a curvatura e ruído nos dados muda com o tempo é essencial variar a extensão da janela para ajustar o grau de suavidade de uma forma ótima. A super-suavizante (FRIEDMAN, 1984) é semelhante a uma variável não-paramétrica de extensão suave. Os dados $\{(z_1,y_1), (z_2,y_2),... (z_L,y_L)\}$ na janela selecionada contendo L amostras são aproximados utilizando quadrados mínimos (*least square*) para ajuste de uma linha reta dada por:

$$f(z_i) = b + az_i \qquad 1 \le i \le L \tag{5.4}$$

onde z_i são os valores das entradas projetadas linearmente (variáveis latentes). Os parâmetros a e b da equação (5.4) são calculados para diferentes valores do comprimento da janela, e o melhor comprimento para cada ponto-base é selecionado via validação

cruzada. FRIEDMAN (1984) sugeriu alguns valores de $L = 0.05N, 0.2N \; e \; 0.5N$, onde N é o numero total de dados, correspondentes a alta, média e baixa freqüências, respectivamente. A suavidade da curva obtida pode ser melhorada por uma nova suavização associada ao aumento da baixa freqüência (especificada pelo usuário). Detalhes adicionais deste algoritmo são descritos por FRIEDMAN (1984).

A super-suavizante, proposta por FRIEDMAN (1984), adapta-se bem para todos os tipos de suavidade e ruído e pode ser implementada a partir de um algoritmo rápido, devido à simplicidade de cálculo. Infelizmente, a função de ativação f_m é definida somente para os valores dos dados de treinamento. Conseqüentemente, a determinação do valor da função de ativação f_m em outras regiões do espaço e entrada, além dos pontos amostrados que compõem os dados de treinamento, requer interpolação ou extrapolação baseada na suposição de suavidade entre os dados adjacentes de treinamento. A necessidade de armazenamento dos valores da função de ativação f_m para todos os dados de treinamento resulta em aumento da memória utilizada. Além disso, outras propriedades da função de ativação, tais como suas derivadas, devem ser estimadas por diferenciação numérica direta, que pode levar à degradação de desempenho da super-suavizante para dados não apresentados previamente.

5.3.2 Polinômios de Hermite

Para um dado conjunto $\{(z_l, y_l\}_{l=1}^{N}$ a função de suavização pode ser obtida por uma combinação linear de funções ortonormais de Hermite de ordem *n*, na forma:

$$f(z) = \sum_{i=0}^{n} c_i h_i(z) \tag{5.5}$$

onde c_i são os coeficientes de regressão e $h_i(z)$ são funções ortonormais de Hermite. Os polinômios de Hermite $H_i(z)$ podem ser construídos de uma forma recursiva, como segue:

$$\begin{aligned} &H_0(z) = 1, \\ &H_1(z) = 2z, \\ &H_{i+1}(z) = 2(zH_i(z) - iH_{i-1}(z)), \; i > 0 \end{aligned} \tag{5.6}$$

Os polinômios de Hermite são ortogonais no intervalo (–∞, +∞) com respeito à função ψ^2, isto é,

$$\int_{-\infty}^{\infty} \Psi^2(z) H_i(z) H_j(z) dz = \begin{cases} 0 & \text{se } i \neq j \\ \sqrt{\pi} 2^i i! & \text{se } i = j \end{cases} \tag{5.7}$$

onde $\psi(z)$ é a função gaussiana dada por:

$$\psi(z) = e^{-\frac{z^2}{2}} \tag{5.8}$$

As funções ortonormais de Hermite podem então ser definidas como:

$$h_i(z) = \frac{H_i(z)\psi(z)}{\sqrt{\sqrt{\pi} \cdot 2^i \cdot i!}}, \quad i = 0,1,...,n \tag{5.9}$$

e produzem

$$\int_{-\infty}^{\infty} h_i(z) h_j(z) dz = \begin{cases} 0 & \text{se } i \neq j \\ 1 & \text{se } i = j \end{cases}.$$

Os coeficientes de regressão c_i na equação (5.5) são calculados pela pseudo-inversa, visando minimizar o erro quadrático de aproximação.

Definindo:

$$H = \begin{bmatrix} h_0(z_1) & h_1(z_1) & \cdots & h_n(z_1) \\ h_0(z_2) & \ddots & & \vdots \\ \vdots & & \ddots & \\ h_0(z_N) & \cdots & \cdots & h_n(z_N) \end{bmatrix}, \quad c = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} \quad e \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{5.10}$$

encontrar o valor ótimo de c, equivale a resolver o seguinte problema de otimização :

$$\min_c \left\| y - Hc \right\|^2 = \min_c \frac{1}{2} (y - Hc)^T (y - Hc) \tag{5.11}$$

Tomando $p < N$, a solução ótima c^*, no sentido dos quadrados mínimos, para o problema de otimização acima fica:

$$c^* = (H^T H)^{-1} H y \tag{5.12}$$

As derivadas de h_i (i=0,...,n) e f em relação a z podem ser facilmente calculadas analiticamente, devido às propriedades especiais da função de Hermite:

$$\begin{aligned} \frac{dh_i}{dz}(z) &= \frac{1}{\sqrt{2i}} h_{i-1}(z) - z h_i(z) \\ \frac{df}{dz}(z) &= \sum_{i=0}^{n} c_i \left[\frac{1}{\sqrt{2i}} h_{i-1}(z) - z h_i(z) \right] \end{aligned} \tag{5.13}$$

Usando funções de Hermite como suavizantes univariáveis, conseguimos resolver alguns dos inconvenientes da super-suavizante (HWANG *et al.*, 1994). Desde que funções de Hermite podem ser computadas analiticamente e recursivamente, nenhuma armazenagem da função suavizada nem intervalo conveniente de interpolação são requeridos para predição dos dados de teste. Outras informações, tais como derivadas, podem ser facilmente calculadas analiticamente (equações 5.13), ao invés de numericamente como nas super-suavizantes.

Um dos maiores problemas na utilização de polinômios de Hermite é que a ordem n, tem que ser escolhida *a priori* e é, algumas vezes, crítica para o sucesso da aproximação. Uma seleção inadequada pode levar a resultados insatisfatórios no treinamento e teste. Este é usualmente explicado como a manifestação do dilema bias-variância (GEMAN *et al.*, 1992). Usando um n muito grande é supostamente possível diminuir assintoticamente o erro de aproximação, mas correndo-se o risco de aumentar o erro de generalização. Uma forma de estimar o número n de funções de Hermite a serem utilizadas na suavização é através de validação cruzada.

5.3.3 Splines suavizantes

Esta técnica pode ser formalizada como um problema de otimização, cujo objetivo é minimizar uma combinação entre o critério dos quadrados mínimos e um termo que assume valores inversamente proporcionais à suavidade da função f, na forma:

$$\sum_{i=1}^{N} \{y_i - f(z_i)\}^2 + \lambda \int \{f''(t)\}^2 dt \tag{5.14}$$

O segundo termo na equação (5.14) é ponderado pelo parâmetro de suavidade λ e mede a ausência de suavidade da função f no espaço de aproximação, a partir da integral da derivada segunda de f. O valor de λ controla a suavidade da função, sendo que valores maiores de λ resultam em ajustes mais suaves. O valor do parâmetro suavizante pode ser escolhido automaticamente por validação cruzada generalizada (ROOSEN & HASTIE, 1994). *Splines* suavizantes automáticos não requerem armazenagem da função suavizada para computar valores para os dados de teste e as derivadas da função básica podem ser calculadas através das funções de *splines*.

Os *splines* suavizantes podem ser considerados generalizações de filtros passa-baixa, tendo os dados de aproximação $\{(z_l, y_l)\}_{l=1}^{N}$ como entrada, visto que a função de regularização penaliza componentes de alta freqüência em f (WAHBA, 1975).

5.3.4 Redes neurais artificiais

Uma *RNA* com neurônios na camada intermediária que apresentam função de ativação fixa pode também ser empregada como função de suavização. A suavidade do ajuste é basicamente determinada pelo número de neurônios na camada intermediária, que pode ser selecionado por validação cruzada, usando dados de teste. Predição dos dados de teste requer armazenamento de todos os parâmetros da *RNA*, que pode ser menos compacta que os polinômios de Hermite ou *splines* suavizantes para algumas formas de funções de ativação. *RNA*s podem também não proporcionar boas aproximações dos dados no caso de convergência para mínimos locais.

5.4 Critérios de busca para direções de projeção

Métodos baseados em projeção linear diferem no critério de busca usado para determinação das direções de projeção, conforme mostrado na Tabela 5.1. Um critério de busca para modelagem de múltiplas entradas e uma única saída, para métodos lineares como *OLS, PLS* e *PCR*, foi proposto por STONE & BROOKS (1990) como:

$$\max_{v_m}\left\{[corr^2(\mathbf{y}, \mathbf{x}\cdot v_m)][\mathrm{var}(\mathbf{x}\cdot v_m]^{\gamma}\right\} \tag{5.15}$$

onde, **y**, **x**, v_m são respectivamente o vetor de saídas, o vetor de entradas e a direção de projeção. A função objetiva dada pela equação (5.15) se aplica a *OLS, PLS* e *PCR* para γ igual a 0, 1 e ∞, respectivamente. Outros valores de γ entre 0 e ∞ resultam em métodos que estão no intervalo entre *OLS* e *PCR*. O valor ótimo de γ e o número de funções básicas determinam a generalidade do modelo e podem ser estimados via validação cruzada. O modelo *CR* é pelo menos tão preciso e compacto quanto aqueles obtidos por *OLS*, *PLS* ou *PCR* (STONE & BROOKS, 1990; WISE & RICKER, 1993).

A não-linearidade da função de ativação não altera o critério de otimização utilizado pelos métodos *PCR* e *NLPCR* na determinação da direção de projeção, visto que ambos os métodos se concentram na transformação do espaço de entrada via maximização da variância capturada pela entrada projetada, na forma:

$$\max_{v_m} \{\text{var}(\mathbf{x} \cdot v_m)\} \tag{5.16}$$

Já as técnicas de *OLS, PPR, PPL* e *RNA* se concentram inteiramente na minimização do erro de predição, visto que este critério de otimização é equivalente a maximização da correlação quadrática entre a saída atual e a saída aproximada (BAKSHI & UTOJO, 1998).

A equação (5.15), com $\gamma = 1$, tem sido usada como critério de otimização para modelagem *NLPLS* por WOLD *et al*. (1989) para *PLS* com polinômios quadráticos, WOLD (1992) para *PLS* com *splines*, e HOLCOMB & MORARI (1992) para redes neurais/*PLS*. A técnica *NLPLS*, desenvolvida por FRANK (1990) e QUIN & MCAVOY (1992), usa o critério de otimização *PLS* linear devido à sua fácil computação para determinação da direção de projeção ótima, e é baseado na suposição de que a função de ativação não-linear não tem um efeito significativo sobre a direção de projeção para a modelagem *NLPLS*.

BAKSHI & UTOJO (1998), propuseram um critério de otimização que englobasse todos os métodos baseados em projeção, na forma:

$$\max_{d_m} \left\{ [corr^2(y, f_m(\mathbf{x} \cdot v_m))]^{1+\gamma-2\gamma^2} [\text{var}(\mathbf{x} \cdot v_m)]^{-3\gamma-2\gamma^2} \right\} \tag{5.17}$$

onde o valor de γ igual a 0,1 e ∞ resulta em {*RNA*, *PPR* ou *OLS*}, {*NLPLS* ou *PLS*} e {*NLPCR* ou *PCR*}, respectivamente

Embora outras técnicas de determinação da direção de projeção para os diversos métodos de projeção linear tenham sido apresentadas na literatura, este estudo vai se restringir às técnicas desenvolvidas para o *PPL*.

5.4.1 Definição das direções de projeções para PPL

O método original de aproximação por busca de projeção proposto por FRIEDMAN & TUKEY (1974) era muito rudimentar. A direção de projeção era ajustada manualmente e de forma contínua. Por inspeção visual, o usuário determinava qual a direção de projeção que fornecia o gráfico mais interessante. DIAGONIS (1985) apresentou estimativas do número de projeções em função da dimensão do espaço de aproximação, demonstrando a impraticabilidade de se explorar exaustivamente, na forma descrita acima, as projeções em espaços de dimensão maior ou igual a três. Justifica-se, portanto, a implementação de um procedimento automatizado de busca da direção de projeção, capaz de emular a estratégia utilizada pelo usuário e substituí-lo, principalmente no caso de problemas de aproximação de dimensão elevada.

No entanto, não existe um consenso do que venha a ser uma projeção interessante para permitir sua aplicação a procedimentos automáticos de busca de projeção. Em vista disso, é muitas vezes preferível definir formalmente uma direção não-interessante e otimizar no sentido contrário a este conceito.

Em vista da dificuldade de modelagem de um único índice de projeção que satisfaça todas as expectativas, recorre-se a diversos índices de projeção, cada qual fornecendo uma solução específica. VON ZUBEN (1996) apresenta uma estudo detalhado de diversos índices de projeção.

5.5 Metodologia de treinamento

As metodologias de treinamento para construção de modelos de aproximação pertencem a duas principais categorias, dependendo de como os parâmetros do modelo são estimados. A abordagem de *modelagem monolítica* determina todos os parâmetros do modelo ao mesmo tempo. Exemplos desta abordagem incluem decomposição em auto-valores para computação das direções de projeção no *PCR* e *PLS*, e o algoritmo de

retropropagação do erro para *RNA* (RUMELHART & MCCLELAND, 1986). A abordagem de *modelagem modularizada* determina os parâmetros do modelo para uma função básica (equivalente a um módulo) por vez, aproximando o resíduo resultante da introdução anterior de outras funções básicas. Exemplos desta abordagem incluem o algoritmo de quadrados mínimos parcial iterativo não-linear (*NIPLAS*), proposto por MARTENS & NAES (1989) para *PCR* e *PLS*, *cascade correlation* para *RNA,* proposto por FAHLMAN & LEBIERE (1990), o algoritmo *PPR* proposto por FRIEDMAN & STUETZLE (1984) e o algoritmo *PPL* proposto por HWANG *et al*. (1994).

Métodos de modelagem modularizada são usualmente mais flexíveis que os métodos de modelagem monolítica, visto que o modelo existente pode ser facilmente adaptado através da adição de novos módulos para capturar o erro residual de aproximação, quando necessário.

5.6 Métodos construtivos × métodos de poda

A técnica de poda na rede neural pode ser implementada ou usando estratégia *off-line* ou *on-line*. A estratégia de poda *off-line* poda os pesos e neurônios redundantes, ou com papel insignificante, depois que uma rede neural de grande dimensão é treinada, respeitando-se critérios de generalização. Exemplos típicos são *o OBD (Optimal Brain Damage)* (LECUN *et al.*; 1990) e o método de redução da aproximação de FROBENIUS (HUNG & HU, 1991). Em contraste, a estratégia de poda *on-line* poda os pesos e neurônios durante o curso de treinamento, incluindo à função de custo um termo de regularização que força os pesos de pequena magnitude a convergirem para zero (HANSON & PRATT, 1989; WEIGEND *et al.*; 1991).

A abordagem de poda tem algumas desvantagens:

- na prática, não se conhece o quão grande deve ser a dimensão da rede neural inicial;
- já que a maior parte do tempo de treinamento é gasto com redes de dimensão maior que o necessário, este método é computacionalmente caro;
- muitas redes com diferentes tamanhos podem ser capazes de implementar soluções aceitáveis. Assim, como a abordagem de poda começa com uma rede de dimensão

elevada, que vai tendo sua dimensão reduzida ao longo do treinamento, ela pode não permitir encontrar a solução de menor dimensão, dentre as aceitáveis;

- estes procedimentos de poda usualmente medem a mudança no erro quando neurônios ou pesos da rede são removidos. Entretanto, estas medidas geralmente são aproximadas, devido à natureza da implementação computacional, e assim pode-se introduzir erros cumulativos, especialmente quando muitas unidades são podadas. Tipicamente, a aproximação envolve somente os primeiros ou segundos termos da expansão em série de Taylor para a mudança no erro. Outras aproximações são possíveis, computando estes valores como média ponderada durante o curso de aprendizagem, ou supondo que a hessiana da superfície de erro é diagonal.

Estas são as razões pelas quais algoritmos construtivos são considerados mais promissores que algoritmos de poda. A teoria de regularização (CHAUVIN, 1989; HANSON & PRATT, 1989; WEIGEND *et al.*, 1991) fornece resultados suficientes para solucionar alguns destes problemas, mas é requerido um delicado balanço entre o termo de erro e o termo de penalidade. Esta também aumenta o tempo de treinamento, e o termo de penalidade tende a criar mínimos locais adicionais, no qual o treinamento pode ficar estacionado quando se busca minimizar a função-objetivo (HANSON & PRATT, 1989).

5.7 Tipos de métodos construtivos no contexto de redes neurais artificiais

Os métodos construtivos a serem considerados neste estudo e associados ao treinamento de redes neurais artificiais são: aprendizado por busca de projeção (*PPL: projection pursuit learning*), aprendizado por correlação em cascata (*CCL: cascade-correlation learning*). Vamos fazer uma breve discussão sobre cada um deles.

5.7.1 Aprendizado por busca de projeção (PPL)

A rede neural com aprendizado por busca de projeção (*PPL*) é uma rede neural generalizada, a qual será melhor investigada no capítulo 6, onde receberá a denominação de dispositivo neurocomputacional híbrido. Sua origem está associada a procedimentos estatísticos propostos para análise de dados empregando técnicas de modelagem não-paramétrica (veja figura 5.2). O nome deste método de aprendizado deriva do fato de que é necessário interpretar dados pertencentes a espaços de dimensão elevadas, buscando direções de projeção "interessantes". Uma versão mais simplificada do *PPL*, denominada regressão por busca de projeção (*PPR*), foi originalmente proposta por FRIEDMAN & STUETZE (1981). Mais tarde, FRIEDMAN (1984) apresentou uma versão generalizada, conveniente para múltiplas respostas de regressão e classificação. O *PPL* (HWANG *et al.*, 1994) é matematicamente modelado como um aprendizado de uma rede estática com uma única camada intermediária, e seu mapeamento de entrada-saída coincide com aquele empregado pelos métodos baseados em projeção linear, já apresentados na equação (5.1) e reproduzidos aqui por conveniência:

$$\hat{y}_k(x) = \sum_{m=1}^{p} c_{mk} f_m \left(\sum_{j=1}^{N_0} v_{jm} x_j \right) \tag{5.18}$$

onde $\{c_{mk},\ m=1,...,p;\ k=1,...,N_s\}$ são os pesos da camada de saída conectando o m-ésimo neurônio da camada intermediária a todas as unidades de saída, f_m é uma função do k-ésimo neurônio da camada intermediária, e $\{v_{jm},\ j=1,...,N_0;\ m=1,...,p\}$ denota os pesos da camada intermediária, conectando todas as unidades de entrada ao m-ésimo neurônio. Para justificar os pesos da saída, HWANG *et al.* (1994) sugere que, como normalmente há várias saídas, é vantajoso normalizar os sinais de ativação da camada intermediária para serem posteriormente ponderados independentemente na composição de cada saída.

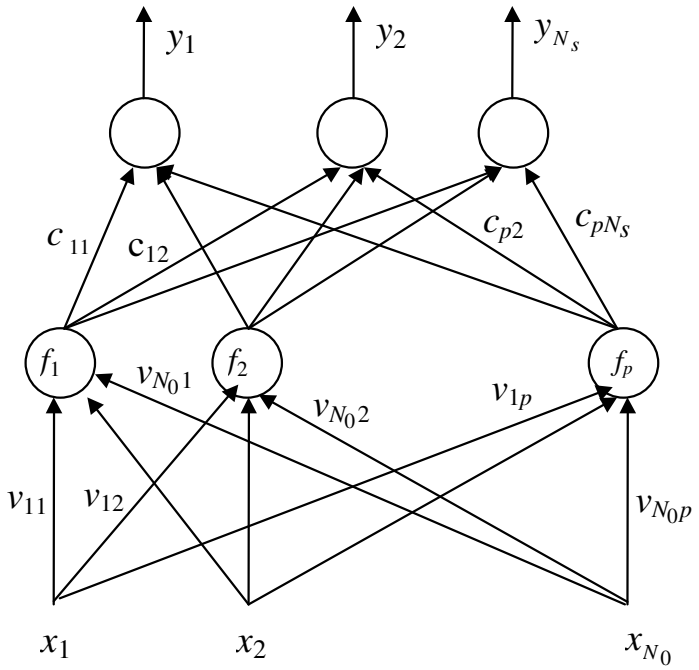


Figura 5.2 - Estrutura de uma rede neural com *PPL*

O *PPL* é responsável pelo ajuste do tamanho da rede, dos pesos e também das funções de ativação (HWANG *et al.,* 1991; HWANG *et al.,* 1992; HWANG *et al.,* 1993, MAECHLER *et al.*, 1990). As redes sujeitas a *PPL* realizam uma aproximação do erro residual por adição de uma unidade candidata com função de ativação ajustável, dentro de uma única camada intermediária, sem conexões em cascata. Redes *PPL* são consideravelmente mais parcimoniosas e precisas do que os *perceptrons* multicamadas, treinados a partir da informação fornecida pela retropropagação do erro em problemas de regressão (HWANG *et al.,* 1994). Para problemas de classificação, redes *PPL* são também recomendadas, por produzirem superfícies de classificação mais suaves nas bordas que a arquitetura *cascade–correlation*, melhorando o desempenho em termos de generalização (HWANG *et al*., 1991). Existe uma garantia teórica de convergência do processo de minimização do erro residual pelo acréscimo seqüencial de novos neurônios à rede (JONES, 1992)

5.7.2 Aprendizado por correlação em cascata (CCL)

O aprendizado por correlação em cascata (*CCL*) também é uma técnica de aprendizado dinâmico, no sentido de conduzir a modificações na arquitetura da rede neural ao longo do treinamento (FAHLMAN & LEBIERE, 1990). Similar ao caso do *PPL*, uma rede neural submetida ao aprendizado por correlação em cascata (*CCL*) cria novas unidades candidatas (dentre um conjunto de possíveis unidades candidatas), uma por vez, e treina os pesos desta unidade candidata, congelando os pesos associados às unidades já existentes. Ao contrário do *PPL*, em que cada unidade candidata recebe somente as entradas, a unidade candidata não apenas recebe as entradas, mas também a saída de todas as unidades já instaladas na rede, de modo a habilitar a capacidade de detecção de características que podem ser bem representadas por estruturas em cascata (figura 5.3).

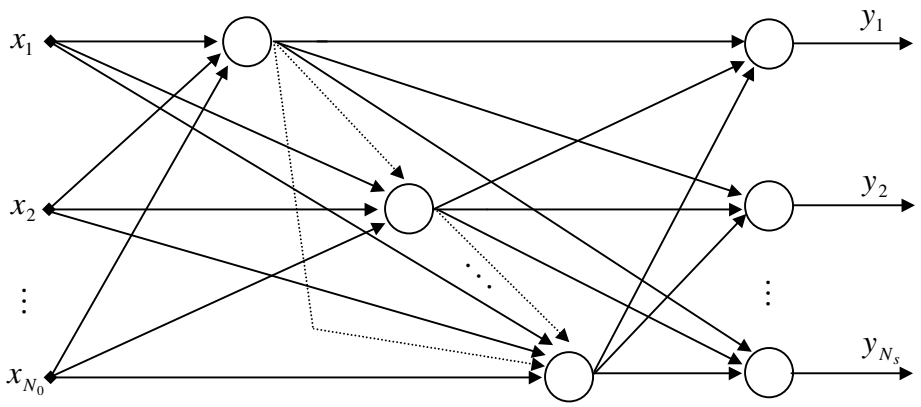


Figura 5.3 - Estrutura de uma rede neural com *CCL* (setas tracejadas referem-se para um conexão entre as unidades)

5.8 Diferenças entre PPR e PPL

Uma das principais diferenças entre *PPR* (*projection pursuit regression*) e *PPL* (*projection pursuit learning*) está na forma do ajuste de parâmetros. No *PPR*, os parâmetros são ajustados com base em métodos iterativos e em técnicas de otimização elementares. No *PPL*, todas as técnicas de otimização mais avançadas, assim como todas as etapas lineares do processamento, são devidamente exploradas.

Entretanto, como notado em (HWANG *et al.,* 1994), o *PPR* com funções de ativação super-suavizantes usa grandes tabelas de regressão, apresenta aproximação grosseira no cálculo das derivadas e requer a interpolação de grandes quantidades de intervalos na computação dos valores de ativação. ROSSEN & HASTIE (1993) e ROSSEN & HASTIE (1994) usaram *splines* suavizantes como funções de ativação dos neurônios, o que permite o cálculo exato de derivadas e interpolação suave, mas requer a adoção de validação cruzada generalizada para selecionar o grau de suavidade das funções (HASTIE & TIBSHIRANI, 1990).

Enquanto as propriedades de convergência do *PPR* são conhecidas, para o *PPL* estas não têm sido rigorosamente estudadas. Em KWOK & YEUNG (1996), demonstra-se que redes *PPL*, na forma original proposta por HWANG *et al.* (1994), não apresentam propriedade de aproximação universal para uma ordem finita dos polinômios de Hermite utilizados na definição das funções de ativação, e não podem convergir para a função desejada, mesmo com uma quantidade arbitrariamente grande de unidades na camada intermediária. Isto ocorre em virtude do tipo de função de ativação (composição de funções ortonormais de Hermite). No entanto, incluindo um termo de polarização em cada projeção linear das variáveis do preditor, a rede *PPL* pode ganhar esta capacidade, mesmo para uma ordem finita dos polinômios. Experimentalmente, KWOK & YEUNG (1996) mostram que esta modificação incrementa a taxa de convergência em relação ao número de unidades na camada intermediária, melhora o desempenho de generalização e faz com que o *PPL* seja menos sensível à ordem dos polinômios de Hermite. KWOK & YEUNG (1996) aplicam *PPL* para predição de séries temporais caóticas, e obtêm resultados superiores quando comparados com a arquitetura de correlação em cascata (*CCL*).

5.9 Diferenças entre PPL e CCL

Uma rede neural do tipo *CCL* (FAHLMAN & LEBIERE, 1990; HOEHFELD & FAHLMAN, 1991) treina incrementalmente os pesos associados a cada nova unidade candidata (unidades em cascata a serem adicionadas à rede) mantendo fixos os pesos das unidades já introduzidas. Mais especificamente, somente um único vetor de pesos associado a cada nova unidade introduzida em cascata está sendo ajustado a cada instante,

baseado em um critério de máxima correlação para pesos das unidades em cascata e critério de quadrados mínimos para pesos da camada de saída. Portanto, o treinamento pode ser muito rápido. Um conceito similar de treinamento modular foi também proposto em HRYCERJ (1990). Sem o uso de pesos em cascata, o *PPL* adota um tipo similar de estratégia modular e treinamentos rápidos, onde uma única unidade na camada intermediária está sendo ajustada a cada instante de tempo. Em contraste, no *CCL*, o crescimento do número de pesos é necessário em função da composição em cascata, que aumenta a dimensão da entrada das unidades que vão sendo introduzidas em seqüência, ainda guardando a mesma ativação simples não-linear, e assim torna o treinamento (ajuste de pesos em espaços de maior dimensão) mais e mais custoso, conforme são introduzidas novas unidades. Sendo assim, o *PPL* trata do ajuste individual de cada neurônio sempre em um espaço de dimensão fixa, talvez à custa da necessidade de mais unidades para completar a tarefa de aproximação.

Mas a grande distinção entre *PPL* e *CCL*, e talvez a principal vantagem do *PPL*, está na possibilidade de reajuste dos neurônios já introduzidos, toda vez que um novo neurônio é adicionado à estrutura da rede neural. Esse procedimento é denominado retro-ajuste, e não é possível de ser implementado eficientemente no *CCL,* devido à cascata de conexões.

Por outro lado, redes neurais do tipo *CCL* realizam importante papel como detectoras de características de alta ordem, através da combinação da informação de saída (informação pré-processada) das unidades existentes. As conexões em cascata habilitam a nova unidade recém-introduzida a aproximar o erro residual (o qual não pôde ainda ser eliminado por unidades já introduzidas) empregando também as saídas das unidades já introduzidas, as quais operam como regressores não-lineares das entradas originais.

5.10 Vantagens adicionais dos métodos construtivos

A flexibilidade presente nos métodos construtivos será muito útil na solução de problemas de controle e identificação, por estarem a cargo do ajuste de um número muito maior de fatores, que caso não fossem otimizados automaticamente deveriam ser arbitrados pelo usuário. No método construtivo, além de ser possível ajustar os pesos e definir

automaticamente o tamanho da rede, é possível também ajustar a forma da função de ativação das unidades na camada intermediária. Tradicionalmente, as funções de ativação são fixas e idênticas para todas as unidades. As mais utilizadas são as funções sigmoidais, embora outras funções, tais como *splines*, *hiper-hill* (também chamada barra unidimensional) (HARTMAN & KEELER, (1991)) sejam comumente utilizadas. Então, fazendo as funções de ativação flexíveis, no sentido de que as formas funcionais podem ser modificadas por um conjunto de parâmetros, elas podem se adaptar a cada problema independentemente, de acordo com as necessidades impostas pela aplicação. MOODY & YARVIN (1992), por exemplo, consideraram o uso de polinômios, funções racionais e séries de Fourier como funções de ativação, as quais apresentaram experimentalmente melhor desempenho, na generalização, quando comparadas às funções sigmoidais.

5.11 Algoritmo para Rede Neural Construtiva

Com base no estudo anterior será apresentado um algoritmo para redes construtivas (veja figura 5.2, a qual representa o resultado final de um processo de construção que incorporou p neurônios à estrutura da rede neural)

Passo 1) Faça $n = 1$;

Passo 2) Defina um valor inicial para $\mathbf{v}_m = [v_{m1}, \cdots, v_{mN_0}]$, empregando alguma técnica de busca (veja seção 5.4.1)

Passo 3) Para $\mathbf{v}_m$ fixo, resolva o seguinte problema de otimização

$$\min_{f_m} \frac{1}{N} \sum_{l=1}^{N} (d_l - f_m(\mathbf{v}_m^T \mathbf{x}_l))^2 + \lambda_m \phi(f_m) \tag{5.19}$$

onde d_l é o resíduo do processo de aproximação (ou seja, o erro de aproximação ainda existente mesmo considerando a participação e todos os neurônios já introduzidos e que não tenham índice m) e λ_m é um parâmetro de regularização. A função ϕ mede a suavidade de f_m.

Passo 4) Para f_m fixo, partindo do valor atual de $\mathbf{v}_m$, resolva iterativamente o seguinte problema de otimização (este é um problema de otimização paramétrica que pode ser resolvido empregando o Método de Newton modificado)

$$\min_{\mathbf{v}_m} \frac{1}{N} \sum_{l=1}^{N} (d_l - f_m(\mathbf{v}_m^T \mathbf{x}_l))^2 \tag{5.20}$$

Passo 5) Enquanto não houver convergência, retorne ao Passo 3;

Passo 6) Processo de retro-ajuste: Enquanto não houver convergência, aplique os passos 3 e 4 a todos os neurônios já introduzidos, de forma cíclica e atualizando o resíduo devidamente;

Passo 7) Faça $n = n+1$ e retorne ao passo 2.

5.12 Visão sintética do capítulo

Neste capítulo, foi apresentada uma formalização unificada dos métodos de modelagem baseados em projeção, os quais foram descritos buscando um consenso em termos de argumentação para evidenciar as potencialidades e principais linhas de aplicação. Além disso, foram revistos os métodos construtivos e os métodos de poda para treinamento de redes neurais artificiais, ressaltando as vantagens e desvantagens de cada um. Particularmente, o método construtivo PPL (*projection pursuit learning*) será útil no capítulo 6, quando, baseado em uma abordagem de neuroagentes, apresentamos os dispositivos neurocomputacionais híbridos.

Capítulo 6

Uma visão multi-agentes para dispositivos neurocomputacionais híbridos

6.1 Introdução

Sistemas de engenharia voltados para aplicações práticas, principalmente aqueles encontrados na indústria, geralmente necessitam de ações de controle e são extraordinariamente diversos. A estratégia de controle adotada nem sempre satisfaz todas as necessidades requeridas, devido a algumas dificuldades de projeto. Normalmente, tais dificuldades referem-se à complexidade inerente à planta a ser controlada, a qual é caracterizada por modelos de identificação pouco representativos, ou seja, modelos que consideram algumas restrições simplificadoras, tais como: linearidade, invariância no tempo e inexistência de dinâmicas de ordem elevada. Além disso, o sistema como um todo geralmente é composto por múltiplos subsistemas, que podem operar sobre diferentes escalas de tempo. Em algumas situações, todas as decisões de controle possíveis podem não ser conhecidas com antecedência e o conjunto de decisões disponíveis pode mudar quando o sistema está em operação. O critério pelo qual o desempenho do sistema é avaliado pode também variar com o tempo. Todas estas considerações, junto com a alta dimensionalidade do espaço de decisão e a presença de não-linearidades, incertezas e imprecisão tornam a busca por soluções viáveis de controle uma tarefa extremamente difícil.

Por outro lado, não se tem produzido resultados esperados em tecnologia de informação para aplicação industrial, a qual pode ser definida como uma área de atuação multi-disciplinar, com alto grau de interdeplendência entre os ramos técnico-científicos

envolvidos. Estes fatores tornam o emprego de tecnologia de informação na indústria uma atividade extremamente complexa, pela multiplicidade de objetivos e pelos desafios impostos aos especialistas em solução de problemas industriais. Soluções utilizando inteligência artificial, como tomada de decisão apoiada em sistemas baseados em conhecimento, têm sido exploradas no atendimento de requisitos de projeto, diagnóstico e análise. Entretanto, tais soluções tratam problemas específicos, sem conseguir atender à questão fundamental da indústria, que é a obtenção de modelos genéricos e capazes de extrair aspectos complementares presentes em ferramentas de solução já propostas e disponíveis para uso imediato. O que se busca é a interoperabilidade dos diversos e heterogêneos sistemas de informação tecnológica, predominantes na indústria.

As dificuldades que surgem em controle de sistemas complexos, descritos acima no contexto de aplicações industriais, podem ser classificadas em três categorias: complexidade computacional devido à quantidade de informação, não-linearidades e incertezas. De acordo com NARENDRA & MUKHOPADHYAY (1996), sistemas de controle capazes de dominar estas três categorias de dificuldades são chamados de *sistemas de controle inteligente*. Em termos qualitativos, quanto maior a habilidade no tratamento das dificuldades acima, mais inteligente será o sistema de controle.

Recentemente, há considerável discussão, tanto no meio acadêmico quanto no meio comercial e industrial, sobre o que vem a ser uma máquina inteligente ou um sistema inteligente. Para os técnicos, tal conceito está vinculado à presença de sofisticados sensores e atuadores, poderosos computadores ou microprocessadores e grandes projetos de software. Para os não-técnicos, tais máquinas são dispositivos robóticos indispensáveis em indústrias com alto nível de automatização, dotados de eficientes métodos de extração, processamento, organização, armazenamento e uso de informação com o propósito de elevar a produtividade e melhorar a qualidade dos produtos. Ambos os pontos de vista são coerentes, embora imprecisos, e é seguro dizer que a capacidade e a influência de tais máquinas e sistemas continuará crescendo drasticamente nas próximas décadas.

O termo *sistema inteligente* tem sido usado desde a origem da inteligência artificial, por volta de 1950, quando pesquisadores tentaram implementar programas computacionais que exibissem aspectos da inteligência humana. Tais programas provavam teoremas e/ou jogavam xadrez, pertencendo a uma classe de raciocínio puramente de máquina (raciocínio

mecânico), sendo considerados *sistemas inteligentes estáticos*. Em contraste com estes sistemas, desde o início dos anos 70 alguns esforços têm sido realizados visando projetar sistemas autônomos que extraiam suas informações diretamente do mundo real e tomem ações de controle apropriadas. Tais sistemas são de natureza dinâmica e adaptativa, sendo mais apropriadamente denominados de *sistemas de controle inteligente*.

Atualmente, a atenção tem sido voltada mais para conceitos de sistemas modulares e modelos hierárquicos. A idéia é construir uma abordagem que apresente maior flexibilidade de processamento através de combinações refinadas de diversos módulos especializados, denominados agentes.

No que diz respeito ao interesse desta dissertação, embora a capacidade de aproximação universal (HORNIK *et al.*, 1989; HECHT-NIELSEN, 1989; KURKOVÁ, 1992) indique que uma rede neural é capaz de representar qualquer mapeamento não-linear contínuo via aprendizado e, portanto, capaz de tratar eficientemente as três categorias de dificuldades encontradas em uma ampla classe de sistemas complexos, este é um resultado de natureza existencial, não fornecendo qualquer direcionamento para a definição do grau de flexibilidade que deve ser adotado pela rede neural para atender à complexidade inerente do problema a ser resolvido (MÜHLENBEIN, 1990).

Procedimentos de tentativa e erro, embora bem-sucedidos na definição de um grau de flexibilidade adequado junto a alguns problemas de importância prática, são comprovadamente insatisfatórios no contexto amplo de aplicação considerado aqui, por representarem técnicas de busca exaustiva em espaços de elevada dimensão. Sendo assim, este capítulo se ocupa da descrição e análise de sistemas modulares construtivos baseados em técnicas de aprendizado construtivo para redes neurais artificiais, utilizando conceitos provenientes da teoria de agentes.

Sistemas modulares não representam uma idéia nova. Sua utilidade tem sido reconhecida em muitas áreas de atuação científica, uma vez que são vitais para a simulação e compreensão de sistemas complexos. Visando uma sintonia do grau de flexibilidade dos modelos resultantes, a aplicação de redes neurais e outras metodologias de modelagem requerem um perspectiva modular. Numerosas sugestões têm sido propostas na literatura considerando sistemas modulares na forma de redes neurais artificiais, sendo que exemplos são encontrados em REILLY *et al.* (1987) e ŚMIEJA (1991). Em ŚMIEJA & MÜHLENBEIN

(1992), métodos de modularização são brevemente comparados. Dois tipos básicos de sistemas modulares não-hierárquicos podem ser identificados: aqueles que decompõem o espaço de entrada ao longo dos módulos e aqueles que não decompõem o espaço de entrada ao longo dos módulos. O último tipo é também referido como tendo um grupo de arquiteturas diferentes (BEYER & ŚMIEJA, 1993). Um problema fundamental a ser enfrentado por tais sistemas é conhecido como problema de decomposição e recombinação (ŚMIEJA & MÜHLENBEIN, 1992; JOE *et al.*, 1990): como dividir automaticamente o espaço de entrada e depois recombiná-lo após o processamento através dos agentes. Uma solução para tais questões é fazer com que a informação presente nos dados conduza o processamento para que este seja dependente do problema (BEYER & ŚMIEJA, 1993; GEMAN *et al.*, 1992). Como resultado, em ŚMIEJA (1996) são desenvolvidos três métodos diferentes de decomposição dinâmica para a arquitetura que recebe a denominação de *Pandemonium*.

Por outro lado, sistemas de computadores baseados em teoria de agentes são amplamente conhecidos na comunidade de Inteligência Artificial. Eles têm a função de reduzir o problema de sobrecarga de informação, e mais importante, facilitar a interoperabilidade de sistemas heterogêneos distribuídos (HERMANS, 1997).

Neste contexto, agentes são entidades de software com autonomia suficiente e inteligência para habilitá-los a realizar tarefas específicas com pouca ou nenhuma supervisão humana. Um agente interage com um ambiente dinâmico visando atender a algum objetivo. A abordagem de agentes proporciona uma metáfora útil para projeto e desenvolvimento de sistemas modulares. Ela enfatiza certos atributos tais como: autonomia, habilidade social, responsabilidade, pró-atividade e alto nível de comunicação que estão relacionados ao contexto de projeto colaborativo. Estes atributos conduzem a novos conceitos (como por exemplo, negociação cooperativa e/ou competitiva), que, por sua vez, dão origem a novas ferramentas e técnicas que podem enriquecer a estratégia de projeto multi-agentes tradicional. NDUMU & TAH (1999) argumentam, em essência, que a abordagem baseada em agentes proporciona uma estrutura integrada, por canalizar resultados de diversas disciplinas de Inteligência Artificial em projeto de sistemas práticos.

Desta forma, agentes inteligentes são atualmente objetos de pesquisa em muitos campos, tais como engenharia, psicologia, sociologia e inteligência artificial. Esta é a razão

pela qual uma grande quantidade de abordagens, inclusive as apresentadas nesta dissertação, e projetos usam o termo agentes, desde interfaces de usuários a complexos sistemas distribuídos. Sendo assim, é necessário especificar apropriadamente o contexto de aplicação a ser desenvolvido neste estudo.

6.2 Objetivos

Este capítulo tem como objetivo apresentar alguns dos vários conceitos sobre agentes e uma classificação de agentes tendo em vista as diversas áreas de abrangência e a diversificação de abordagens. Em seguida, busca-se apresentar as vantagens da utilização dos agentes na teoria conexionista e o modelo do neuro-agente proposto, unidade básica do dispositivo neurocomputacional híbrido. Além disso, será realizado um esforço no sentido de identificar os principais aspectos que possam contribuir para controle e identificação de sistemas dinâmicos.

6.3 Originalidade da abordagem

VON ZUBEN (1996) apresentou métodos de análise e síntese de modelos paramétricos e não-paramétricos de redes neurais artificiais, utilizando resultados derivados da teoria de aproximação de funções e análise numérica. A flexibilidade destas estruturas conexionistas não-lineares foi explorada com base em técnicas de regularização e métodos de otimização irrestrita. A rede neural não-paramétrica resultante realiza mapeamentos não-lineares estáticos via métodos construtivos caracterizados por redução de dimensionalidade e propriedades de aproximação bem definidas. Neste capítulo, uma nova interpretação das redes neurais construtivas, usando alguns conceitos de teoria de agentes, é apresentada com o intuito de melhorar a eficiência e eficácia da formalização e aplicação de redes neurais submetidas a métodos construtivos de treinamento.

6.4 Métodos construtivos e teoria de agentes

A teoria de agentes se ocupa do estudo de métodos, modelos e ferramentas que suportem o tratamento conjunto de controle e informação na implementação de sistemas autônomos dotados de comportamento inteligente. As iniciativas pioneiras em tratar controle e informação nos organismos vivos e nas máquinas foram introduzidas formalmente no final dos anos 40 (WIENER, 1948). No entanto, foi com o desenvolvimento da indústria da informática, observado principalmente a partir dos anos 80, que se pôde dispor de grande poder de memória e processamento a baixo custo, condição necessária para a efetivação da pesquisa envolvendo teoria de agentes.

Além das motivações da seção anterior, o estudo de agentes no contexto deste trabalho é impulsionado principalmente pela sua caraterística de interagir ativamente com o ambiente. Aqui, o ambiente pode ser visto como a própria representação ou conjunto de informações associadas ao problema a ser resolvido computacionalmente.

É possível entender os resultados expressivos obtidos ultimamente no desenvolvimento da teoria de agentes (AGRE & ROSENSCHEIN, 1995) analisando-se os desenvolvimentos paralelos que ocorreram no estudo de sistemas inteligentes, eminentemente na área conhecida como inteligência computacional (ZURADA *et al*., 1994; PALANISWAMI et al., 1995). A inteligência computacional é vinculada, em linhas gerais, à habilidade de aprender a lidar com novas situações através de um ou mais dos seguintes atributos da razão: associação, generalização, descoberta e abstração.

Algumas propriedades encontradas em técnicas empregadas em teoria de agentes, e que as caracterizam como estratégias de inteligência computacional, são:

- capacidade de adquirir conhecimento pela interação com o ambiente;
- capacidade de mudar seu comportamento utilizando técnicas de adaptação e/ou aprendizado;
- capacidade de encontrar soluções que atendam a múltiplos objetivos simultaneamente, utilizando o conhecimento adquirido a partir de sua interação com o ambiente;

- capacidade de operação em condições adversas, tais como: ausência de um conjunto completo de informações para um planejamento prévio de seu comportamento, ruído nos sensores e atuadores;

Uma questão fundamental é decidir que tipo de propriedades do conjunto de agentes devem ser projetadas previamente (propriedades inatas) e quais delas devem ser aprendidas por interação com o ambiente e/ou outros agentes (propriedades não-inatas). Uma vez definido o conjunto de propriedades não-inatas, é necessário definir como implementá-las por processo de aprendizagem ou aquisição de conhecimento.

Para tanto, as técnicas mais avançadas de teoria de informação, identificação e controle de processos e otimização não-linear devem ser empregadas. A teoria conexionista fornece diversos modelos, ferramentas e metodologias para a implementação destas técnicas (VON ZUBEN, 1996). O sucesso das estratégias conexionistas na abordagem de sistemas de agentes está associado à possibilidade de pronta incorporação de outras técnicas de aproximação de funções e inferência estatística (VAPNIK, 1995).

No contexto de teoria de agentes, os estudos se concentram no desenvolvimento de poderosos sistemas de representação e manipulação do conhecimento, enquanto que em teoria conexionista tem-se concentrado no desenvolvimento de poderosos mecanismos de adaptação e aprendizado.

Assim, sob o ponto de vista da teoria de agentes, é possível estudar e implementar os métodos construtivos na geração de dispositivos neurocomputacionais, considerando os neurônios como agentes que interagem entre si e com o ambiente através de mecanismos de competição e cooperação, ganhando mais flexibilidade e autonomia quando comparados com as unidades básicas presente nas redes *MLP*, por exemplo.

Através dos métodos construtivos, emergem soluções de alta qualidade para problemas de complexidade elevada, a partir da interação intensiva dos agentes (neurônios artificiais, com funções de ativação ajustáveis) na tentativa de solução de problemas.

6.5 A origem da teoria de agentes

A idéia de empregar agentes para delegar certas tarefas implementadas em computador foi devidamente formalizada por NEGROPONTE & KAY (1984). O termo agente foi inicialmente definido em trabalhos preliminares na área de inteligência artificial, onde pesquisadores dedicavam-se à tarefa de produzir uma entidade artificial que expressasse algumas habilidades humanas.

As primeiras pesquisas na área foram direcionadas ao uso de agentes como uma forma de se obter uma comunicação de alto nível entre o homem e a máquina, usando uma forma de expressão semelhante à humana. Recentemente, diversas indústrias adotaram a idéia de agentes para ilustrar sua visão de interface do futuro. Apesar do grande esforço que tem sido realizado para modelar e construir modelos de processamento de informação baseados em agentes, as técnicas atualmente disponíveis estão longe de permitir as esperadas interações de alto nível, semelhante àquelas mais elementares que ocorrem entre seres humanos.

Muitos trabalhos têm sido elaborados utilizando agentes. Segundo MORREALE (1998), inicialmente os agentes desenvolvidos estavam restritos a um único computador, ou no máximo a uma rede homogênea, e limitados a tarefas pré-estabelecidas. Atualmente, os agentes estão quebrando este confinamento ganhando mobilidade e aprendendo a executar tarefas com base em sua própria experiência.

6.6 Agentes: Conceitos Principais

Pesquisadores envolvidos no estudo e desenvolvimento da teoria de agentes têm oferecido uma variedade de definições, cada um esperando explicar a razão pela qual usou a palavra "agente". Estas definições tanto podem ser gerais quanto restritas. Suspeita-se que cada uma delas tenha sido originada a partir do conjunto de exemplos de agentes que cada pesquisador tinha em mente.

As dificuldades da definição do termo "agente" devem-se principalmente ao uso indiscriminado deste pela indústria de software, a qual utiliza-o de diferentes formas e com significados variados, como um apelo para a aceitação de seus produtos no mercado.

Com o propósito de ilustrar esta realidade exposta acima, dentre as diversas definições para agente existentes na literatura, serão destacadas as seguintes (GUDWIN, 1999):

"Segundo MINSKY (1994), a palavra agente deve ser usada para referenciar uma máquina que executa algo sem que seja necessário saber como ela trabalha, algo semelhante a uma *caixa-preta*".

"Agentes de software são programas de computador que aplicam técnicas de inteligência artificial para prover assistência ao usuário em tarefas computadorizadas" (MAES, 1994).

"Um agente é um software que sabe como fazer coisas que você provavelmente faria se tivesse tempo". (HERMANS, 1997)

"O termo *agente* é utilizado para representar dois conceitos ortogonais. O primeiro é a habilidade de execução autônoma e o segundo é a habilidade em domínios específicos" (VIRDHAGRISWARAN, 1995).

"Um agente é qualquer coisa que pode ser vista percebendo um ambiente por meio de sensores e atuando no mesmo por meio de atuadores". (RUSSEL & NORVIG, 1995)

"Agentes autônomos são sistemas computacionais que habitam um ambiente complexo e dinâmico, realizam sensoriamento e atuam autonomamente neste ambiente, realizando desta maneira uma série de metas e tarefas para as quais foram projetados" (MAES, 1995).

"Um agente é uma entidade persistente de software dedicada a um propósito específico". (SMITH *et al.*, 1994)

"Agentes inteligentes realizam continuamente três funções : **percepção** das condições dinâmicas de um ambiente, **ação** de modo a afetar condições do ambiente e **raciocínio** para interpretar as percepções, realizar as inferências e determinar as ações". (HAYES-ROTH, 1995)

"Agentes inteligentes são entidades de software que realizam um conjunto de operações em nome de um usuário ou outro programa com certo grau de independência ou autonomia, e desta maneira empregam algum conhecimento ou representação das metas e/ou desejos do usuário". (GILBERT, *et al*, 1995)

"Um agente é um sistema de hardware e/ou software que goza das seguintes propriedades (WOOLRIDGE & JENNINGS, 1995):

- **autonomia:** agentes operam sem a necessidade de intervenção humana ou outra qualquer, e tem um certo controle sobre suas ações e estados internos;
- **habilidade social:** agentes interagem com outros agentes (possivelmente humanos) por meio de uma linguagem de comunicação de agentes (*ACL – Agents Communication Language*);
- **reatividade:** agentes percebem seu ambiente (que pode ser o mundo real, um usuário via uma *GUI*, uma coleção de outros agentes, a internet ou uma mistura de todos estes) e respondem prontamente a mudanças que nele ocorram;
- **pró-ativismo** (*pro-activeness*): agentes não simplesmente reagem em resposta ao ambiente, mas são capazes de exibir um comportamento baseado em metas, tomando a iniciativa."

"Agentes de Software são programas que se empenham em diálogos de forma a negociar e coordenar a transferência de informação" (KAUTZ, 1994).

"Um agente é algo que atende a um conjunto de critérios (LENNY FONER 1993 - MIT Media Lab - condensado):

- autonomia: ação periódica, execução espontânea e iniciativa;
- personalidade: capacidade de aprendizagem e memória;
- habilidade de discurso: diálogo que resulte em um compromisso entre as partes sobre o que deve ser efetuado;
- risco e confiança: habilidade para concretizar a tarefa especificada;
- domínio: razoavelmente bem definido;
- degradação suave: em casos de desencontro na comunicação;
- cooperação: agentes devem colaborar para atingir um objetivo comum;
- antropomorfismo: intenção de parecer humano;
- expectativas: devem ser realistas quanto à capacidade do agente."

"Agentes autônomos são sistemas capazes de uma ação autônoma e propositada no mundo real." (BRUSTOLONI, 1991; FRANKLIN, 1995)

"Um agente autônomo é um sistema que é parte de um ambiente, estando situado dentro dele, e sente e age sobre este ambiente, no tempo, de acordo com seus próprios propósitos, de modo a alterar o que sentirá no futuro" (FRANKLIN & GRAESSER, 1996)

"Agentes são entidades autônomas, perseguidoras de metas, persistentes, racionais, produtivas e comunicativas, que agem em nome de outras, ou seja, não são auto-motivadas" (MURCH & JOHNSON (adaptado), 1999)

"Agente inteligente é um sistema autônomo com a capacidade de auto-organização, auto-representação e auto-interpretação para agir em ambientes dinâmicos e complexos" (SOUZA E SILVA, 1998).

Independente das definições, há um consenso entre os pesquisadores de que um agente deve ser um componente de software que atua em nome do usuário, liberando-o de tarefas repetitivas e cansativas. Um exemplo adequado é o de um agente de viagens, para o qual delegamos a tarefa de reservar passagens, hotéis etc., não nos importando como isto será feito.

Devido à multiplicidade de papéis que os agentes podem assumir, uma definição consensual de agentes se torna muito difícil. Uma maneira de substituir uma definição formal é enumerarmos uma lista das características gerais que devem ser associadas a um agente.

Podemos dividir estas características em duas categorias (HERMANS, 1997): a primeira está associada a uma noção mais conceitual de agentes, que define a abordagem inicial das pesquisas na área, e a segunda associa-se a uma noção mais pragmática. No primeiro caso, podemos enumerar:

- *Autonomia*: Agentes devem operar sem a intervenção humana e devem ter uma espécie de controle sobre suas ações e estado interno;
- *Cooperação:* Agentes interagem com outros agentes e possivelmente seres humanos, através de algum tipo de linguagem específica;
- *Reflexibilidade:* Um agente percebe seu ambiente e responde prontamente a alterações que ocorram neste;
- *Pró-atividade:* Um agente não só responde a mudanças em seu ambiente como também, ao tomar iniciativas, pode exibir um comportamento direcionado a propósitos;
- *Continuidade temporal*: Agentes são processos que estão continuamente em operação podendo estar ativos ou passivos;

- *Orientação a eventos:* Um agente é capaz de suportar tarefas complexas, podendo dividi-las em tarefas menores e definir em qual ordem vai executá-las.

Uma noção que pode ser tomada como parâmetro de um agente foi extraída do campo de pesquisa denominado Inteligência Artificial e considera as seguintes características mais pragmáticas, pertencentes à segunda categoria mencionada acima:

- *Mobilidade:* A habilidade de um agente em se locomover;
- *Benevolência*: Assume-se que os agentes não possuem conflitos de objetivos, fazendo o que for solicitado e buscando cooperação;
- *Racionalidade*: Um agente sempre tentar alcançar seus objetivos, procurando realizar a melhor ação;
- *Aprendizado/Adaptabilidade*: Um agente deve estar apto a aprender e a se ajustar a mudanças no ambiente;
- *Colaboração*: Um agente, para alcançar seus objetivos, deve compartilhar esforços e recorrer a modos de comunicação com outros agentes;
- *Competição*: Um agente deve competir com outros agentes por recursos computacionais disponíveis. A alocação destes recursos será realizada de acordo com o resultado do processo competitivo.

Segundo HERMANS (1997), não existe ainda nenhum agente que possua todas estas características implementadas, embora existam alguns protótipos que apresentem grande parte delas. Também não há ainda um consenso sobre a importância de cada uma destas características nos agentes, mas a maioria dos pesquisadores concorda que são estas as responsáveis pela diferença entre um agente e qualquer outra entidade computacional.

No apêndice A, é apresentada uma classificação detalhada de variados tipos de agentes.

6.7 Aplicações em diversas áreas

As aplicações atuais de agentes são na maioria experimentais. Universidades, como por exemplo *MIT* (*Massachusets Institute of Technology*), e centros de pesquisa, como os da IBM e Microsoft, estão realizando projetos nesta área. Uma grande parte das pesquisas

desenvolvidas enfoca áreas de aplicação básicas, que possam mostrar resultados definidos à curto prazo. Tais aplicações envolvem principalmente:

- Agentes que manipulam *e-mails* (mensagens eletrônicas);
- Agentes que pesquisam através da *Web* diversas bases de dados, procurando por informações de interesse do usuário;
- Agentes que planejam reuniões baseados em lista de participantes ou na agenda eletrônica de cada participante em particular.

Para atender às funcionalidades acima, já existe um grande número de produtos sendo comercialmente vendidos e protótipos em fase final. Dentre eles, podemos citar:

- MAXIMS (METRAL, 1993,1994): um agente que auxilia o usuário com seus e-mails;
- LETIZIA (LIEBERMAN, 1995) e FOOTPRINTS (WEXELBLAT & MAES,1999): agentes de interface que auxiliam a navegação na *Web*;
- MCL (MAES, 1994): auxilia no agendamento de reuniões, negociando horários entre os participantes.

Podemos identificar áreas de aplicação mais complexas, com trabalhos de pesquisa ainda em desenvolvimento, onde o uso de agentes pode trazer grandes benefícios:

Uso Pessoal

- controle de agendas e apontamentos (datas e horas de compromissos), gerenciamento de arquivos e informações, localização de informações, sugestão de locais diferentes para busca de informações.

Gerenciamento de Redes de Computadores

- gerenciamento da arquitetura das redes, tráfego de pacotes (fazendo um re-roteamento quando o volume crescer em demasia), gerenciamento de dados, arquivos, conteúdo de páginas *Web* (atualização automática), localização de dispositivos de hardware em grandes empresas (impressoras, máquinas de fax, copiadoras, etc.), controle de manutenção de equipamentos, etc.

Busca de Informação e Acesso à Internet

- gerenciar, filtrar, priorizar, re-rotear, descartar, monitorar e compartilhar informações de todos os tipos;

- sugerir modos de se obter a informação desejada da maneira mais eficiente possível.

Gerenciamento de Mobilidade;

- assegurar o correto deslocamento de informações entre dispositivos como *notebooks*, computadores corporativos, *pagers, PDA's (Personal Desktop Assistant)*, agendas eletrônicas, com a velocidade e a confidência necessárias ao conteúdo da informação sendo transferida;
- busca de informações vitais em momentos de emergência e situações inesperadas.

Comércio Eletrônico

- seleção de produtos, elaboração de especificações, negociação de preços, cobrança e recebimento imediato de pagamentos;
- para as empresas: encontrar o cliente que deseja seus produtos sem incomodar aqueles que não estão interessados;
- para os clientes: encontrar o negócio que mais satisfaz suas necessidades (relação custo/benefício).

Interface de Usuários com Computadores

- facilitar a comunicação entre usuários e computadores;
- descobrir e registrar as preferências do usuário, interpretar o desejo do usuário, aprender termos e saber aplicá-los no diálogo com o usuário, sugerir ações úteis a usuários inexperientes e experientes e tomar decisões inteligentes que poupem o tempo do usuário.

Desenvolvimento de Aplicações

- localizar programas, códigos, módulos, *designs*, especificações e outras ferramentas técnicas que auxiliem a criação de futuras aplicações (localmente ou na internet, gratuitos ou pagos).

6.8 O grau de inteligência de um agente

6.8.1 Agência e inteligência

Embora a definição de um agente possa implicar a visão deste como uma "caixa-preta" (MINSKY, 1994), onde não há interesse em saber os detalhes de como as tarefas são executadas, há momentos em que se torna necessário saber como e em que grau de autonomia o agente executa suas tarefas. O grau de autonomia e autoridade outorgada a um agente é chamado de agência (*agency*).

A agência pode ser medida qualitativamente pela natureza da interação do agente com outras entidades do sistema no qual ele opera. Assume-se que um agente deve, no mínimo, agir assincronamente e de maneira generalizada: ele deve interagir com outras entidades, tais como softwares aplicativos e bases de dados, e também colaborar e negociar com outros agentes.

Definir o que faz um agente ser inteligente é um conceito polêmico em inteligência artificial. Segundo MINSKY (1994), um agente nunca conseguirá assumir inteligência semelhante aos seres humanos pois são ainda incapazes de adquirir o que ele chama de senso comum (*common-sense*), isto é, noções de espaço e tempo, capacidade de reconhecer propriedades de materiais apenas com o recurso visual, capacidade de atender a objetivos genéricos e planejar a longo prazo, enfim todo conjunto de conhecimentos inerentes aos seres humanos.

Em HERMANS (1997), encontramos a definição de inteligência como sendo o grau de raciocínio e comportamento aprendido, isto é, a habilidade do agente em aceitar declarações dos objetivos do usuário e realizar as tarefas que permitem atender a estes objetivos.

6.8.2 Interação de pessoas com agentes

O conceito de agente, especialmente quando associado à palavra inteligência, evoca a idéia de autonomia, trabalhando sem ou quase sem nenhuma supervisão em tarefas que nos beneficiem. Existem muitos mitos e promessas colidindo com a realidade. No entanto,

o problema principal reside na pergunta: como os agentes inteligentes irão interagir com as pessoas e o que as pessoas pensam sobre os agentes?

Segundo NORMAN (1994) um fator que não pode ser descartado é o "sentimento de controle" inerente ao ser humano. Sob o ponto de vista psicológico, o ser humano valoriza o sentimento de controle que ele tem sobre suas atividades e ações. A utilização de agentes pode mascarar este controle. Por exemplo supondo que uma aeronave possa navegar inteiramente bem com o piloto automático, por mais eficiente que ele seja, as pessoas não vão se sentir à vontade sem uma piloto humano a bordo. Em relação ao agente de software, é bem provável que as pessoas queiram ter maneiras de saber a respeito das ações de seus agentes como uma forma de se sentir seguras.

Por outro lado, há o perigo das pessoas exagerarem na expectativa das capacidades dos agentes. Muitos fabricantes têm uma tendência natural de apresentar seus agentes no formato humano, atribuindo-lhes nomes próprios ou mesmo uma face humana ao modelo. Esta personificação sugere promessas de desempenho que não poderão ser atingidas.

Algumas questões sobre a interação homem-agente podem ser levantadas, tais como:

- Como as pessoas devem instruir e controlar um agente;
- Como um agente solicita informações às pessoas;
- Como um agente oferece informações às pessoas.

Todas estas questões devem ser levadas em consideração no projeto de sistemas inteligentes.

6.8.3 Projeto de agentes inteligentes

Como discutido nas subseções anteriores, existem diversas considerações sobre quais tarefas um agente inteligente deve executar, assim como qual deve ser o grau de interação deste com o usuário. Outro aspecto que vem sendo discutido é a forma como o agente opera e como ele adquire conhecimento, isto é, a sua competência.

Segundo MAES (1994) dois problemas principais devem ser resolvidos para a construção de agentes inteligentes, que são os seguintes:

1. O primeiro problema diz respeito à competência, isto é, como um agente adquire o conhecimento que ele precisa para decidir quando e como ajudar um usuário.
2. O segundo problema diz respeito à confiança: como podemos garantir que os usuários se sintam seguros em utilizar um agente.

6.9 Técnicas de Aprendizado

Um dos problemas na construção de agentes inteligentes é garantir a obtenção de informações precisas sobre o interesse do usuário, necessidades e preferências. Uma questão fundamental é saber quais as técnicas a serem utilizadas para um agente adquirir conhecimentos.

Existem algumas técnicas de aprendizado tradicionais utilizadas no campo da Inteligência Artificial, tais como (GREEN *et al*., 1997):

Técnica de classificação simbólica: busca classificar um conjunto de exemplos dentro de um número finito de classes abstratas. Por exemplo, uma árvore de decisão pode ser montada a partir de um conjunto de exemplos, onde cada elemento consiste em um número de atributos relevantes de uma classe.

Técnica de classificação sub-simbólica: Esta técnica é baseada em sistemas de redes neurais. Estes sistemas recebem padrões como entrada e classificam-nos em um número finito de categorias.

As técnicas citadas acima, apesar de tradicionais, apresentam o problema de não oferecerem o aprendizado em tempo-real e sim *off-line*. O aprendizado em tempo real é que define um agente autônomo, capaz de agir sem intervenção do usuário. Para este tipo de agente, a técnica utilizada é a do *aprendizado a partir dos estímulos de ambiente.*

Segundo MAES (1994), na técnica de aprendizado em tempo real, um agente pode adquirir conhecimentos a partir de quatro fontes diferentes:

- Observando e imitando o usuário*;*
- Baseado na realimentação por parte do usuário: o usuário pode recusar uma sugestão do agente de forma implícita ou explícita;

- Pode ser treinado pelo usuário através de exemplos (condicionamentos): podem ser fornecidos exemplos hipotéticos de eventos e situações;
- Pode solicitar ajuda interagindo com outros agentes que assistem a outros usuários.

6.10 Computação emergente e sistemas complexos

O surgimento de processamento de informação global em sistemas computacionais caracterizados pela interação local de seus agentes constituintes é denominado computação emergente. Sistemas com comportamento emergente apresentam auto-organização, fenômenos coletivos e comportamento cooperativo, recebendo a denominação de sistemas complexos.

No contexto desta dissertação, é importante descrever as principais propriedades apresentadas pelos sistemas multi-agentes, particularmente quando seus agentes constituintes apresentam a capacidade de adaptação ou aprendizado:

- uma coleção de componentes primitivos, denominados agentes;
- interação não-linear entre agentes e entre agentes e seu ambiente;
- propriedades globais não-antecipáveis, geralmente resultantes desta interação;
- adaptação do comportamento dos agentes em função de restrições impostas por outros agentes e seu ambiente;
- evolução do comportamento do sistema com o tempo.

Um agente é considerado adaptativo quando ele é capaz de realizar as seguintes funções em uma seqüência coerente:

- observar o ambiente;
- com base nesta observação, decidir o que fazer;
- agir de acordo com suas próprias decisões.

A teoria de agentes representa um novo paradigma dentro da área de inteligência artificial, pela constatação de que regras para conferir inteligência a um programa computacional são muito complexas para serem codificadas da maneira usualmente empregada pelos sistemas especialistas, desenvolvidos nos anos 70. Com este novo paradigma, o que se busca é a emergência de soluções de alta qualidade para problemas

com complexidade elevada, a partir da interação intensiva de regras simples e habilidade de adaptar estas soluções frente a novas situações.

6.11 Princípios de neurocomputação

Os dispositivos neurocomputacionais são compostos basicamente por processadores de informação não-lineares simples, dotados de grande poder de adaptação e que interagem localmente para produzirem elementos dentre uma ampla classe de modelos paramétricos e não-paramétricos multidimensionais.

O processo de análise em neurocomputação envolve primordialmente aspectos científicos derivados da neurologia, psicologia e ciência cognitiva (PYLYSHYN, 1984). Portanto, as fontes de inspiração para a neurocomputação continuam sendo as propriedades e o comportamento do cérebro natural encontrado nos organismos vivos, particularmente nos mamíferos. No entanto, em lugar de buscar a reconstrução parcial ou total deste sistema natural tão fielmente quanto possível, a neurocomputação procura adotar modelos teóricos utilizando o menor número de ingredientes associados ao sistema natural que ainda assim permitam preservar uma série de propriedades essenciais, como capacidade de aprendizado e generalização, habilidade de manipulação de incertezas e potencial de representação não-linear.

Por outro lado, o processo de síntese em neurocomputação se restringe basicamente a aspectos de engenharia, de forma a permitir a geração de modelos conexionistas com potencial de representação e poder de processamento adequados para aplicação junto a problemas de elevada complexidade. Estes modelos são formados pela interação de múltiplos agentes neurocomputacionais, também denominados neurônios artificiais generalizados, dotados de capacidade de adaptação e submetidos a efeitos de competição e cooperação (VON ZUBEN, 1996).

Os modelos mais simples de um dispositivo neurocomputacional assumem neurônios artificiais com funções de ativação idênticas, quase sempre do tipo sigmoidal, representando um efeito de saturação (para maiores detalhes, veja capítulo 2). No entanto, para aumentar o poder de representação sem requerer um aumento proporcional na

quantidade de neurônios artificiais e conexões, é fundamental permitir que diferentes neurônios artificiais possam apresentar funções de ativação completamente distintas e específicas para cada aplicação (como na abordagem neuro-agente a ser descrita mais adiante), resultando em redes neurais artificiais híbridas e projetadas no sentido de utilizar otimamente os recursos computacionais alocados.

A denominação de dispositivos neurocomputacionais, em lugar de redes neurais artificiais, se justifica pelo fato da estrutura de processamento resultante, assim como sua forma de obtenção baseada na teoria de agentes, ser diferente daquela empregada tradicionalmente na área de redes neurais artificiais. No entanto, o potencial de representação universal e a grande flexibilidade de aplicação a problemas caracterizados por relações não-lineares continuam presentes nos dispositivos neurocomputacionais apresentados na próxima seção (redes neurais construtivas baseadas na teoria de agentes: neuro-agente).

HAYKIN (1999) e HECHT-NIELSEN (1991) apresentam discussões introdutórias mais detalhadas a respeito de dispositivos neurocomputacionais clássicos, inclusive evidenciando o aspecto histórico da pesquisa e as motivações biológicas associadas.

6.12 Uma abordagem neural multi-agentes

6.12.1 A arquitetura Pandemonium

Como uma proposta de solução para problemas complexos, baseada em teoria de agentes, SELFRIDGE (1958) propôs originalmente uma arquitetura chamada *Pandemonium*. A idéia é dividir-para-conquistar um domínio de um problema complexo, através do uso de um número de agentes especializados, trabalhando em paralelo. Aqui, todos os agentes, ou demônios, processam o mesmo sinal em paralelo e cada um proporciona uma possível resposta. O agente que produzir melhor resposta é mais credível, uma vez que a questão apresentada reside na sua região de especialidade. Por exemplo, para o problema de classificação das 26 letras do alfabeto são necessários 26 agentes, sendo que cada um deles é especializado no reconhecimento de uma letra em particular. Cada agente dispõe de técnicas de modelagens pouco diferenciadas e assim reconhecem aspectos característicos

do padrão associado à sua área de especialidade. O procedimento de aprendizado utilizado é o gradiente descendente para adaptar os pesos que determinam o filtro usado pelos agentes. Existe um "agente de decisão", no topo da hierarquia, que tem a função de escolher o agente que produziu maior saída dentre todos os demônios. Após o processo de aprendizado, temos o cenário descrito na figura 6.1, na qual o primeiro demônio (treinado para reconhecer particularidades referentes à letra A) produz uma saída mais alta que os outros demônios para a mesma entrada (uma letra qualquer, com ruído ou não) significando que a entrada contém grande similaridade com seu padrão armazenado. Os outros agentes podem perceber aspectos de suas especialidades particulares para a entrada apresentada, mas não produzem uma saída tão alta quanto a produzida pelo primeiro demônio. Assim o *agente de decisão* escolhe o primeiro agente e classifica a entrada como uma letra A.

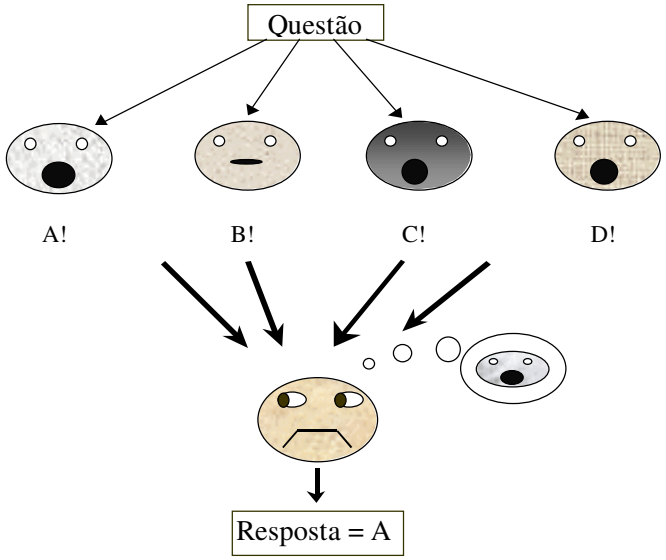


Figura 6.2 - Pandemonium de Selfridge

6.12.2 Generalização da arquitetura Pandemonium

ŚMIEJA, (1996) utilizou a idéia geral empregada no sistema *Pandemonium* de SELFRIDGE (1958), como mostrada acima, e a estendeu especificando um sistema modular de redes neurais. Cada um dos demônios mostrados na figura 6.2 é substituído por um

módulo adaptativo usando o princípio de dividir-para-conquistar. Cada módulo terá um grau de especialização para uma tarefa em particular.

Com esta extensão, conserva-se a especialização dividir-para-conquistar como uma estratégia de solução de problemas. No entanto, ŚMIEJA (1996) desenvolveu uma solução de distribuição de conhecimento baseada em conhecimento global, enquanto a solução de SELFRIDGE (1958) era baseada em conhecimento local. Este conhecimento não fornece simplesmente uma resposta escalar como no *Pandemonium* de SEFRIDGE (1958), mas sim um vetor como resposta, o que habilita cada agente a produzir, além do valor escalar, um valor de confiança.

Para tornar possível a distribuição do conhecimento, baseado em conhecimento global, na arquitetura *Pandemonium*, uma exigência necessária é que os agentes sejam reflexivos (ŚMIEJA & MÜHLENBEIN 1992; BEYER & ŚMIEJA 1993) . Isto significa que eles têm a propriedade ilustrada na figura 6.3.

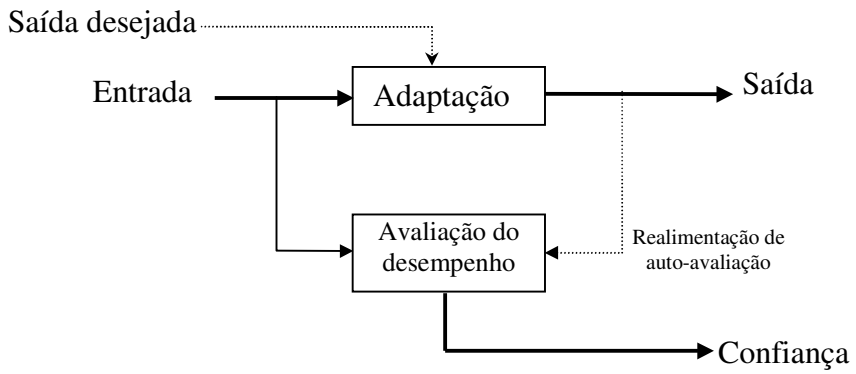


Figura 6.3 – Agente adaptativo reflexivo

Reflexão é o método pelo qual um agente observa seu próprio comportamento e produz uma informação relacionada com a qualidade de sua saída. A tarefa do agente é associar um elemento *x* de um conjunto com um elemento *y* de um outro conjunto, através de uma função interna *f(x).* O padrão *{(x,y)}* é fornecido pelo mestre e compõem o conjunto de treinamento. Após o treinamento, o agente fornece um saída *y* para um *x* previamente desconhecido. A auto-observação usa a informação *{(x,y)},* produzida para todo *x,* para realizar uma estimativa da precisão de *y*. Com esta capacidade extra, o agente é

transformado em um agente reflexivo. A informação produzida pela reflexão é chamada confidência (*c(x)* - grau de confiança na predição *f(x)*).

A função *c(x)* tenta estabelecer a zona de competência do agente no espaço *x*. A função é adaptativa e é ajustada para o conjunto de treinamento. O resultado da reflexão é discutido em mais detalhes em BEYER & ŚMIEJA (1993) .

O módulo no sistema Pandemonium de ŚMIEJA (1996) que produz a saída e confidência, dependendo do tipo de problema, foi chamado de *MINOS*. A principal aplicação do *Pandemonium* é em reconhecimento de estruturas de grande escala e em problemas com fortes descontinuidades no espaço de representação.

Um agente MINOS é mostrado esquematicamente na figura 6.4. Ele é composto por dois módulos, chamados trabalhador e monitor. O trabalhador aprende o mapeamento $x \mapsto y$ e o dever do Monitor é produzir a confidência *c(x)*. O trabalhador e o monitor podem ser, cada um, uma rede neural tipo MLP, treinada usando o procedimento de treinamento supervisionado (veja capítulo 2).

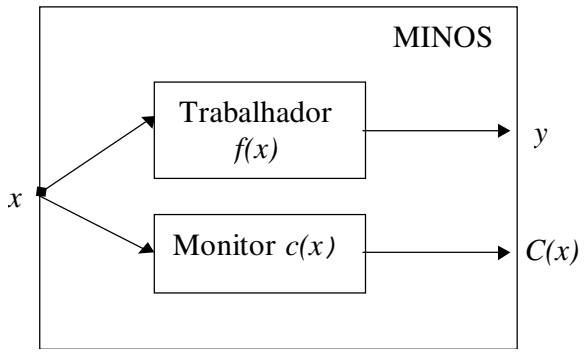


Figura 6.4 – Um agente MINOS

Um sistema *Pandemonium* de ŚMIEJA(1996) tem as seguintes características:

- os múltiplos módulos são escolhidos para resolver uma tarefa comum;
- os módulos não são interconectados e podem ser treinados e simulados independentemente;
- qualquer módulo MINOS pode também compor parte de outro conjunto de módulos MINOS para formar um novo *Pandemonium* na realização de outra

tarefa. Desta forma, funções genéricas são especializadas para obter um MINOS em um *Pandemonium,* podendo ser utilizadas para outro *Pandemonium.*

O *Pandemonium* é definido em termos da entrada e saída da tarefa a ser realizada e o número de módulos pode ser alterado dinamicamente. É tarefa do *Pandemonium* alocar a informação disponível sobre o problema aos módulos individuais *MINOS* – problema de decomposição – e escolher os módulos apropriados para fornecer uma resposta – problema de recombinação. A decomposição pode ser realizada de várias formas. A recombinação sempre é realizada escolhendo-se o *MINOS* com maior confiança para o padrão em questão.

Três métodos de decomposição dinâmica foram apresentados por ŚMIEJA (1996). Cada método tem suas vantagens e desvantagens, dependendo do problema a ser solucionado. Assim, um método foi usado para mapeamentos contínuos no espaço de entrada-saída, outro para mapeamento contínuo no espaço de entrada, mas com saídas booleanas (problema das duas espirais), e outro para problemas com alta dimensionalidade, entradas binárias e saídas booleanas (problema de paridade). Para problemas mutuamente não-exclusivos (sobreposição de classes), ŚMIEJA (1996) recomenda não decompor o espaço de entradas e sim apresentar todas as entradas a todos os agentes, como será utilizado nesta dissertação na concepção dos neuro-agentes construtivos.

6.12.3 Formas de integração de conceitos de redes neurais e sistemas multi-agentes

PHAM (1995) propôs uma abordagem neural multi-agentes chamada *neurOagent*, baseada em conceitos de macro-conexionismo, que proporcionam uma dupla integração. A primeira das integrações diz respeito ao nível de associação (aspectos conexionistas) e simbolismo (aspectos de representação do conhecimento, tais como: regras, redes semânticas e estruturas), onde unidades de processamento neuro-simbólicas completamente integradas proporcionam capacidade de aprendizagem e mecanismo de inferência distribuída, ao passo que a segunda diz respeito ao nível de conhecimento.

Segundo PHAM (1995), há duas formas principais de integrar as abordagens multi-agentes e aquelas baseadas em redes neurais: a primeira diz respeito ao estabelecimento de uma arquitetura distribuída com um protocolo de comunicação entre os módulos, onde as redes neurais são embutidas, e a segunda concentra-se na definição de uma entidade que

apresenta tanto propriedades de sistemas multi-agentes quanto de redes neurais. A abordagem neuro-agente proposta nesta dissertação corresponde ao segundo paradigma.

6.12.4 Dispositivo neurocomputacional híbrido: uma abordagem neuro-agente

O estudo empreendido nesta dissertação visa uma integração dos paradigmas computacionais conhecidos como redes neurais construtivas e teoria de agentes, isto é, uma visão de redes neurais construtivas sob a ótica da teoria de agentes. Esta integração visa explorar, no contexto da teoria de agentes, poderosos sistemas de representação e manipulação do conhecimento. Já no contexto da teoria conexionista, o que se pretende explorar é o potencial de aproximação universal e a grande flexibilidade de aplicação a problemas não-lineares, além dos poderosos mecanismos de adaptação e aprendizado. Esta combinação permitirá soluções compactas e eficientes para problemas de controle e identificação, em que a abordagem clássica de redes neurais tem apresentado limitações quanto a modelagem e aprendizado. Alguns dos resultados obtidos com esta abordagem são apresentados no capítulo 7.

Esta abordagem neuro-agente apresenta algumas similaridades com a estrutura *Pandemonium* apresentada, como mostrado a seguir:

- conserva o princípio dividir-para-conquistar como uma estratégia para atacar a complexidade dos problemas;
- o número de agentes é determinado dinamicamente;
- existem dois níveis hierárquicos: nível de cooperação ou nível de decisão (nível hierárquico superior) e nível de competição ou nível operacional (nível hierárquico inferior).

No entanto, a abordagem neuro-agente apresenta algumas características distintas das abordagens apresentadas no Apêndice A e encontradas na literatura:

- cada agente do nível de cooperação chamado *agente de decisão* é responsável por perceber o ambiente e tomar decisões específicas sobre o mesmo, como por exemplo: no controle da velocidade e posição de um robô haverá dois agentes de decisão, um responsável pela estratégia de controle de velocidade e o outro

pela estratégia de controle de posição. Desta forma, o número de *agentes de decisão* corresponde ao número de decisões a serem tomadas pelo sistema multi-agentes;

- cada *agente de decisão* processa, como saída, uma combinação linear das saídas dos agentes do nível de competição, chamados *agentes trabalhadores*. O número de *agentes trabalhadores é* determinado dinamicamente de acordo com a tarefa a ser realizada;
- os agentes trabalhadores são treinados à medida que são requisitados pelos *agentes de decisão* para compor o sistema multi-agentes, podendo ser ou não treinados novamente (*retroajuste*) quando outros *agentes trabalhadores* são requisitados para contribuírem na realização da mesma tarefa;
- cada *agente trabalhador* é projetado para trabalhar em determinado sistema multi-agentes para a realização de uma determinada tarefa, podendo ser utilizados em outras tarefas desde que sejam treinados novamente neste novo sistema multi-agentes, ao contrário da estrutura *Pandemonium,* na qual os módulos são treinados individualmente e pode se formar um novo *Pandemonium* sem a necessidade de novo treinamento;
- cada *agente trabalhador* recebe o mesmo conjunto de entradas (percepções do ambiente) com a função de identificar os aspectos relevantes não analisados pelos outros *agentes trabalhadores* (saída residual);
- tanto os *agentes trabalhadores* como os *agentes de decisão* são modelados como um *agente comunicativo* (para maiores detalhes veja apêndice A).

Dentre as principais características da abordagem neuro-agente, podemos citar:

- cada agente trabalhador é uma representação de um neurônio na camada escondida de uma rede neural construtiva;
- não existe comunicação entre os agentes trabalhadores, mas uma comunicação entre cada *agente de decisão* e os *agentes trabalhadores* através de passagem de parâmetros. Esta comunicação torna-se necessária para que o processo de competição e cooperação, que norteia o processo, possa ser eficientemente coordenado;

- dos processos de competição e cooperação emergem fenômenos de auto-organização;
- os agentes construtivos são geralmente inicializados com o número de agentes no nível de decisão igual ao número de decisões (este número mantém-se inalterado durante todo o processo de treinamento) e um agente no nível operacional. À medida que ocorre a interação de cada *agente de decisão* com o ambiente, novos *agentes trabalhadores* podem ser requisitados pelo *agente de decisão* para compor o sistema multi-agentes, caso o modelo do mundo gerado não esteja adequado para o atendimento dos requisitos de desempenho.
- cada *agente trabalhador* cria um modelo particular do mundo ao projetar as entradas em uma direção tal que esta projeção possa refletir aspectos importantes do ambiente, ainda não percebidas pelos outros *agentes trabalhadores* (ver conceitos de projeção no capítulo 5). Durante este processo de descoberta de porções desconhecidas do ambiente, os agentes competem entre si de forma a expandir seus conhecimentos sobre o mundo. Após a convergência deste processo de competição, isto é, quando nenhum *agente trabalhador* é capaz de melhorar seu modelo do ambiente, inicia-se o processo de cooperação. Neste processo, cada *agente de decisão* realiza uma combinação linear dos modelos do mundo de cada *agente trabalhador* para gerar um modelo para o mundo que seja o mais próximo possível do desejado. Caso este modelo ainda não esteja de acordo com os critérios estabelecidos, um novo *agente trabalhador* é adicionado ao nível de competição e o processo de competição e cooperação é repetido;
- após o modelo do mundo ter sido gerado, os *agentes de decisão* verificam o desempenho de cada *agente trabalhador*. De acordo com o desempenho dos *agentes trabalhadores* (por exemplo: contribuição para altas freqüências, geralmente associadas a ruído não-informativo) o *agente de decisão* pode eliminá-lo ou não do sistema multi-agentes, sempre buscando melhorar o desempenho deste sistema. Caso algum *agente trabalhador* seja retirado, todos os outros sofrerão um treinamento de reforço, mas nenhum novo agente será adicionado.

A estrutura de processamento resultante da aplicação de técnicas inspiradas em teoria de agentes no desenvolvimento de estratégias construtivas de treinamento para modelos conexionistas será chamada de *dispositivo neurocomputacional híbrido*. A denominação de dispositivo neurocomputacional em lugar de rede neural construtiva, conforme mencionado anteriormente, se justifica pelo fato da estrutura resultante, assim como a forma de obtenção baseada na teoria de agentes, ser diferente daquela empregada tradicionalmente na área de redes neurais artificiais. Além disso, o termo híbrido está associado ao fato de que cada *agente trabalhador* tem habilidades diferentes para tratamento das entradas (percepções do ambiente), isto é, cada agente produz um mapeamento não-linear diferente. Um dispositivo neurocomputacional híbrido com um *agente de decisão* e dois *agentes trabalhadores* é apresentado na figura 6.5.

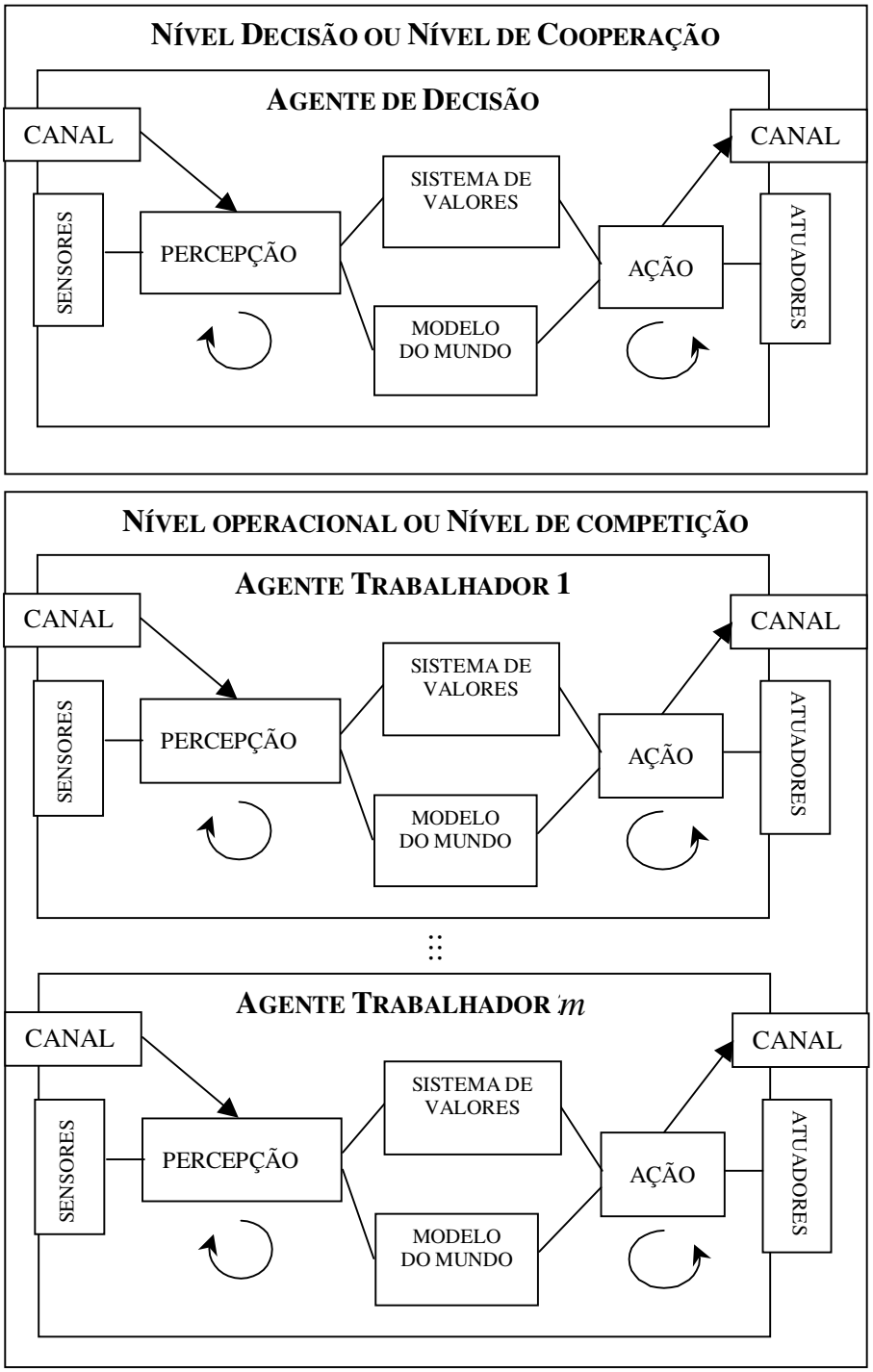


Figura 6.5 - Estrutura do dispositivo neurocomputacional híbrido: CANAL representa o canal de comunicação entre o agente de decisão e o agente trabalhador

6.13 Estrutura dos agentes

O estudo da estrutura que compõe o dispositivo neurocomputacional híbrido baseado no conceito de neuro-agente pode ser realizado através de uma abordagem modular ou funcional.

6.13.1 Abordagem modular

A estrutura modular, conforme apresentada na figura 6.5, consiste de quatro principais elementos para cada *agente de decisão* e *trabalhador*:

- *Módulo de percepção no nível de cooperação* (MP)– recebe um conjunto ou par {entrada, saída} oriundo dos sensores e compara as expectativas geradas pelo modelo do mundo ou modelo do ambiente. Em MP, são integradas as similaridades e diferenças entre as observações e as expectativas, de modo a detectar eventos, reconhecer padrões, objetos e inter-relacionamentos no ambiente;
- *Módulo de modelo do mundo no nível de cooperação* (MA) – oferece, a cada instante, a melhor estimativa do estado do ambiente, vista pelo agente de decisão. Esta estimativa é gerada com base no modelo individual de cada agente no nível de competição;
- *Módulo de ação no nível de cooperação* – com base no modelo do ambiente, gera-se uma ação a ser tomada pelo sistema multi-agentes;
- *Módulo de sistema de valores no nível de cooperação* – determina o que é bom ou ruim, recompensa ou pune os agentes no nível de competição. Além disso, realiza uma avaliação do estado atual do ambiente, podendo disparar um mecanismo para criação de novos agentes no nível de competição;
- *Módulo de percepção no nível de competição* – recebe um conjunto ou par {entrada, saída} do agente no nível de competição, oriundo dos sensores, e compara as expectativas geradas pelo modelo do ambiente;
- *Módulo de modelo do mundo no nível de competição* - oferece a cada instante a melhor estimativa do estado do ambiente. Cria uma base de dados sobre o ambiente. É capaz de atender a requisições de informações solicitadas pelos

agentes de decisão. Com base no sistema de valores, deve ser capaz de gerar planos de projeção dos dados de entrada, representando perspectivas de visualização da informação presente no espaço de entrada original. As direções dos planos de projeção iniciais são calculadas por índices de projeção ou por direções uniformemente distribuídas ao longo de um círculo de raio unitário. Os resultados dos planos gerados são preditos e avaliados pelo módulo de sistema de valores. Então, seleciona-se o plano com melhor avaliação, o qual é executado através dos atuadores, calculando-se os novos parâmetros do agente;

- *Módulo de sistema de valores no nível de competição* – realiza uma avaliação do desempenho do modelo atual do ambiente frente às expectativas.

6.13.2 Abordagem funcional

As figuras 6.6 e 6.7 mostram a estrutura funcional do *dispositivo neurocomputacional híbrido,* que consiste de três elementos básicos:

- *O envelope de comunicação e ativação,* que assegura um padrão de comunicação entre os *agentes de decisão* e os *agentes trabalhadores* e que controla o estado *dos agentes trabalhadores*;
- A *zona nominal,* que descreve o agente (nome, sinônimo);
- O *processo interno,* que determina a especificação funcional.

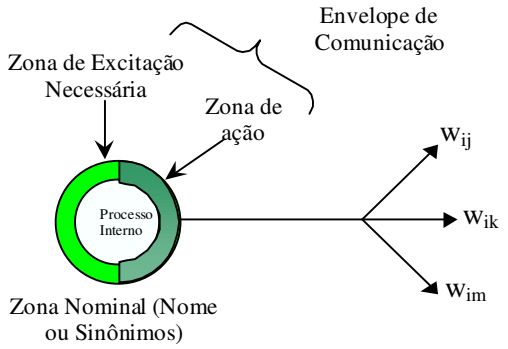


Figura 6.6 - Estrutura funcional de um agente trabalhador

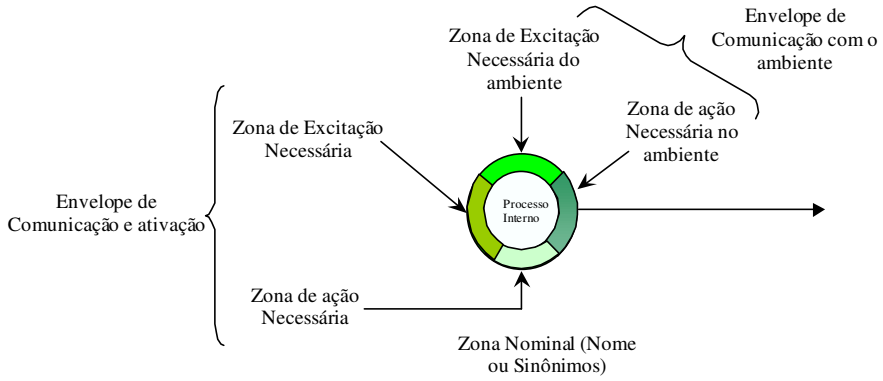


Figura 6.7 - Estrutura funcional de um agente de decisão

A diferença da estrutura funcional entre um *agente de decisão* e um *agente trabalhador*, conforme ilustrado acima, está *no envelope de comunicação com o ambiente,* necessária para o *agente de decisão*. A função deste é possibilitar que os *agentes de decisão* possam perceber o ambiente e agir sobre o mesmo. *Os agentes trabalhadores* não necessitam do *envelope de comunicação com o ambiente*, pois não comunicam-se diretamente com o ambiente, somente realizam comunicação com os *agentes de decisão*.

6.13.2.1 Envelope de comunicação e ativação

O envelope de comunicação e ativação contém:

1. Zona de excitação necessária ou Zona de percepção;
2. Zona de ação necessária.

A zona de excitação necessária corresponde a áreas que irão receber os estímulos, sejam eles externos (percepções do ambiente) ou não. No caso dos agentes trabalhadores estes estímulos são oriundos dos *agentes de decisão*, mais especificamente da zona de ação necessária. Para que os *agentes trabalhadores* possam se tornar ativos (realizando alguma tarefa) algumas condições devem ser validadas. Estas condições são necessárias para garantir coerência no processo de cooperação e competição, além de permitir que recursos computacionais não sejam gastos com agentes ativos, mas não realizando nenhuma tarefa útil.

A zona de ação necessária corresponde à área por onde os agentes irão agir no ambiente. No caso dos agentes trabalhadores, esta ação será realizada sobre o *agente de decisão*, mais especificamente na zona de excitação necessária. Além do resultado da ação, cada agente trabalhador passará para o *agente de decisão* um parâmetro que o identifica.

6.13.2.2 Zona nominal

A zona nominal contém o nome do agente e seus sinônimos:

- *Rótulo*: é a principal identificação do agente. Este é o nome principal do agente que será usado para designá-lo durante a simulação;
- *Sinônimos*: são usados para atribuir outros nomes para o agente, caso seja necessário, como por exemplo, um nome que o usuário deseja ao invés daquele utilizado na simulação.

6.13.2.3 Processo interno

Há dois tipos de ação: ação cognitiva e ação produtiva. A ação cognitiva tem efeito direto sobre o desempenho do agente. Nos agentes trabalhadores, tem o objetivo de descobrir uma direção de projeção e a forma da função de ativação, através de polinômios de Hermite ou *splines* suavizantes, que melhor aproximam os dados projetados, recebidos pela zona de excitação necessária. Já nos *agentes de decisão*, o objetivo é descobrir os pesos que, multiplicados pela ação de cada *agente trabalhador*, produzem uma saída correta ou o mais próxima possível da correta. Uma das formas possíveis de descobrir estes pesos é utilizar a pseudo-inversão (para maiores detalhes, veja capítulo 5).

A ação produtiva produz uma saída baseada na direção de projeção e na função de ativação aproximada (no caso dos agentes trabalhadores) ou os pesos (no caso dos *agentes de decisão*), obtidos na ação cognitiva. A ação produtiva somente poderá ocorrer depois que a ação cognitiva ocorreu.

6.14 O comportamento do neuro-agente

O comportamento básico do dispositivo neurocomputacional híbrido é determinado pelo estado dos parâmetros, pelos estados dos agentes trabalhadores, pelo mecanismo de propagação dos estímulos, pelo aprendizado e pelo tipo de comunicação.

6.14.1 Parâmetros dos neuro-agentes

Os principais parâmetros que conduzem ao comportamento resultante do neuro-agente são os seguintes:

1. Ordem de propagação;
2. Monotonicidade;
3. Período refratário e fase de propagação;
4. Estímulo externo.

O parâmetro "ordem de propagação" pode ser especificado para *sentido direto* de tal forma que todos os *agentes trabalhadores* e de *decisão* sejam validados. Assim, cada *agente trabalhador* executa uma ação produtiva e propaga a informação para *o agente de decisão*, que age sobre o ambiente. Entretanto, é possível mudar esta ordem para *sentido inverso,* de tal forma que cada agente possa atualizar seus parâmetros.

Um raciocínio monotônico é um tipo de raciocínio baseado na suposição de que, uma vez que um fato tenha sido determinado, não será alterado mesmo que uma informação nova pertinente apareça. Isto limita seriamente um grande número de aplicações para as quais a evolução de vários estados deve ser levado em conta. Por outro lado, um raciocínio não-monotônico é um tipo de raciocínio no qual as suposições são sistematicamente ajustadas à luz de novas informações.

Tradicionalmente, sistemas baseados em conhecimento consistem de duas partes principais: uma máquina de inferência e uma base de conhecimento estática. Somente a máquina de inferência pode gerenciar o mecanismo de raciocínio e, portanto, a noção de monotonicidade e não-monotonicidade. As abordagens tradicionais de redes neurais utilizam um raciocínio monotônico, uma vez que, fixada a arquitetura, mesmo que parte da informação não possa ser representada na arquitetura estabelecida, não existe nenhum

mecanismo que possa alterar a arquitetura da rede neural, após inicializado o treinamento. Em muitas aplicações, como por exemplo em identificação e controle, se estas informações coletadas forem relevantes o comportamento global do sistema pode inclusive comprometer a estratégia de controle adotada. Desta forma, tendo uma rede neural ou qualquer outro sistema estritamente monotônico, pode-se inviabilizar a obtenção de solução para o problema. Portanto, deveria ser possível para a rede neural adaptar-se a cada novo contexto.

A abordagem baseada em *dispositivos neurocomputacionais híbridos* é uma das técnicas que pode gerenciar tanto raciocínio monotônico quando não-monotônico. A não-monotonicidade é diretamente gerenciada pelos *agentes de decisão*. O parâmetro de monotonicidade é então alterado dinamicamente, conforme o contexto em vigor.

Embora um período refratário exista em sistemas neurobiológicos, no caso de sistemas conexionistas artificiais este fator é raramente simulado (PHAM, 1995). Quando um período refratário é configurado para "sim", um *agente trabalhador*, que já está validado ou inibido, não pode ser validado ou inibido de novo durante este período. Se o período refratário é configurado para "não", o *agente trabalhador* ou *agente de decisão* é sensível a qualquer novo estímulo e não leva em conta o período refratário. Esta situação é interessante, por exemplo, para ajudar a gerenciar o processo de cooperação e competição na abordagem de agentes construtivos.

Por outro lado, a noção de fase indica se o agente trabalhador já realizou a fase de atualização dos parâmetros. Neste caso, o agente trabalhador apresentará um período refratário temporário, isto é, ficará um período inerte. O período refratário e a fase podem participar no controle de laços iterativos.

Os estímulos externos permitem a validação da zona de excitação necessária dos *agentes de decisão*. Estímulos são informações capturadas pelos sensores.

6.14.2 Estado do neuro-agente

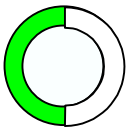
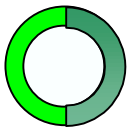
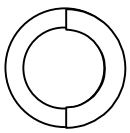
Para qualquer dado instante, cada agente trabalhador é caracterizado por seu estado, o qual é estabelecido a partir dos estímulos recebidos no seu envelope de comunicação e ativação.

Os envelopes de comunicação e ativação do *agente trabalhador* e do *agente de decisão* podem ter diferentes configurações, conforme os estímulos que eles recebem na zona de excitação necessária, ou realizam na zona de ação necessária.

6.14.2.1 Definindo o estado do neuro-agente

Dois são os estados possíveis para o neuro-agente:

1. Validado;
2. Inibido.

Configuração do agente	Zona de Excitação Necessária	Zona de ação necessária	Estado do agente	Motivo
	Validada	Sem conexão	Validado	Realizando ação cognitiva
	Sem conexão	Validada	Validado	Perda da percepção – comportamento padrão *
	Validada	Validada	Validado	Realizando ação produtiva
	Inibido	Inibido	Inibido	O agente está inerte

* Ocorre quando o agente de decisão tem seus sensores danificados ou quando um *agente trabalhador* perdeu a conexão com o *agente de decisão*. No contexto de aplicação desta dissertação, esta situação não será contemplada.

A zona de excitação necessária é validada quando todas as condições para esta zona, isto é, todas as condições descritas no *agente trabalhador* ou de *decisão* são validadas. Se o agente está conectado, significa que ele está realizando alguma função.

6.14.3 Mecanismo de propagação

Há dois mecanismos de propagação, a saber:

- Propagação de sentido direto;
- Propagação de sentido inverso.

Na propagação de sentido direto, a informação é propagada diretamente, sem atualização dos parâmetros, e corresponde ao passo direto no treinamento de redes neurais. Enquanto isso, na propagação de sentido inverso a resposta do sistema é conhecida. Com base nesta resposta e nos dados de entrada, os parâmetros dos agentes são atualizados de tal forma a conseguir modelar o ambiente e alcançar os objetivo previamente definidos.

6.14.4 Aprendizagem

Com a abordagem via dispositivos neurocomputacionais híbridos, é possível projetar uma base de conhecimento através do aprendizado. O aprendizado proporciona três resultados:

- automaticamente estabelece os pesos das conexões, como nos modelos conexionistas tradicionais;
- automaticamente estabelece a função de ativação do neuro-agente;
- automaticamente estabelece a dimensão da rede.

O algoritmo de aprendizagem usado para ajuste de parâmetros nos dispositivos neurocomputacionais híbridos é descrito no capítulo 5. A pseudo-inversa é utilizada para encontrar os pesos que ponderam as respostas dos agentes trabalhadores e descobrir os coeficiente dos polinômios de Hermite, quando estes forem utilizados para produzir a forma da função de ativação.

Após o treinamento, será possível extrair da estrutura de agentes resultante as direções de projeção relevantes, as contribuições individuais de cada *agente trabalhador* para o resultado e o número de agentes trabalhadores utilizados. Além disso, a capacidade de generalização para dados ruidosos e também para aqueles não apresentados durante o treinamento podem ser verificadas, conforme mostrado no capítulo 7.

6.14.5 Tipo de comunicação

Em sistemas multi-agentes, dois tipos principais de comunicação podem ser distinguidos: sistema de quadro (*blackboard* – escreve em uma região da memória e permite que todos os agentes possam ler) ou sistemas de passagem de mensagem. Por definição, o sistema de quadro não é adotado em sistemas baseados em neuro-agente. O sistema de passagem de mensagem é utilizado.

6.15 Algoritmo construtivo baseado na teoria de agentes

1º Passo – O *agente de decisão* percebe o ambiente;
2º Passo – Enquanto não conseguir criar um modelo adequado do mundo;
    2.1º Passo – Cria um novo *agente trabalhador* no nível de competição;
        2.1.1º Passo – Realiza a fase de competição entre os *agentes trabalhadores*;
        2.1.2º Passo – Realiza a fase de cooperação;
3º Passo – Realiza a fase de poda;
4º Passo – Realiza um treinamento de reforço, mas sem adicionar novos agentes.

6.15.1 Fase de Cooperação

Nesta fase, cada neurônio (agente) tenta contribuir para melhorar o desempenho da rede como um todo, diminuindo assim o erro de aproximação. Para tanto, tomando-se o *n*-ésimo neurônio, são ajustados dois conjuntos de parâmetros:

- os polinômios de Hermite ou *splines* suavizantes: fixada a direção de projeção, o objetivo é resolver a equação apresentada a seguir, isto é, tenta-se obter a melhor aproximação para os dados projetados (veja capítulo 5):

$$\min_{f_n} \frac{1}{N} \sum_{l=1}^{N} (d_l - f_n(v_n^T x_l))^2 + \lambda_n \phi(f_n) \tag{6.1}$$

- a direção de projeção : fixada a função de ativação, o objetivo é resolver a equação a seguir, isto é, tenta-se descobrir iterativamente através de métodos de otimização com busca unidimensional (por exemplo, Newton Modificado), a melhor direção de projeção.

$$\min_{v_n} \frac{1}{N} \sum_{l=1}^{N} (d_l - f_n(v_n^T x_l))^2 \tag{6.2}$$

A atualização iterativa de f_n e v_n é realizada até que se consiga convergência, a qual tem garantias teóricas de existência (HUBER, 1985).

6.15.2 Fase de Competição

Nesta fase, cada neurônio (agente) tenta rever sua posição, devido a mudanças ocorridas, tais como: a entrada de novos neurônios (agentes), alteração do comportamento de outros neurônios (agentes). Também nesta fase, f_n e v_n são atualizados para todos os N_c neurônios já criados ($n=1,\ldots,N_c$).

6.15.3 Fase de poda

Nesta fase, define-se uma porcentagem mínima de qual deverá ser a contribuição de cada agente (neurônio) para a saída da rede. Caso algum agente tenha uma contribuição inferior ao mínimo especificado, ele será eliminado da arquitetura da rede. Além disso, verifica-se a contribuição de agentes para a representação de ruído: caso este valor seja elevado, o agente também será eliminado.

6.16 Abordagem tradicional de redes neurais × abordagem baseada em teoria de agente

Algumas das vantagens desta nova abordagem, quando comparada à visão tradicional de redes construtivas, são:

- *Uma nova visão da abordagem construtiva* – dispositivos neurocomputacionais híbridos representam uma abordagem eficiente no tratamento de problemas de alta complexidade, através do aumento da flexibilidade da função de ativação dos neurônios da camada intermediária, conforme descrito no capítulo 5. Com a introdução de conceitos de agentes, os mecanismos envolvidos no processo de construção da rede neural podem ser entendidos através de processos de cooperação e competição entre os agentes. Esta nova visão, além de ser intuitiva, pode facilitar na identificação e compreensão da operação global da rede, bem como do papel de seus elementos básicos, aqui denominados *neuro-agentes*;
- *Permite que modificações estruturais possam ser realizadas de uma forma mais fácil* – modificações podem ser incorporadas à estrutura de processamento da rede neural, pela inclusão ou poda automática de neurônios. Além disso, é possível vislumbrar novos procedimento de análise e validação para as modificações propostas, com menos esforço e recorrendo-se às estruturas modulares envolvidas;
- *Melhor utilização dos recursos* – a quantidade de recursos computacionais utilizada no processo de geração do dispositivo neurocomputacional híbrido é maior que aquela empregada no treinamento de uma rede neural convencional. No entanto, a estrutura de processamento resultante é sempre mais econômica em termos de uso de recursos computacionais. Em outras palavras, recorre-se a um custo computacional maior na obtenção de uma solução computacional mais parcimoniosa. Além disso, no caso do dispositivo neurocomputacional híbrido, existe uma relação custo-benefício muito bem definida associada à solução obtida (quanto maior a quantidade de recursos computacionais, maior a

qualidade da solução) o que não ocorre no caso de redes neurais convencionais (GONÇALVES *et al*., 1998);

- *Facilidade para inserção ou extração de conhecimento* – extrair ou inserir conhecimento em uma rede neural é uma tarefa muito complexa. Com a estrutura proposta é possível inserir ou extrair conhecimento de uma forma mais direta, já que a rede neural é agora interpretada como um sistema multi-agentes, em que cada agente tem a sua individualidade e aceita uma interpretação particular;
- *Uma visão coerente com as abordagens de sistemas inteligentes* – aspectos parciais da inteligência, tais como raciocínio usando conhecimentos vagos ou incompletos, aprendizado e predição vem sendo estudados em conjunto, no contexto de inteligência computacional (ZURADA *et al.*, 1994). Esta nova abordagem apresenta-se como uma ferramenta promissora para explicar tais aspectos da inteligência. Tanto os agentes trabalhadores quanto os agentes de decisão podem ser modelados segundo o modelo de ALBUS (1991) para sistema inteligentes (figura 6.8), o qual possui quatro módulos que se interconectam: percepção sensorial, julgamento de valor, modelo do ambiente e geração de comportamento. Desta forma, suspeita-se também que os neuro-agentes, base do projeto de implementação de dispositivos neurocomputacionais híbridos, possa ser uma das possíveis arquiteturas para sistemas inteligentes. Uma visão mais detalhada do modelo proposto por ALBUS (1991) pode ser encontrado em SOUZA E SILVA (1998);
- *Facilidade para extensão* – A extensão de redes neurais construtivas para contemplar outros tipos de interação dos neuro-agentes, gerando outras arquiteturas, torna-se uma real possibilidade a partir da abordagem proposta nesta dissertação.

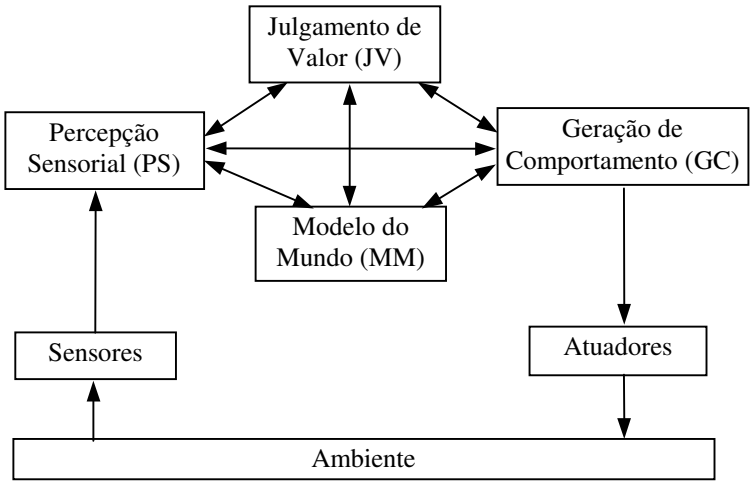


Figura 6.8 - Módulos de um Sistema Inteligente proposto por ALBUS (1991)

6.17 Dispositivos neurocomputacionais e otimização não-linear

Para explorar adequadamente a capacidade de representação apresentada pelos dispositivos neurocomputacionais (neuro-agentes) descritos acima, é necessário desenvolver algoritmos de otimização que permitam tratar o poder de adaptação presente nessas estruturas. Para tanto, os mais eficientes métodos de otimização não-linear (LUENBERGER, 1989) são comumente empregados.

Os problemas de otimização resultantes são de difícil solução por apresentarem as seguintes características:

- são inerentemente não-lineares;
- são geralmente problemas de grande porte, por envolverem um número muito grande de variáveis e/ou restrições;
- são problemas não-convexos, conduzindo à existência de mínimos locais, nem sempre satisfatórios para os propósitos da aplicação;
- as soluções associadas não podem ser obtidas algebricamente, exigindo a aplicação de algoritmos iterativos, com convergência nem sempre garantida para tempo de processamento finito;
- as soluções associadas demandam um elevado custo computacional;

- geralmente a solução não é única.

Estes aspectos, vinculados aos problemas de otimização que envolvem dispositivos neurocomputacionais, devem ser considerados em conjunto no processo de geração da solução.

6.18 Aplicação dos dispositivos neurocomputacionais

Várias são as possíveis aplicações dos dispositivos neurocomputacionais. No entanto, nesta dissertação elas se restringem às seguintes:

- aproximação de funções;
- identificação e controle de sistemas dinâmicos não-lineares.

6.18.1 Dispositivos neurocomputacionais e teoria de aproximação de funções

As funções constituem as ferramentas matemáticas básicas para descrição e análise de processos físicos. Enquanto em alguns casos estas funções são conhecidas explicitamente, muitas vezes é necessário extrair um conjunto de informações acerca do processo e construir aproximações baseadas neste conjunto.

Como um ramo da análise funcional, a teoria de aproximação de funções tornou-se uma ferramenta teórica fundamental na implementação de métodos de aproximação capazes de aplicarem eficientemente os poderosos recursos computacionais atualmente disponíveis. Dispositivos neurocomputacionais de processamento numérico, arquitetura em camadas e fluxo de informação com e sem realimentação produzem estruturas de aproximação de funções com grande poder de adaptação e capacidade de representação de mapeamentos não-lineares.

Como conseqüência, sempre que métodos tradicionais, derivados da teoria de sistemas lineares, não apresentarem um desempenho satisfatório no tratamento de problemas envolvendo, por exemplo, controle e identificação de sistemas dinâmicos, os dispositivos neurocomputacionais despontam como alternativas bastante competitivas,

devido principalmente à sua capacidade de representação de comportamentos não-lineares arbitrários (HORNIK *et al*., 1990).

6.18.2 Dispositivos neurocomputacionais e identificação de sistemas dinâmicos

A teoria de identificação de sistemas engloba o estudo de modelos de aproximação para sistemas dinâmicos a partir de dados amostrados. A escolha da estrutura do modelo de aproximação é guiada por conhecimentos preliminares acerca do sistema que gerou os dados, conduzindo, por exemplo, a uma classe de modelos paramétricos. Na ausência de conhecimento preliminar, modelos não-paramétricos são geralmente empregados.

É sabido que, em aplicações práticas, é impossível obter um modelo capaz de descrever o sistema exatamente (SJÖBERG, 1995). O que se busca, na verdade, é obter a melhor aproximação do sistema real, baseada em algum critério. Apesar de quase nunca corresponder à melhor aproximação, é comum assumir que o sistema a ser identificado é linear, viabilizando a aplicação de resultados bem sedimentados e abrangentes da teoria de sistemas lineares.

Em muitos casos, no entanto, existe a necessidade de se utilizar modelos não-lineares (paramétricos ou não), tornando a tarefa de identificação muito mais complexa, já que a maior parte dos resultados válidos para o caso linear não mais se aplicam (VIDYASAGAR, 1993). Uma razão para a maior complexidade associada ao uso de modelos de aproximação não-lineares é a existência de um número infinitamente maior de alternativas para a estrutura do modelo, já que modelos não-lineares correspondem a todos os modelos que não são lineares, ou seja, a não-linearidade pode se manifestar de inúmeras formas.

Com isso, modelos de aproximação não-lineares precisam ser mais flexíveis, o que requer um número maior de parâmetros que o caso linear, de forma que classes abrangentes de não-linearidades possam ser adequadamente representadas. Redes neurais artificiais (modelos paramétricos) e dispositivos neurocomputacionais (modelos não-paramétricos) representam alternativas mais flexíveis e confiáveis quando comparadas a modelos

baseados na definição arbitrária de estruturas paramétricas com não-linearidades previamente definidas. BILLINGS (1980) apresenta uma coletânea de modelos de identificação deste último tipo, enquanto que NARENDRA & PARTHASARATHY (1990) e CHEN & BILLINGS (1992) descrevem alguns modelos de identificação utilizando redes neurais artificiais. VON ZUBEN (1996) sugeriu estratégias de implementação de dispositivos neurocomputacionais como modelos de identificação.

A discussão acima é válida tanto para sistemas dinâmicos de tempo contínuo como tempo discreto, mas em função do tipo de modelos de redes neurais a serem utilizados, os resultados a serem desenvolvidos nesta dissertação envolvem apenas sistemas dinâmicos de tempo discreto.

6.18.2.1 Aplicações do modelo de identificação

Existem quatro aplicações básicas para os modelos de identificação de sistemas dinâmicos:

- utilização em tarefas de predição ou simulação;
- auxílio no desenvolvimento de controladores para sistemas dinâmicos (método de controle indireto);
- participação na composição de um controlador adaptativo para sistema dinâmico (controle adaptativo direto);
- utilização em avaliação de desempenho e diagnóstico de comportamento anormal do sistema dinâmico.

6.18.3 Dispositivos neurocomputacionais e controle de processos

Uma descrição detalhada de aplicação de redes neurais clássicas a controle de processos pode ser encontrada em BARTO (1990), HUNT *et al.* (1992), SONTAG (1993), ZBIKOWSKI (1994) e NARENDRA & MUKHOPADHYAY (1994).

A aplicação de redes neurais a controle de processos requer a definição preliminar de um paradigma básico de controle, sendo adotada neste estudo a formalização utilizada

por SONTAG (1993) e apresentada na figura 6.9. No caso linear, este paradigma de controle é muito difundido na descrição de sistemas realimentados (DOYLE *et al*., 1989; IGLESIAS & GLOVER, 1991).

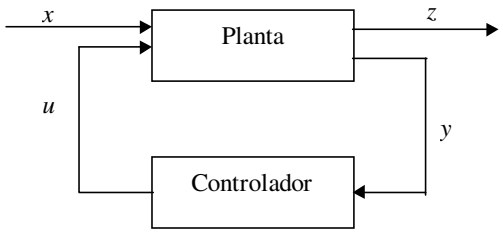


Figura 6.9 - O paradigma de controle

A planta representa o sistema a ser controlado. O sinal *x* contém todas as entradas que não podem ser diretamente afetadas pelo controlador, incluindo perturbações e sinais de referência. O sinal *y* representa as variáveis de saída medidas, podendo conter também sinais de entrada via conexão direta. O sinal *u* corresponde à parte da entrada que pode ser manipulada, enquanto que o sinal *z* engloba os objetivos de controle. Por exemplo, o sinal *z* pode representar a diferença entre uma certa função dos estados do sistema e uma função de referência, sendo que o objetivo de controle é a anulação deste sinal.

A introdução de redes neurais convencionais ou dispositivos neurocomputacionais como componentes da estrutura de controle está associada a um maior detalhamento do bloco denominado controlador, conforme apresentado na figura 6.10.

O controlador é dividido em duas partes: o sintonizador e o sistema ajustável. O sistema ajustável vai operar sobre a entrada y e representa um sistema dinâmico ou um mapeamento estático com parâmetros ajustáveis. Os parâmetros ajustáveis estão agrupados em um vetor **w**, atualizado pelo sintonizador. Há também um parâmetro discreto k que determina a flexibilidade do controlador, seja em termos da dimensão do vetor **w** ou em termos do número de variáveis de estado do controlador. O sintonizador ajusta os parâmetros de acordo com uma lei de adaptação.

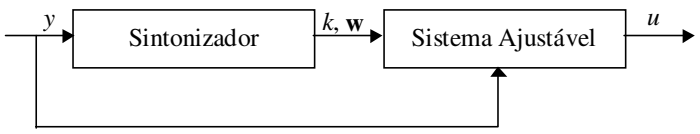


Figura 6.10 O controlador adaptativo

Tipicamente, é interessante operar com uma classe de modelos suficientemente flexíveis para permitir a representação de uma variedade de comportamentos estáticos ou dinâmicos. É neste caso que as redes neurais artificiais e os dispositivos neurocomputacionais surgem como alternativas bastante competitivas, principalmente quando associações não-lineares significativas entre parâmetros estão presentes.

Vale ressaltar, no entanto, que esta estratégia de controle não-linear adaptativo descrita acima, embora permita a utilização de redes neurais ou dispositivos neurocomputacionais em sua implementação, não possui nenhuma propriedade especificamente vinculada às particularidades presentes nestas estruturas conexionistas. Logo, todo o tratamento de controle de processos constante do capítulo 7, empregando dispositivos neurocomputacionais híbridos, deve sempre ser entendido como uma alternativa de abordagem, dentre as disponíveis.

6.19 Visão sintética do capítulo

Neste capítulo, foram apresentados alguns dos vários conceitos sobre agentes, sendo que uma visão mais ampla é apresentada no apêndice A. Posteriormente, foram explicitadas as vantagens da utilização dos agentes na teoria conexionista e o modelo do neuro-agente proposto, unidade básica do dispositivo neurocomputacional híbrido. Em seguida, realizamos um esforço no sentido de identificar neste modelo proposto os principais aspectos que possam contribuir para controle e identificação de sistemas dinâmicos.

Capítulo 7

Aplicação a problemas de identificação e controle de sistemas dinâmicos

7.1 Introdução

Conforme já mencionado, redes neurais artificiais constituem uma tecnologia emergente que acena com perspectivas e realizações antes consideradas utópicas, como por exemplo, capacidade de realizar tarefas de alto nível cognitivo, reconhecimento de padrões desconexos e não-linearmente separáveis, controle e identificação de sistemas dinâmicos não-lineares. De acordo com o grau de parametrização das redes neurais, os modelos apresentados e analisados em capítulos anteriores podem ser divididos em dois grupos: redes neurais tradicionais e dispositivos neurocomputacionais híbridos. Rede neurais tradicionais são aquelas em que as funções de ativação e a dimensão da rede (número de neurônios na camada intermediária) são fixas e definidas arbitrariamente pelo usuário antes do início do treinamento, ao passo que nos dispositivos neurocomputacionais híbridos as funções de ativação podem assumir qualquer forma e a dimensão é determinada dinamicamente para cada tarefa específica, isto é, os dispositivos neurocomputacionais híbridos são aqueles que não impõem forte restrições com relação ao modelo de aproximação, garantindo a flexibilidade necessária para aproximar uma ampla classe de funções, restritas a regiões compactas do espaço de aproximação.

Este capítulo apresenta alguns resultados referentes à aplicação de modelos lineares, redes neurais tradicionais e dispositivos neurocomputacionais híbridos, desenvolvidos nos capítulos anteriores, a problemas de aproximação de funções, predição de séries temporais, identificação de sistemas dinâmicos e controle de processos. Em todos os casos, os modelos são obtidos pela aplicação de métodos de treinamento supervisionado, a partir de um conjunto de dados de treinamento e teste. A título de ilustração e comparação, redes neurais recorrentes também são empregadas.

Os problemas propostos a seguir exigem soluções dotadas invariavelmente de estruturas não-lineares. Isto não significa , no entanto que métodos puramente lineares não tenham importância prática, já que estas vêm sendo empregados com sucesso em inúmeras aplicações (BALDI & HORNIK, 1995). Além disso, os resultados deste capítulo não podem ser interpretados como conclusivos, visto que o propósito básico aqui é essencialmente o de salientar o grande potencial dos dispositivos neurocomputacionais e as variadas possibilidades de aplicação, através da escolha arbitrária de problemas-exemplo e outras particularidades de cada aplicação.

7.2 Modelos de aproximação utilizados

Antes de entrarmos em detalhes nos resultados das simulações realizadas, precisamos definir os vários tipos de arquiteturas de redes neurais e algoritmos de treinamento supervisionado implementados e utilizados para realizar um estudo comparativo. Dentre os modelos estáticos, temos:

RNTmlpGRn: rede neural tradicional com uma camada intermediária de *n* neurônios com funções de ativação idênticas $f_j(.) = f(.) = tanh(.)$ $(j=1,...,n)$ e parâmetros ajustados via método do gradiente.

RNTmlpGCn: rede neural tradicional com uma camada intermediária de *n* neurônios com funções de ativação idênticas $f_j(.) = f(.) = tanh(.)$ $(j=1,...,n)$ e parâmetros ajustados via método do gradiente conjugado escalonado híbrido (DOS SANTOS & VON ZUBEN, 2000);

RNTmlpFKn: rede neural tradicional com uma camada intermediária de *n* neurônios com funções de ativação idênticas $f_j(.) = f(.) = tanh(.)$ $(j=1,...,n)$ e parâmetros ajustados via *Filtro de Kalman Estendido* (HAYKIN, 1999).

DN_Hermite: dispositivo neurocomputacional gerado pelo algoritmo de aproximação construtivo apresentado no cap. 5 e analisado no cap. 6, com funções de ativação paramétricas na forma de expansão ortonormal truncada, utilizando funções de Hermite como funções básicas, conforme descrito no cap. 5. A ordem máxima das funções de Hermite utilizadas é $P = 6$.

DNP_Spline: dispositivo neurocomputacional gerado pelo algoritmo de aproximação construtivo apresentado no cap. 5 e analisado no cap. 6, com funções de ativação não-paramétricas na forma de *splines* polinomiais suavizantes, conforme descrito no cap. 5.

Dentre os modelos dinâmicos, temos:

RNTlocalGRn: rede neural tradicional com recorrência interna local com uma camada intermediária de *n* neurônios e funções de ativação idênticas $f_j(.) = f(.) = tanh(.)$ $(j=1,...,n)$ com parâmetros ajustados via método do gradiente conjugado escalonado híbrido (DOS SANTOS & VON ZUBEN, 2000);

RNTglobalGRn: rede neural tradicional com recorrência interna global com uma camada intermediária de *n* neurônios e funções de ativação idênticas $f_j(.) = f(.) = tanh(.)$ $(j=1,...,n)$ com parâmetros ajustados via método do gradiente conjugado escalonado híbrido (DOS SANTOS & VON ZUBEN, 2000);

RNTexterGRn: rede neural tradicional com recorrência externa com uma camada intermediária de *n* neurônios e funções de ativação idênticas $f_j(.) = f(.) = tanh(.)$ $(j=1,...,n)$ com parâmetros ajustados via método do gradiente conjugado escalonado híbrido (DOS SANTOS & VON ZUBEN, 2000);

RNTtotalGRn: rede neural tradicional com recorrência total com uma camada intermediária de *n* neurônios e funções de ativação idênticas $f_j(.) = f(.) = tanh(.)$ $(j=1,...,n)$ com parâmetros ajustados via método do gradiente conjugado escalonado híbrido (DOS SANTOS & VON ZUBEN, 2000);

7.3 Aplicações

7.3.1 Aproximação de funções

O processo de aproximação utilizando redes neurais tradicionais e dispositivos neurocomputacionais híbridos foi considerado em cinco problemas de aproximação, com duas variáveis independentes ($N_0 = 2$) e uma variável de saída ($N_s = 1$), dimensões apropriadas para a apresentação de resultados gráficos. Estes exemplos foram também utilizados por HWANG *et al*. (1994) para comparar métodos utilizando redes neurais tradicionais e o método de busca de projeção. VON ZUBEN (1996) também utilizou estes exemplos para comparar modelos paramétricos e não-paramétricos de redes neurais artificiais. Cada exemplo considera para o problema de aproximação uma das cinco funções não-lineares $g^k: [0,1]^2 \rightarrow \Re$, $k = 1,...,5$, apresentadas a seguir e ilustradas nas figuras 7.1 a 7.5:

1) *função de interação simples*:

$$g^{(1)}(x_1, x_2) = 10{,}391[(x_1 - 0{,}4)(x_2 - 0{,}6) + 0{,}36]; \tag{7.1}$$

2) função radial:

$$g^{(2)}(x_1, x_2) = 24{,}234[r^2(0{,}75 - r^2)], \ \ com \ \ r^2 = (x_1 - 0{,}5)^2 + (x_2 - 0{,}5)^2; \tag{7.2}$$

3) função harmônica:

$$g^{(3)}(x_1, x_2) = 42{,}659[0{,}1 + \bar{x}_1(0{,}05 + \bar{x}_1^4 - 10\bar{x}_1^2\bar{x}_2^2 + 5\bar{x}_2^4)], \tag{7.3}$$

com $\bar{x}_{1/2} = x_{1/2} - 0{,}5$;

4) função aditiva:

$$g^{(4)}(x_1, x_2) = 1{,}3356[1{,}5(1 - x_1) + e^{2x_1 - 1}\,\mathrm{sen}(3\pi(x_1 - 0{,}6)^2) + e^{3x_2 - 1{,}5}\,\mathrm{sen}(4\pi(x_2 - 0{,}9)^2)]; \tag{7.4}$$

5) função de interação complicada:

$$g^{(5)}(x_1, x_2) = 1{,}9[1{,}35 + e^{x_1 - x_2}\,sen(13(x_1 - 0{,}6)^2)\,sen(7x_2)]; \tag{7.5}$$

As funções $g^k(x)=g^k(x_1,x_2)$ $(k=1,....,5)$ são supostas desconhecidas, sendo possível apenas obter amostras do valor de cada função para vetores de entrada $\boldsymbol{x}=[x_1\ x_2]^T$ definidos de forma a explorar adequadamente o espaço de aproximação. Seguindo o mesmo procedimento adotado por HWANG *et al.* (1994), para produzir um conjunto de dados de treinamento de dimensão $N=225$, foram gerados vetores de entrada $\boldsymbol{x_l}=[x_{1l},\ x_{2l}]^T$ $(l=1,...,225)$ com base em uma distribuição uniforme no intervalo [0,1].

A matriz de variáveis de entrada $\mathbf{X_{2\times225}}$ é a mesma para os cincos exemplos, as amostras de cada função avaliada em $\boldsymbol{x_l}=[x_{1l},\ x_{2l}]^T$ $(l=1,...,225)$ podem ou não estar sujeitas a ruído de média zero e variância unitária. O nível de aproximação é avaliado através da comparação do valores preditos pelo modelo $\hat{g}$ com os valores de um conjunto de entrada-saída para teste (independente dos dados para treinamento), com dimensão $N_{te}=5000$, onde os vetores de entrada $\boldsymbol{x_l}=[x_{1l},\ x_{2l}]^T$ $(l=1,...,5000)$ são gerados com base em uma distribuição aleatória uniforme na região $[0,1]^2$. A comparação é baseada na fração de variância não-explicada (*FVNE*), dada por:

$$FVNE_k = \frac{\sum_{l=1}^{N_{te}} (\hat{g}^{(k)}(x_{1l}, x_{2l}) - g^{(k)}(x_{1l}, x_{2l}))^2}{\sum_{l=1}^{N_{te}} (g^k(x_{1l}, x_{2l}) - \bar{g}^{(k)})^2}, \ k = 1,....,5, \tag{7.6}$$

onde

$$\bar{g}^{(k)} = \frac{1}{N}\sum_{l=1}^{N_{te}} g^{(k)}(x_{1l}, x_{2l}). \tag{7.7}$$

A fração de variância não-explicada é um critério de avaliação do nível de aproximação que também foi adotado por FRIEDMAN & STUETZLE (1981), ROOSEN & HASTIE (1994), HWANG *et al.* (1994) e VON ZUBEN (1996).

Para cada um dos cinco exemplos, foram escolhidas algumas das formas possíveis de representação de uma rede neural com uma camada intermediária citados na seção anterior, para aproximar as funções $\hat{g}^k : \Re^2 \rightarrow \Re$.

Os resultados de simulação obtidos são apresentados nas tabelas 7.1 e 7.2, onde *n* representa o número de neurônios na camada intermediária. Na tabela 7.2 foi adicionado aos dados amostrados um ruído 0.25ε, com ε uma variável aleatória de distribuição normal, média zero e variância unitária. Foi utilizado o ambiente de simulação computacional MATLAB®.

Tabela 7.1 Desempenho dos modelos de aproximação para dados de amostragem sem ruído

	$g^{(1)}$		$g^{(2)}$		$g^{(3)}$		$g^{(4)}$		$g^{(5)}$	
	n	FVNE	*n*	FVNE	*n*	FVNE	*n*	FVNE	*n*	FVNE
$RNT_{mlp}GR5$	5	0.00906	5	0.15021	5	0.95018	5	0.08978	5	0.45901
$RNT_{mlp}GR15$	15	0.00398	15	0.09386	15	0.48213	15	0.01219	15	0.21092
$RNT_{mlp}GC5$	5	0.00137	5	0.00378	5	0.27425	5	0.01927	5	0.16573
$RNT_{mlp}GC15$	15	0.00061	15	0.00045	15	0.03748	15	0.00068	15	0.09678
DN_Hermite	2	0.00000	3	0.00008	5	0.00009	2	0.00096	5	0.00948
DN_Spline	2	0.00000	3	0.00243	5	0.00379	2	0.00345	5	0.00549

Tabela 7.2 Desempenho dos modelos de aproximação para dados de amostragem com ruído

	$g^{(1)}$		$g^{(2)}$		$g^{(3)}$		$g^{(4)}$		$g^{(5)}$	
	n	FVNE	*n*	FVNE	*n*	FVNE	*n*	FVNE	*n*	FVNE
$RNT_{mlp}GR5$	5	0.05679	5	0.28595	5	1.01178	5	0.09677	5	0.50781
$RNT_{mlp}GR15$	15	0.01245	15	0.15679	15	0.54676	15	0.02454	15	0.27985
$RNT_{mlp}GC5$	5	0.00435	5	0.01499	5	0.32457	5	0.03972	5	0.20977
$RNT_{rmlp}GC15$	15	0.00115	15	0.00987	15	0.45678	15	0.01193	15	0.10989
DN_Hermite	2	0.00031	3	0.00314	5	0.00678	2	0.01445	5	0.02388
DN_Spline	2	0.00023	3	0.00146	5	0.03498	2	0.00512	5	0.03077

Comparando-se o modelos *RNTmlpGC5*, *RNTmlpGC15* (funções de ativação previamente definidas e idênticas) com os modelos *DN_Hermite* e *DN_Spline* (funções de ativação definidas a partir dos dados de aproximação disponíveis), verifica-se que a fixação prévia da função de ativação acaba por exigir um número bem superior de neurônios para obter níveis de aproximação equivalentes, conduzindo a um modelo de aproximação que demanda um custo computacional bem maior para ser operado.

Vale mencionar que existe uma limitação de poder de aproximação apresentado por expansão ortonormal truncada *DN_Hermite* quando comparada com a aproximação polinomial fornecida por *splines* polinomiais suavizantes, *DN_Spline*. Devido à baixa dimensionalidade e pela simplicidade da tarefa de aproximação, estas limitações não se tornaram evidentes. Como veremos nas seções seguintes, tais limitações irão manifestar-se quando estivermos tratando com sistemas dinâmicos. No entanto, quando estivermos empregando estratégia de controle *on-line* devemos ser cautelosos na utilização de *splines,* visto que, apesar de possuírem um alto poder de aproximação, é uma técnica muito sensível aos parâmetros de treinamento arbitrados pelo usuário, como por exemplo, taxa de aprendizado, a qual pode provocar instabilidade no sistema, caso não seja definida adequadamente. Um dos fatores que pode explicar a limitação do poder de representação do modelo *DN_Hermite* quando comparada ao modelo *DN_Spline,* é que a suavidade do modelo *DN_Spline* é definida automaticamente e continuamente a partir dos dados, via validação cruzada, enquanto que o modelo *DN_Hermite* só admite ajuste, também via

validação cruzada, no número *P* (ordem dos polinômios de Hermite) das funções básicas a serem utilizadas na definição de cada função de ativação. Valores elevados de *P* irão produzir erros baixos no treinamento, mas tendem a conduzir a um erro elevado de generalização.

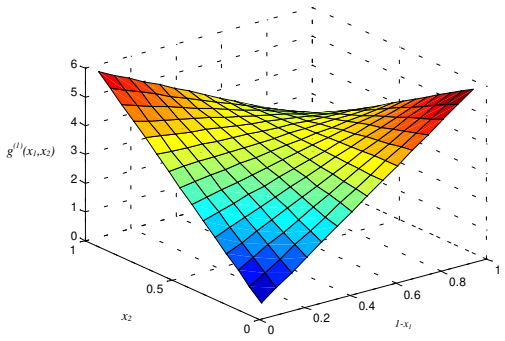


Figura 7.1 - Gráfico de $g^{(1)}(x_1,x_2)$

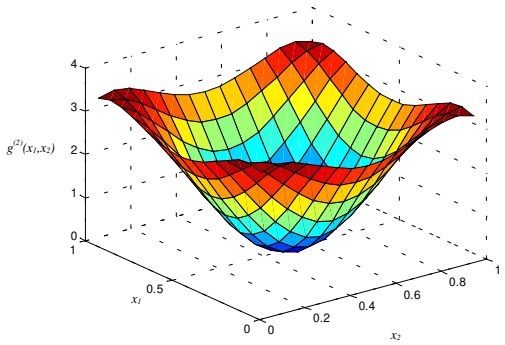


Figura 7.2 - Gráfico de $g^{(2)}(x_1,x_2)$

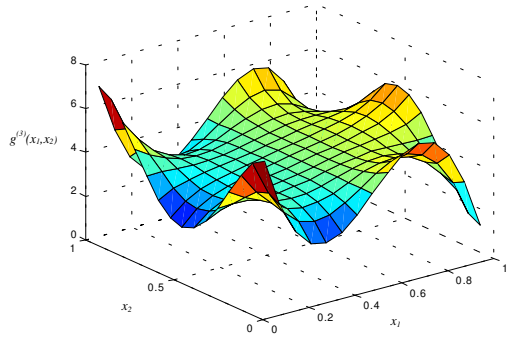


Figura 7.3 - Gráfico de $g^{(3)}(x_1,x_2)$

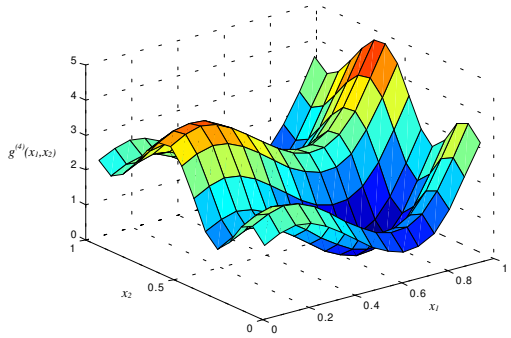


Figura 7.4 - Gráfico de $g^{(4)}(x_1,x_2)$

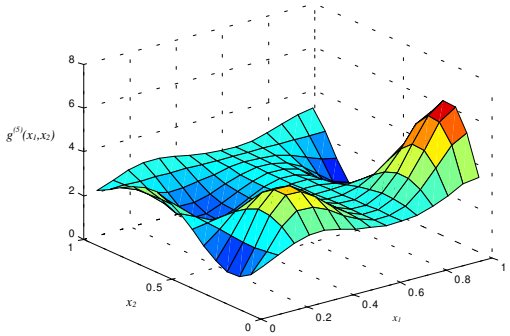


Figura 7.5 - Gráfico de $g^{(5)}(x_1,x_2)$

7.3.2 Predição de séries temporais

Dadas as observações dos estados de um sistema dinâmico em instantes de tempo inferiores ou iguais a t descrevendo uma série temporal, o problema de predição consiste em usar os dados observados para prever com a maior precisão possível $y(t+p)$, onde y representa o estado do sistema dinâmico e p é o passo de predição no futuro.

Grande parte dos processos físicos existentes na natureza são sistemas dinâmicos que, em algum grau, apresentam características não-lineares. Assim as séries temporais originadas por estes sistemas apresentam componentes e inter-relações não-lineares que dificilmente podem ser representados por modelos lineares dentro de limites severos de precisão. Em resposta às limitações inerentes aos modelos lineares, as redes neurais artificiais têm sido aplicadas com sucesso na geração de modelos não-lineares para predição de séries temporais.

A título de ilustração e verificação da adequabilidade de dispositivos neurocomputacionais híbridos em predição de um passo, é utilizada a série temporal *SUN*. Os valores desta série são indicativos da media relativa de manchas solares observadas a cada ano desde 1700 até 1979 (MCDONNELL & WAGEN, 1994), conforme ilustrado na figura 7.6. O conjunto de dados foi particionado em um conjunto de treinamento sobre os anos 1700-1949, e o conjunto de teste corresponde aos anos 1950-1979.

Como modelos de predição alternativo, são utilizados, além dos dispositivos neurocomputacionais híbridos, redes neurais estáticas com atrasos na entrada e redes neurais recorrentes. Todos os modelos tiveram seus parâmetros ajustados recursivamente sobre 250 pontos (1700-1949). Após o treinamento os modelos foram testados, realizando a predição de um passo do conjunto de 30 pontos (1950-1979). O único pré-tratamento realizado sobre a série consistiu em normalizá-la para assumir valores no intervalo [0,1].

Modelos de predição a serem empregados aqui foram devidamente testados para se definir dimensões adequadas à complexidade do problema.

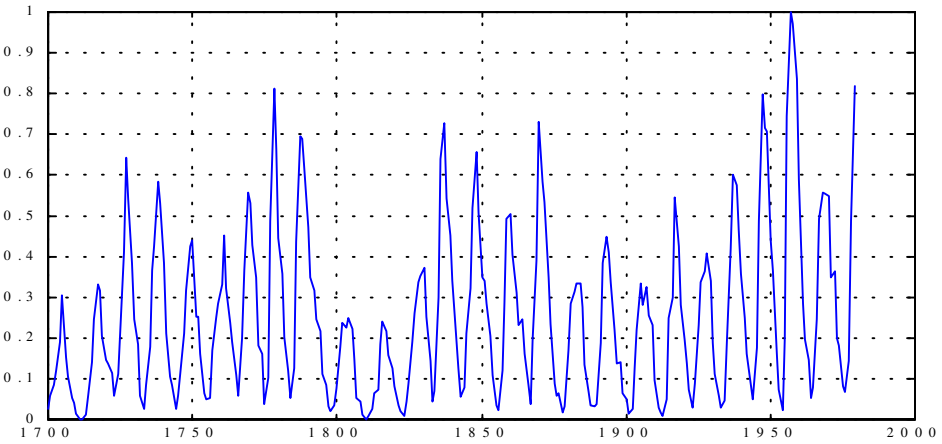


Figura 7.6 – Gráfico da Série Sun

Tabela 7.3 Desempenho dos modelos de predição para amostragem sem ruído

	Série SUN		
	nº de neurônios	Erro Quadrático Médio	
		Treinamento	Predição
$RNT_{local}GR5$	5	0.00351	0.04363
$RNT_{local}GR10$	10	0.00328	0.04253
$RNT_{global}GR5$	5	0.00262	0.03280
$RNT_{global}GR10$	10	0.00249	0.03234
$RNT_{exter}GR5$	5	0.00427	0.02070
$RNT_{exter}GR10$	10	0.00422	0.01753
$RNT_{total}GR5$	5	0.00389	0.01160
$RNT_{total}GR10$	10	0.00307	0.01534
DN_Hermite	8	0.00253	0.01824
DN_Spline	7	0.00239	0.01905

Para os melhores resultados apresentados na tabela 7.3, referentes a cada um dos modelos utilizados, apresentamos a seguir o resultado da predição.

Rede neural com recorrência local com 10 neurônios na camada intermediária e 2 atrasos

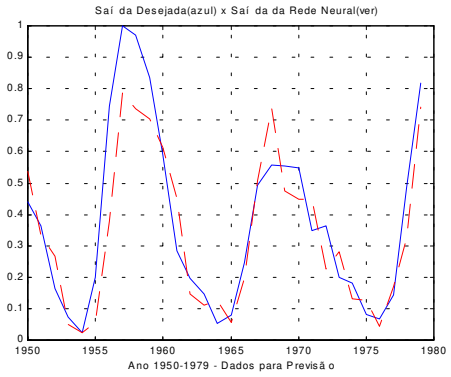


Figura 7.7 – Predição realizada (série temporal (azul) × saída da rede neural (vermelha))

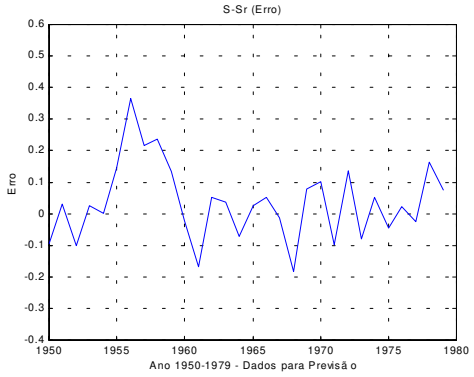


Figura 7.8 - Erro de predição (série temporal - saída da rede neural)

Rede neural com recorrência global com 10 neurônios na camada intermediária e 2 atrasos

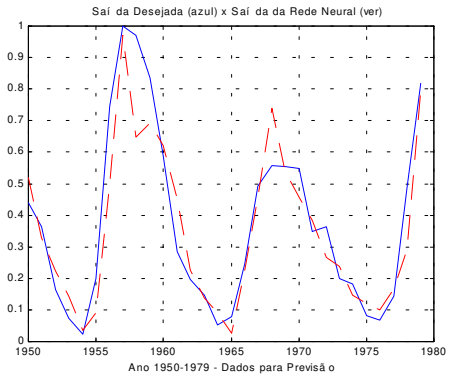


Figura 7.9 - Predição realizada (série temporal (azul) × saída da rede neural (vermelha))

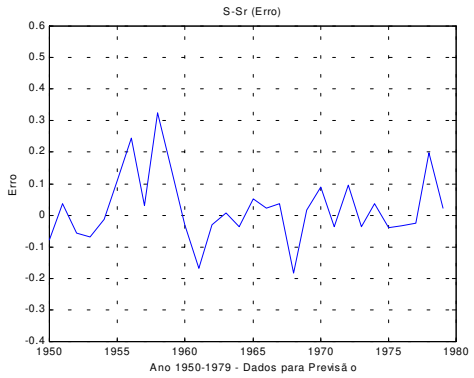


Figura 7.10 - Erro de predição (série temporal - saída da rede neural)

Rede neural com recorrência externa com 10 neurônios na camada intermediária e 2 atrasos

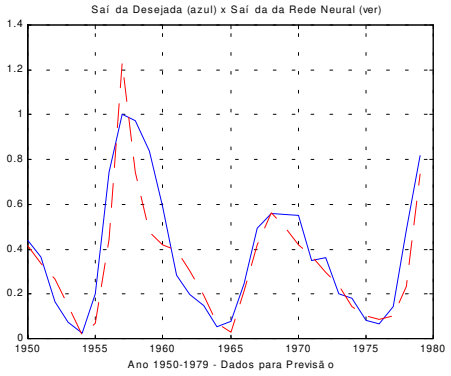


Figura 7.11 - Predição realizada (série temporal (azul) × saída da rede neural (vermelha))

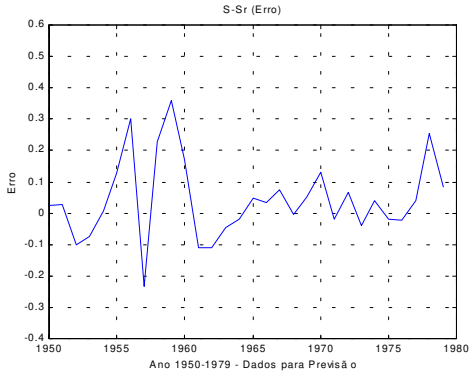


Figura 7.12 - Erro de predição (série temporal - saída da rede neural)

Rede neural com recorrência total com 5 neurônios na camada intermediária

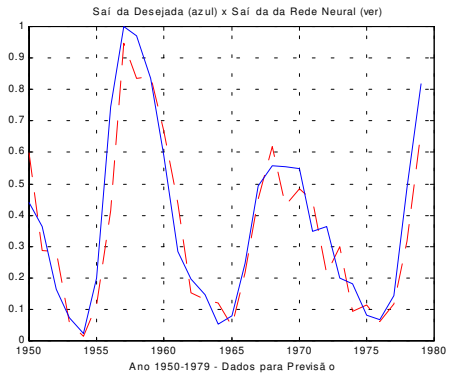


Figura 7.13 - Predição realizada (série temporal (azul) × saída da rede neural (vermelha))

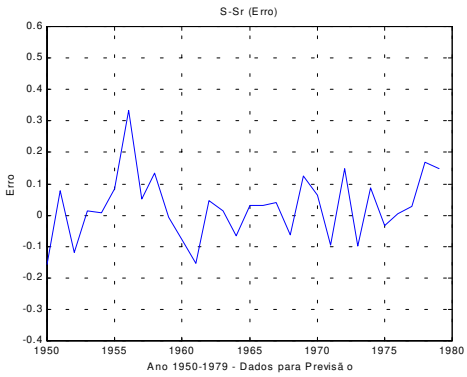


Figura 7.14 - Erro de predição (série temporal - saída da rede neural)

Dispositivo neurocomputacional (*DN_Hermite*)

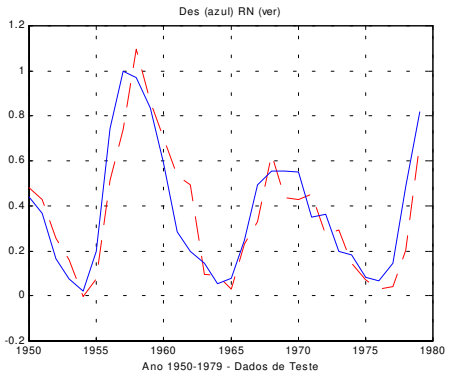


Figura 7.15 - Predição realizada (série temporal (azul) × saída do DN_Hermite (vermelha))

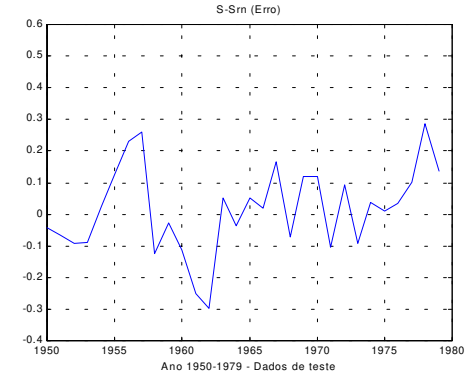


Figura 7.16 - Erro de predição (série temporal - saída do DN_Hermite)

Comparando os resultados apresentados na tabela 7.3, observamos que os dispositivos neurocomputacionais alcançaram bons resultados quando comparados com outras abordagens, como redes recorrentes. Somente redes neurais com recorrência global conseguiram obter resultados equivalentes, na fase de treinamento, aos dispositivos neurocomputacionais com uma arquitetura menor. Na fase de testes, a rede neural com recorrência total superou todas as outras abordagens. Vale salientar que o algoritmo empregado no treinamento das redes neurais recorrentes correspondente a um dos algoritmos com melhor desempenho dentre os encontrados na literatura. Com base nos ótimos resultados alcançados pelos dispositivos neurocomputacionais para predição de séries temporais quando comparados com outras estratégias, como *MLP* e mesmo redes

recorrentes, podemos esperar um bom desempenho desta estratégia para problemas de identificação e controle, conforme será apresentado nas seções seguintes.

7.3.3 Identificação de sistemas dinâmicos

Nesta seção, são apresentados resultados de identificação de plantas não-lineares usando os modelos de identificação sugeridos anteriormente, no capítulo 3. Três exemplos (casos 1, 2 e 3) são apresentados e os modelos utilizados para identificar as plantas correspondem ao modelo IV, descrito no capítulo 3. Cada planta é escolhida para enfatizar um aspecto específico. Na primeira planta, a saída da planta é uma função linear da entrada *u,* ao passo que na segunda planta este relacionamento é não-linear, já no caso da terceira, o intuito é demonstrar a viabilidade dos dispositivos neurocomputacionais híbridos para problemas com múltiplas entradas e múltiplas saídas (*MIMO*).

Caso 1

Este exemplo foi também utilizado por NARENDRA & PARTHASARATHY (1990), na qual a planta a ser identificada é dada por uma equação a diferenças de segunda ordem altamente não-linear. NARENDRA & PARTHASARATHY (1990) apresentam uma discussão mais detalhada, como por exemplo, os pontos de equilíbrio do sistema não forçado, isto é, com $u(k)=0$ $\forall$k.

$$y(k+1) = \frac{y(k)\cdot y(k-1)\cdot[y(k)+2.5]}{1+y^2(k)+y^2(k-1)} + u(k) \tag{7.8}$$

O conjunto de treinamento é composto de 500 pontos gerados a partir da equação (7.8) que descreve a planta, supondo um sinal de entrada aleatório $u(k)$ e uniformemente distribuído no intervalo [−2,+2]. Estes dados são utilizados para construir um modelo para esta planta. Para o caso das *RNTmlpFK* o conjunto de treinamento é composto de 10000 pontos com os parâmetros ajustados *on-line*. Foram realizados vários testes com conjunto de treinamento menor, mas os resultados foram muito ruins. NARENDRA & PARTHASARATHY (1990) utilizaram um conjunto de treinamento com 100.000 pontos onde os pesos da rede neural eram ajustados usando o método do gradiente a cada 5 pontos com

um passo de 0,25. O modelo utilizado para identificar será o modelo IV (capítulo 4), descrito pela equação a diferenças a seguir.

$$y(k+1) = N[u(k), y(k), y(k-1)] \tag{7.9}$$

onde N é uma rede neural estática (rede neural tradicional ou dispositivo neurocomputacional híbrido). O modelo tem três entradas $u(k)$, $y(k)$, $y(k-1)$ e uma saída $y(k+1)$, conforme apresentada na figura 7.17. Após concluído o treinamento, o modelo é testado aplicando um sinal de entrada senoidal $u(k) = sin\left(\frac{2k\pi}{25}\right)$, cujo conjunto de teste corresponde a 120 pontos. Um segundo teste é realizado usando um sinal de entrada $u(k) = 1{,}6 \cdot \cos\left(\frac{2k\pi}{30}\right)$, composto de 180 pontos. A saída desejada e do modelo de identificação para os dois conjuntos de teste são apresentadas na figura 7.18 a 7.27. O resultado é mostrado na tabela 7.4.

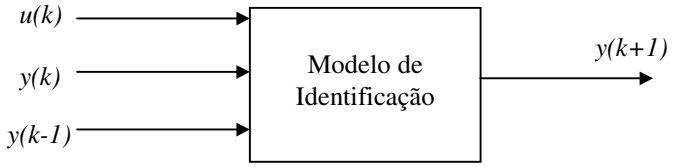


Figura 7.17 - Estrutura do modelo de identificação

Tabela 7.4 Identificação da inversa direta (veja seção do capitulo 3 - modelo estático com linhas de atrasos na entrada)

	Planta 1				
	N.º Neurônios	Critério de parada(épocas ou EQM)	Erro Quadrático Médio		
			Treinamento	Teste 1	Teste 2
RNTmlpGC	6	2000 épocas	0.00437	0.00471	0.00621
RNTmlpGC	10	2000 épocas	0.00184	0.00237	0.00290
RNTmlpGC	15	193/0.0014	0.00139	0.00179	0.00713
RNTPmlpFK	6	----------	----------	0.02740	0.07050
RNTmlpFK	10	----------	----------	0.01792	0.05090
RNTmlpFK	15	----------	----------	0.016966	0.04111
DN_Hermite	6	0.0014	0.00140	0.00123	0.00183
DN_Spline	6	0.0014	0.00130	0.00106	0.00127

Foram adotados dois critérios de parada para treinamento de redes neurais tradicionais. O primeiro diz respeito ao número máximo de épocas, que neste exemplo, foi adotado 2000 épocas. O segundo diz respeito ao erro quadrático médio no treinamento, o qual foi definido como sendo o erro atingido pelo dispositivo neurocomputacional híbrido para o mesmo problema. Caso algum destes critérios seja atingido, o treinamento é finalizado. Quando o treinamento é finalizado antes de completar o número máximo de épocas, tanto o número de épocas quanto o erro são apresentados na tabela 7.4, na segunda coluna.

Os resultados apresentados na tabela 7.4 para redes neurais tradicionais correspondem ao melhor resultado alcançado, para o conjunto de teste, em 10 tentativas realizadas, variando a condição inicial dos pesos. Uma análise dos resultados apresentados na tabela 7.4, mostra que os modelos baseados em dispositivo neurocomputacional híbrido conseguiram fornecer uma solução com menor dimensão e melhor erro de generalização nos dois conjuntos de testes apresentados. Apesar da *RNTmlp15* ter obtido menor erro no conjunto de treinamento, o mesmo não foi verificado no conjunto de teste.

Os gráficos para a abordagem *DN_Hermite* são apresentados a seguir:

- Primeiro conjunto de teste

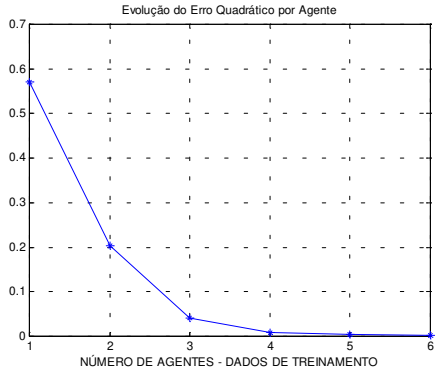


Figura 7.18 - Evolução do erro quadrático durante o processo de construção do dispositivo neurocomputacional híbrido

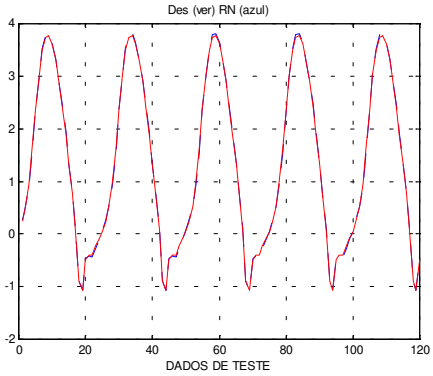


Figura 7.19 - Saída do *DN_Hermite* (RN: azul) × saída da planta (DES: vermelha)

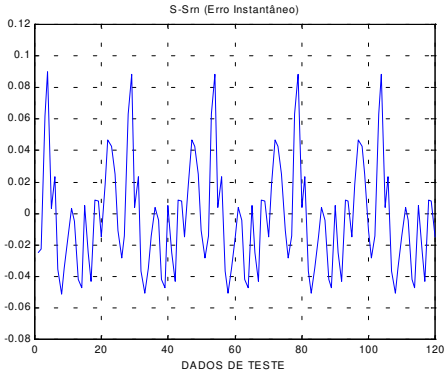


Figura 7.20 - Erro instantâneo: Saída da planta (S) – saída do *DN_Hermite* (Srn)

- Segundo conjunto de teste

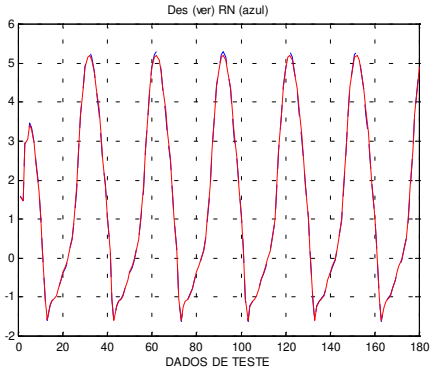


Figura 7.21 - Saída do *DN_Hermite* (RN: azul) × saída da planta (DES: vermelha)

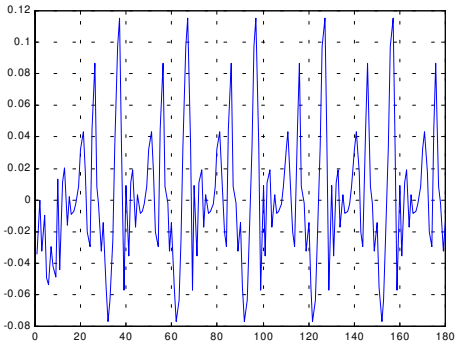


Figura 7.22 - Erro instantâneo: Saída da planta (S) – saída do *DN_Hermite* (Srn)

Os gráficos para a abordagem *DN_Spline* são apresentados abaixo:

- Primeiro conjunto de teste

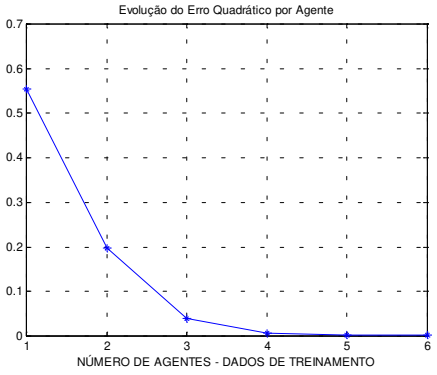


Figura 7.23 - Evolução do erro quadrático durante o processo de construção do dispositivo neurocomputacional híbrido

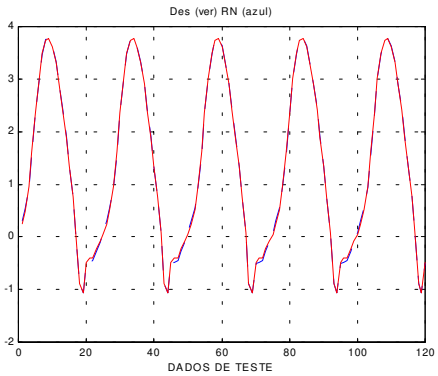


Figura 7.24 - Saída do *DN_Spline* (RN: azul) × saída da planta (DES: vermelha)

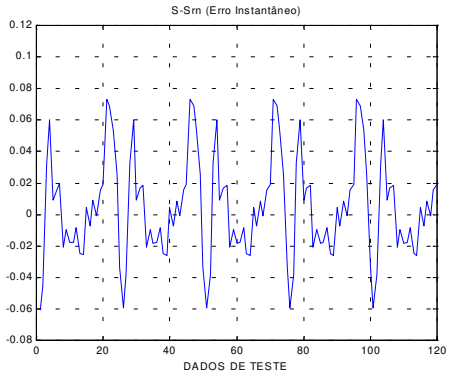


Figura 7.25 - Erro instantâneo: Saída da planta (S) – saída do *DN_Spline* (Srn)

- Segundo conjunto de teste

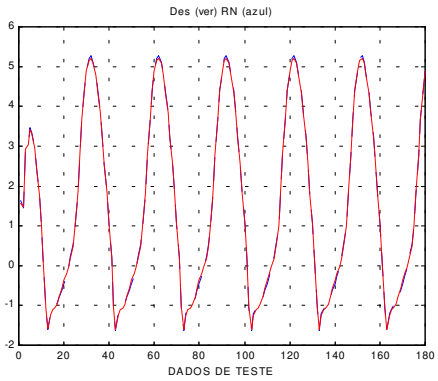


Figura 7.26 - Saída do *DN_Spline* (RN: azul) × saída da planta (DES: vermelha)

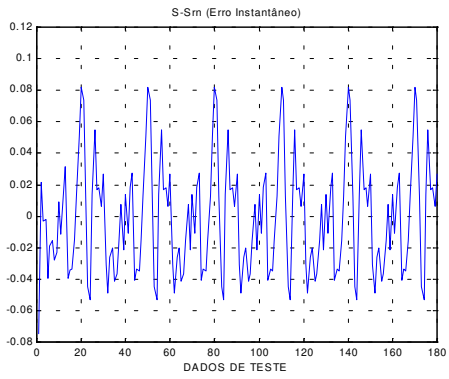


Figura 7.27 - Erro instantâneo: Saída da planta (S) – saída do *DN_Spline* (Srn)

Analisando as figuras 7.19, 7.21, 7.24 e 7.26, observamos que tanto a saída do *DN_Hermite* quanto do *DN_Spline* aproximam a saída da planta, para o primeiro e o segundo conjunto de testes, com um erro razoavelmente baixo. Isto demonstra que os dispositivos neurocomputacionais possuem um alto poder de representação e generalização quando comparados as técnicas tradicionais.

Nas figuras 7.18 e 7.23 pode-se observar duas características marcantes encontradas nos dispositivos neurocomputacionais, que são as seguintes:

- à medida que novos neurônios (neuro-agentes) são adicionados à rede, o erro quadrático sempre diminui, no entanto, a contribuição individual de cada neuro-agente pode variar, devido ao processo de competição ao qual são submetidos.

- a diminuição do erro quadrático pela adição de novos neurônios segue um decaimento exponencial;

Caso 2

Este exemplo foi utilizado por LIU *et al.* (1998b) e NARENDRA & PARTHASARATHY (1990), no qual a planta a ser identificada é dada por uma equação a diferenças de terceira ordem, altamente não-linear. Nesta planta, tanto a entrada de controle como os valores atrasados da saída da planta combinam-se de forma não-linear.

$$y(k+1) = \frac{y(k)\cdot y(k-1)\cdot y(k-2)\cdot u(k-1)[y(k-2)-1]+u(k)}{1+y^2(k-2)+y^2(k-1)} \qquad (7.10)$$

No conjunto de treinamento, a entrada *u* consiste de uma seqüência randômica uniformemente distribuída no intervalo [-1,+1], a qual é composta de 1000 pontos gerados a partir da equação (7.10). Estes dados são utilizados para construir um modelo para esta planta. Para o modelo *RNTmlpFK* o conjunto de treinamento é composto de 10000 pontos com os parâmetros ajustados *on-line*. Após concluído o treinamento, o modelo é testado aplicando um sinal de entrada senoidal *u* a planta, composto de 1000 pontos gerados na forma:

$$u(k) = \begin{cases} sin\left(\dfrac{2k\pi}{250}\right) & 0 \le k \le 500 \\ 0.8\cdot sin\left(\dfrac{2k\pi}{250}\right) + 0.2\cdot sin\left(\dfrac{2k\pi}{25}\right) & 500 < k \le 1000 \end{cases} \qquad (7.11)$$

O modelo utilizado para identificação será o modelo IV (capítulo 3), descrito pela equação diferenças abaixo.

$$y(k+1) = N[u(k), u(k-1), y(k), y(k-1), y(k-2)] \qquad (7.12)$$

onde *N* é uma rede neural estática tradicional ou um dispositivo neurocomputacional híbrido. Conforme apresentado na figura 7.28, o modelo tem cinco entradas *u(k), u(k-1)*, *y(k)*, *y(k-1), y(k-2)* e uma saída *y(k+1).*

Figura 7.28 - Estrutura do modelo identificação

Tabela 7.5 Identificação da inversa direta (modelo estático com linha de atrasos na entrada)

	Planta 2			
	N.º Neurônios	Critério de parada(épocas ou EQM)	Erro Quadrático Médio	
			Treinamento	Teste
RNTmlpGC6	6	2000 épocas	0.000678	0.012031
RNTmlpGC10	10	522 épocas	0.000489	0.005785
RNTmlpGC15	15	386 épocas	0.000495	0.008380
RNTmlpFK6	6	----------	----------	0.012962
RNTmlpFK10	10	----------	----------	0.009888
RNTmlpFK15	15	----------	----------	0.007957
DN_Hermite	6	0.0005	0.000496	0.000126
DN_Spline	6	0.0005	0.000460	0.000818

Os resultados apresentados na tabela 7.5 mostram a superioridade dos dispositivos neurocomputacionais híbridos quando comparados às rede neurais tradicionais. Apesar das redes neurais tradicionais conseguirem níveis de erro de treinamento compatíveis, isto não se repete no conjunto de teste, mesmo para arquiteturas de elevada dimensão, como o caso de *RNTmlpGC15*.

Ao contrário, do exemplo anterior, onde *DN_Spline* conseguiu obter menor erro no conjunto de treinamento e teste, quando comparado ao *DN_Hermite*, para esta planta *DN_Hermite* conseguiu melhores resultados. Isto mostra que, dependendo da planta, os dispositivos neurocomputacionais híbridos utilizando polinômios de Hermite podem ser superiores a *splines*, apesar destes possuírem maior poder de representação.

Os gráficos para a abordagem *DN_Hermite* são apresentados a seguir:

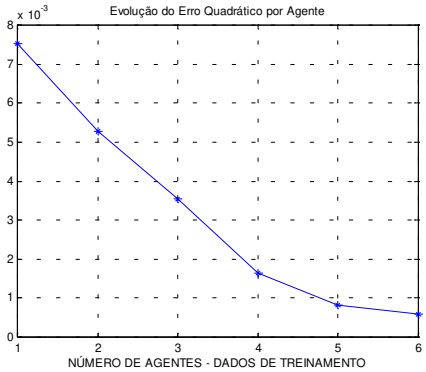


Figura 7.29 - Evolução do erro quadrático durante o processo de construção do dispositivo neurocomputacional híbrido

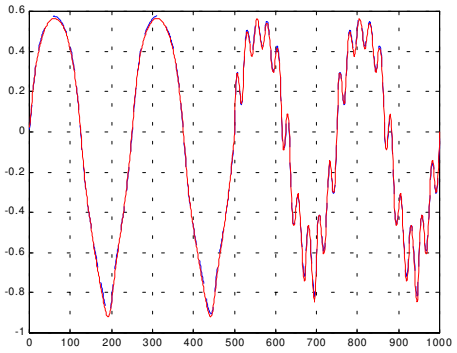


Figura 7.30 - Saída do *DN_Hermite* (RN: azul) × saída da planta (DES: vermelha)

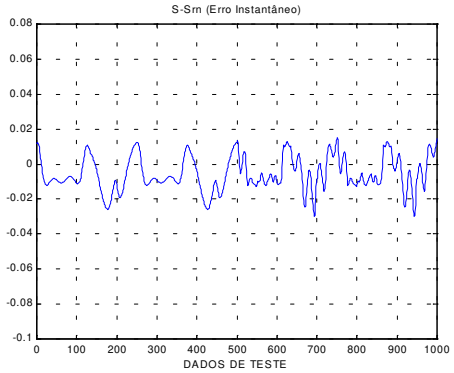


Figura 7.31 - Erro instantâneo: Saída da planta (S) – saída do *DN_Hermite* (Srn)

Os gráficos para o *DN_Spline* são apresentados abaixo:

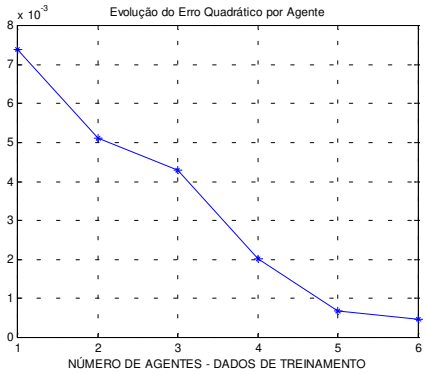


Figura 7.32 - Evolução do erro quadrático durante o processo de construção do dispositivo neurocomputacional híbrido

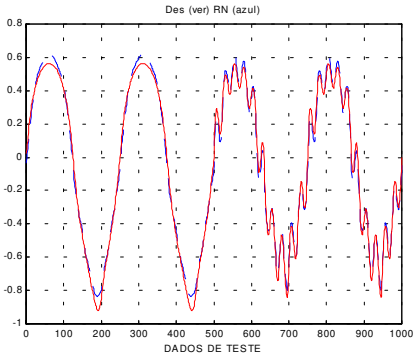


Figura 7.33 - Saída do *DN_Spline* (RN: azul) × saída da planta (DES: vermelha)

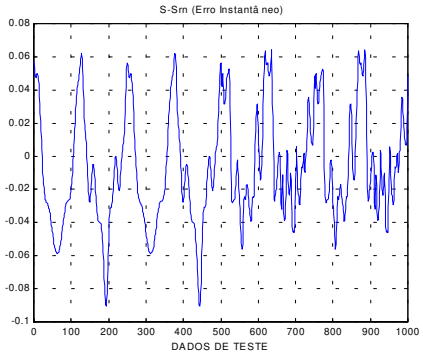


Figura 7.34 - Erro instantâneo: Saída da planta (S) – saída do *DN_Spline* (Srn)

Caso 3

Este exemplo foi utilizado por NARENDRA & PARTHASARATHY (1990), no qual a planta a ser identificada é dada por uma equação a diferenças de primeira ordem não-linear multivariável descrita pela equação abaixo:

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} \dfrac{y_1(k)}{1+y_2^2(k)} \\ \dfrac{y_1(k)y_2(k)}{1+y_2^2(k)} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} \tag{7.13}$$

Este exemplo tem como objetivo mostrar que os dispositivos neurocomputacionais híbridos, assim como outras ferramentas usadas para identificar plantas SISO, podem ser usados para identificar plantas MIMO. O procedimento de identificação foi o mesmo adotado por NARENDRA & PARTHASARATHY (1990), isto é, existirá um modelo de identificação para cada saída da planta. Desta forma, o modelo de identificação para a planta descrita pela equação 7.13, apresenta duas redes neurais, N_1 e N_2, e é descrito pela equação abaixo.

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} N_1[u_1(k), y_1(k), y_2(k)] \\ N_2[u_2(k), y_1(k), y_2(k)] \end{bmatrix} \tag{7.14}$$

onde N_1 e N_2 podem ser redes neurais estáticas tradicionais ou dispositivos neurocomputacionais híbridos. O conjunto de treinamento é composto de 1000 pontos gerados a partir da equação (7.13), aplicando entradas randômicas $u_1(k)$ e $u_2(k)$, independentes e uniformemente distribuídas no intervalo [-1,+1]. A resposta da planta e do modelo de identificação foi testada para o vetor de entradas dado na forma:

$$\begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} = \begin{bmatrix} sin\left(\frac{2\pi k}{25}\right) \\ \cos\left(\frac{2\pi k}{25}\right) \end{bmatrix} \tag{7.15}$$

Cada modelo, conforme apresentado nas figuras 7.35 e 7.36, tem três entradas $u_1(k)$, $y_{p1}(k)$, $y_{p2}(k)$, e $u_2(k)$, $y_{p1}(k)$,$y_{p2}(k)$ e uma saída $y_{p1}(k+1)$ e $y_{p2}(k+1)$ respectivamente.

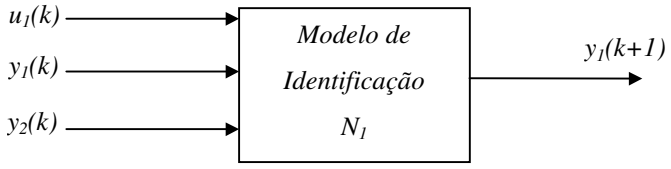


Figura 7.35 - Estrutura do modelo de identificação do modelo N_1

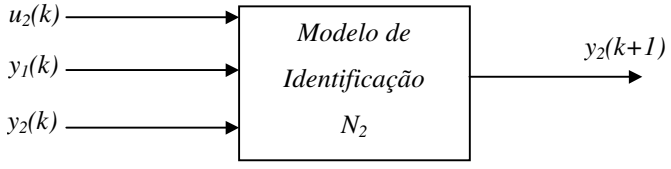


Figura 7.36 - Estrutura do modelo de identificação do modelo N_2

Tabela 7.6 - Identificação da inversa direta para a rede neural N_1 (modelo estático com linha de atrasos na entrada)

	Planta 3			
	N.º Neurônios	Critério de parada(épocas ou EQM)	Erro Quadrático Médio	
			Treinamento	Teste
RNTmlpGC3	3	2000 épocas	0.00012	0.01095
RNTmlpGC5	5	2000 épocas	0.00010	0.00684
RNTmlpFK3	3	----------	----------	0.74413
RNTmlpFK5	5	----------	----------	0.64235
DN_Hermite	3	0.0001	0.00007	0.00658
DN_Spline	3	0.0001	0.00009	0.00713

Os resultados apresentados demonstram que os dispositivos neurocomputacionais híbridos foram superiores em relação às rede tradicionais, inclusive no quesito parcimônia, já que a dimensão da rede neural resultante (3 neurônios) é pouco menor que a obtida com o melhor modelo tradicional (5 neurônios). A justificativa para tal resultado deve-se ao fato de que o modelo da planta adotada para a saída $y_{p1}(k)$ é muito simples, não exigindo grande flexibilidade do modelo de aproximação.

Os gráficos para a abordagem *DN_Hermite* são apresentados abaixo:

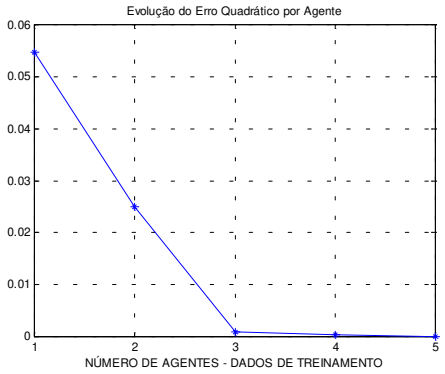


Figura 7.37 - Evolução do erro quadrático durante o processo de construção do dispositivo neurocomputacional híbrido

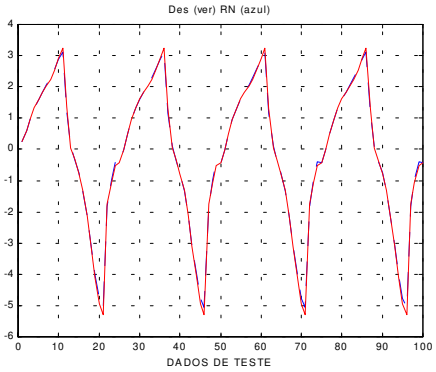


Figura 7.38 - Saída do *DN_Hermite* (RN: azul) × saída da planta (DES: vermelha)

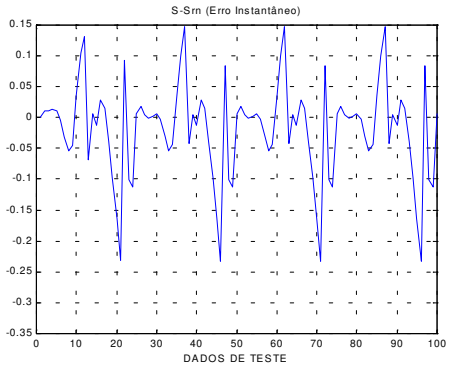


Figura 7.39 - Erro instantâneo: Saída da planta (S) – saída do *DN_Hermite* (Srn)

Os gráficos para a abordagem *DN_Spline* são apresentados abaixo:

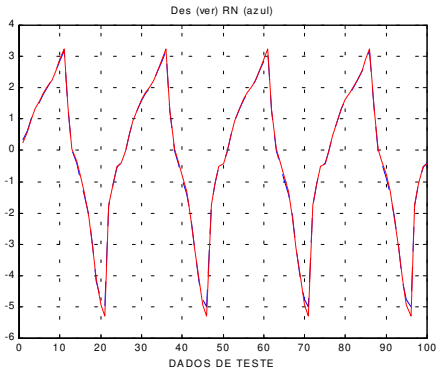


Figura 7.40 - Saída do *DN_Spline* (RN: azul) × saída da planta (DES: vermelha)

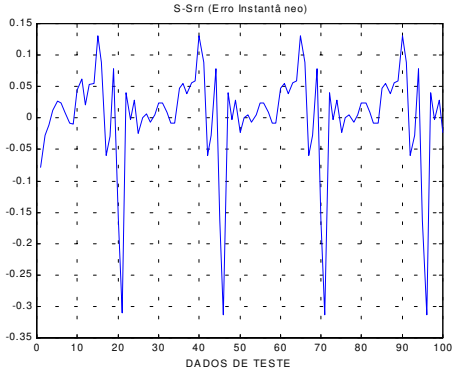


Figura 7.41 - Erro instantâneo: Saída da planta (S) – saída do *DN_Spline* (Srn)

Os gráficos para a abordagem *RNTmlpGC5* são apresentados abaixo:

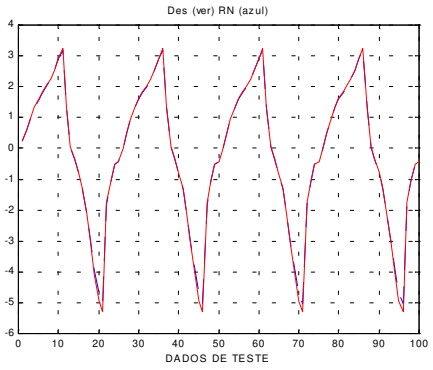


Figura 7.42 - Saída da rede neural (RN: azul) × saída da planta (DES: vermelha)

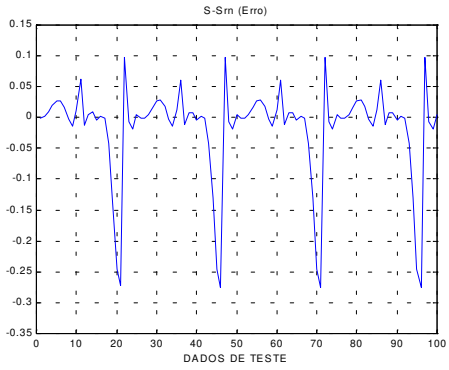


Figura 7.43 - Erro instantâneo: Saída da planta (S) – saída da rede neural (Srn)

Tabela 7.4 – Identificação da inversa direta para a rede neural N_2 (modelo estático com linhas de atraso na entrada)

	Planta 3			
	N.º Neurônios	Critério de parada(épocas ou EQM)	Erro Quadrático Médio	
			Treinamento	Teste
RNTmlp5	5	2000	0.00039	0.03624
RNTmlp10	10	1530/0.0001	0.00001	0.01839
RNTmlp15	15	1360/0.0001	0.00000	0.00732
RNTmlp5FK	5	----------	----------	0.63072
RNTmlp10FK	10	----------	----------	0.63131
RNTmlp15FK	15	----------	----------	0.617913
DN_Hermite	3	0.0001	0.00001	0.00604
DN_Spline	3	0.0001	0.00007	0.00797

Ao contrário do resultado obtido para a saída $y_{p1}(k)$, no caso da saída $y_{p2}(k)$ houve um ganho significativo na dimensão da rede, isto é, para alcançar o mesmo erro de generalização foi necessário um rede três vezes maior que aquela obtida pelos dispositivos neurocomputacionais híbridos. Isto pode ser justificado pelo fato de que a saída $y_{p2}(k)$ é mais complexa que a saída $y_{p1}(k)$.

Os gráficos para a abordagem *DN_Hermite* são apresentados a seguir:

Figura 7.44 - Evolução do erro quadrático durante o processo de construção do dispositivo neurocomputacional híbrido

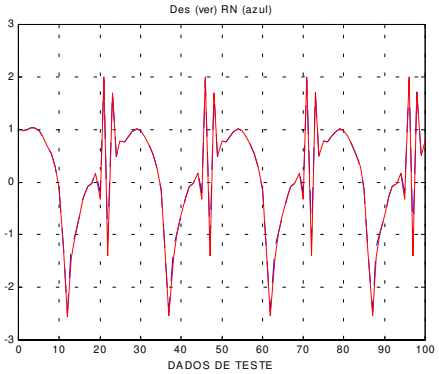


Figura 7.45 - Saída do *DN_Hermite* (RN: azul) × saída da planta (DES: vermelha)

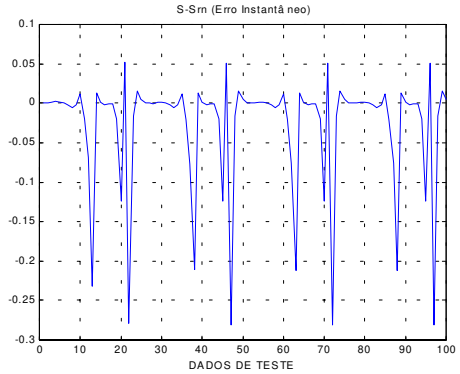


Figura 7.46 - Erro instantâneo: Saída da planta (S) – saída do *DN_Hermite* (Srn)

Os gráficos para a abordagem *DN_Spline* são apresentados a seguir:

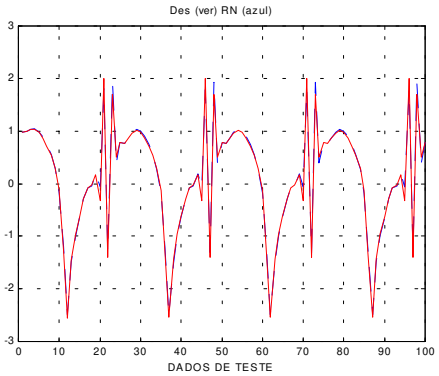


Figura 7.47 - Saída do *DN_Spline* (RN: azul) × saída da planta (DES: vermelha)

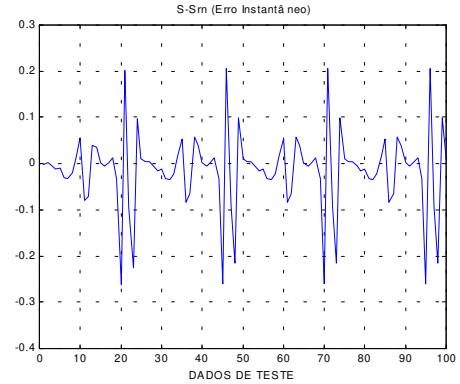


Figura 7.48 - Erro instantâneo: Saída da planta (S) – saída do *DN_Spline* (Srn)

Os gráficos para a abordagem *RNTmlp15* são apresentados abaixo:

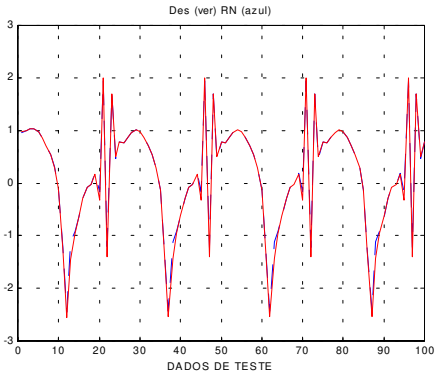


Figura 7.49 - Saída da rede neural (RN: azul) × saída da planta (DES: vermelha)

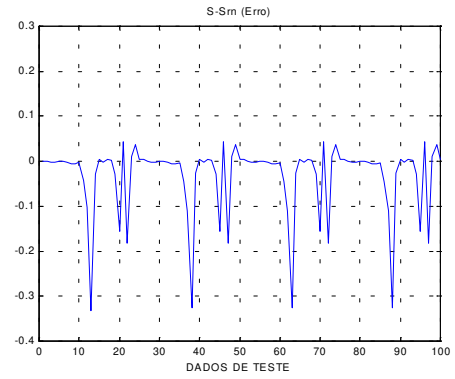


Figura 7.50 - Erro instantâneo: Saída da planta (S) – saída da rede neural (Srn)

7.3.4 Controle e identificação de sistemas

Nesta seção, os procedimentos de simulação adotados no caso de dispositivos neurocomputacionais híbridos são discutidos brevemente.

7.3.4.1 Primeiro estágio - Identificação

Neste estágio os dados de entrada-saída da planta são necessários para construir um modelo de identificação, como descrito no capítulo 3. Este modelo é obtido pela aplicação de entradas randômicas *u(k)* escolhidas convenientemente num intervalo de interesse. Este intervalo deve ser tal que todos os modos da planta possam ser excitados. Vale salientar que a escolha adequada deste intervalo é de vital importância para que o modelo identificado apresente um bom desempenho na fase de teste e no terceiro estágio, para o cálculo do jacobiano da planta, necessário para atualização dos parâmetros dos controlador.

7.3.4.2 Segundo estágio - Projeto off-line do controlador

Para ajuste *on-line,* os valores iniciais dos parâmetros do(s) controlador(es) são críticos para assegurar estabilidade do sistema. Estes valores são determinados *off-line* (dimensão da rede neural ou funções de ativação) usando a técnica de controle direto. Um esquema adotado é apresentado na figura 7.51.

7.3.4.3 Terceiro estágio - Ajuste on-line do controlador e/ou identificador

Neste estágio, os parâmetros são ajustados a cada ponto ou numa janela de tamanho determinado. Para o dispositivo neurocomputacional utilizando polinômios de Hermite como função de ativação foi verificado que quando os parâmetros c_{ij} (coeficiente dos polinômios de Hermite) são também atualizados, juntamente com v (direção de projeção) e w (pesos da camada de saída) os resultados são melhores quando somente os parâmetros v e w são ajustados. No entanto, o esforço computacional gasto no ajuste é muito maior. No caso dos dispositivos neurocomputacionais utilizando *splines* como funções de ativação, os únicos parâmetros ajustáveis são v e w. Poderíamos também ajustar a forma da função, isto

é, a suavidade da função, mas isto somente seria possível se estivéssemos considerando uma janela de tamanho relativamente grande. Nesta dissertação, não entraremos em detalhes acerca da influência do ajuste a cada ponto ou em janela, NARENDRA & PARTHASARATHY (1990) apresentam um estudo neste sentido. Em todas as simulações realizadas foi adotado o ajuste a cada ponto.

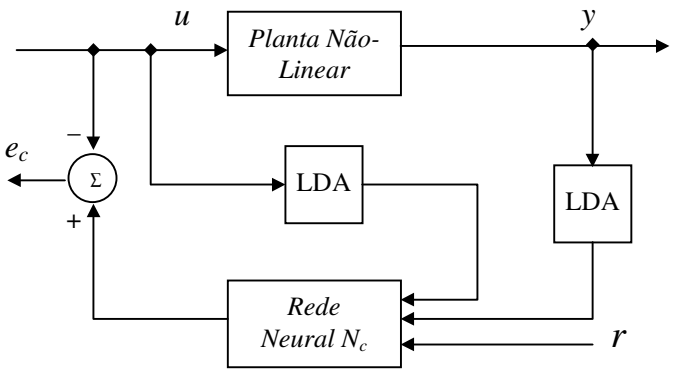


Figura 7.51 - Estrutura utilizada no treinamento *off-line* do controlador

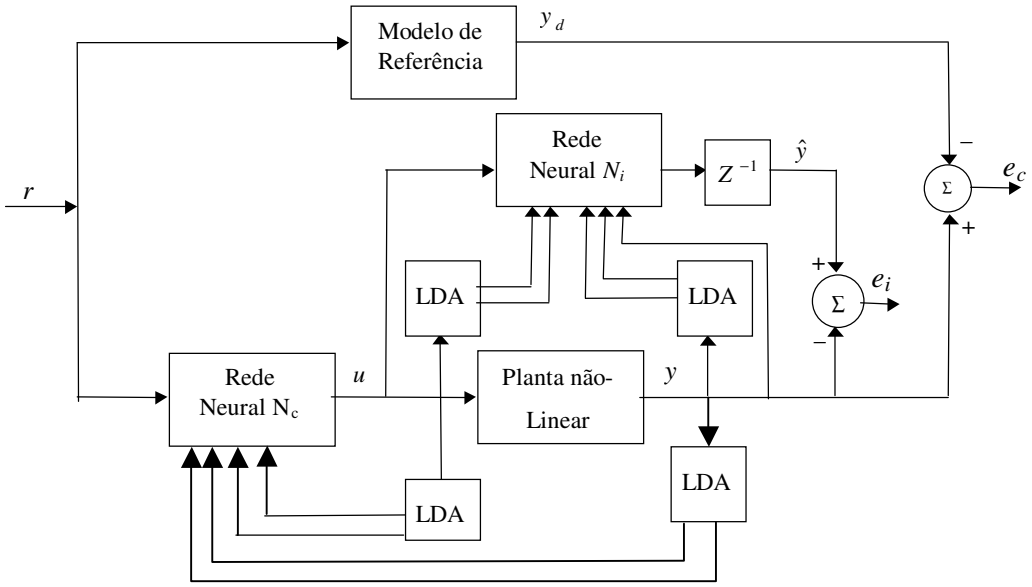


Figura 7.52 - Controle adaptativo indireto usando redes neurais em malha fechada (LDA: linha de derivação de atrasos)

7.3.5 Resultados de simulação

No capítulo 3, foram apresentados vários modelos para uma planta. Alguns destes, foram utilizados, a saber:

- *Modelo I*: Um controlador baseado no modelo linear da entrada-saída foi projetado para propósito de comparações. O modelo de identificação, neste caso, foi descrito pela equação a diferenças:

$$\hat{y}(k+1) = \hat{\theta}_1 y(k) + \cdots \hat{\theta}_{m+1} y(k-m) + \hat{\theta}_{m+2} u(k) + \cdots \hat{\theta}_{m+n+2} u(k-n) \quad (7.16)$$

A saída de controle foi calculada como:

$$u(k) = \frac{1}{\hat{\theta}_{m+2}} [y_m(k+1) - \hat{\theta}_1 y(k) - \cdots \hat{\theta}_{m+1} y(k-m) - \hat{\theta}_{m+3} u(k-1) - \hat{\theta}_{m+n+2} u(k-n)] \quad (7.17)$$

- *Modelo II*: O modelo de identificação foi descrito pela equação a diferença

$$\hat{y}(k+1) = R[y(k), y(k-1)] + \hat{\theta}_1 y(k) + \cdots + \hat{\theta}_{m+1} y(k-m) + \hat{\theta}_{m+2} u(k) + \cdots + \hat{\theta}_{m+n+2} u(k-n) \quad (7.18)$$

onde *R[.]* foi implementada por uma *RBF* (rede neural com funções de base radial, conforme apresentado no capítulo 2), com os centros iniciais distribuídos uniformemente no intervalo de interesse.

A saída de controle foi calculada como:

$$u(k) = \frac{1}{\hat{\theta}_{m+2}} \{ y_m(k+1) - R[y(k), y(k-1)] - \hat{\theta}_1 y(k) - \cdots - \hat{\theta}_{m+1} y(k-m) - \hat{\theta}_{m+3} u(k-1) - \cdots - \hat{\theta}_{m+n+2} u(k-n) \}$$

- *Modelo III*: O modelo de identificação foi descrito pela equação a diferença

$$\begin{aligned} \hat{y}(k+1) = {} & R_1[y(k), y(k-1)] + R_2[y(k), y(k-1)] u(k) + R_3[y(k), y(k-1)] u(k-1) \\ & + \hat{\theta}_1 y(k) + \cdots + \hat{\theta}_{m+1} y(k-m) + \hat{\theta}_{m+2} u(k) + \cdots + \hat{\theta}_{m+n+2} u(k-n) \end{aligned} \quad (7.19)$$

onde $R_i[.]$, $i = 1,...,3$, foi implementada por uma *RBF* (rede neural com funções de base radial, conforme apresentado no capítulo 2), com os centros iniciais distribuídos uniformemente no intervalo de interesse.

A saída de controle foi calculada como:

$$\begin{aligned} u(k) = {} & \frac{1}{R_3[y(k), y(k-1)] + \hat{\theta}_{m+2}} \{ y_m(k+1) - R_1[y(k), y(k-1)] - R_3[y(k), y(k-1)] u(k-1) \\ & - \hat{\theta}_1 y(k) - \cdots - \hat{\theta}_{m+1} y(k-m) - \hat{\theta}_{m+3} u(k-1) - \cdots - \hat{\theta}_{m+n+2} u(k-n) \} \end{aligned} \quad (7.20)$$

- *Modelo IV*: O modelo de identificação foi descrito pela equação a diferença

$$y(k+1) = N[y(k), y(k-1), \cdots, y(k-m), u(k), \cdots, u(k-n)] \tag{7.21}$$

onde *N[.]* foi implementada utilizando uma rede neural tradicional ou um dispositivo neurocomputacional híbrido, onde *m* e *n* são, respectivamente, o número de atrasos da saída da planta e da entrada de controle.

A saída de controle foi computada como:

$$u(k) = N[y(k), y(k-1), \cdots y(k-m), u(k-1), \cdots u(k-n), r(k)]$$

Caso 1

Neste exemplo, a saída da planta depende de seu valor anterior e da entrada de controle em termos desacoplados, conforme descrito abaixo

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \tag{7.22}$$

A saída do modelo de referência é dada por:

$$y_m(k+1) = 0.6 \cdot y_m(k) + r(k) \tag{7.23}$$

O conjunto de treinamento para construção do modelo de identificação e do controlador é composto de 1000 pontos gerados a partir da equação (7.21), supondo um sinal de entrada randômico *u(k)*, uniformemente distribuído no intervalo [-2,+2], para modelo de identificação e [-1.7,+1.7] para o controlador. Após concluído o treinamento, tanto o modelo de identificação quanto o sistema de controle como um todo são testados. O teste ou validação do modelo de identificação é realizado fazendo a entrada de controle *u(k) = r(k)*, onde *r(k)* é dado pela equação (7.24). Já o teste do controlador é realizado colocando-se o controlador em malha fechada com a planta, conforme mostrado na figura 7.51, aplicando o sinal *r(k)* ao controlador e medindo a saída da planta. O sinal *r(k)* é dado por:

$$r(k) = \sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{2\pi k}{10}\right) \tag{7.24}$$

Tabela 7.7 - Dados dos modelos utilizados (modelo estático com linhas de atrasos na entrada)

	Planta 1			
	Identificação		Controle	
	N.º Neurônios	Tipo de entrada	N.º Neurônios	Tipo de entrada
Modelo I	------	*y(k), u(k)*	----	*r(k), y(k)*
Modelo II	36*	*y(k), u(k)*	36	*r(k), y(k)*
Modelo III	108*	*y(k), u(k)*	108	*r(k), y(k)*
Modelo IV	5	*y(k), u(k)*	8	*r(k), y(k)*
DN_Hermite	3	*y(k), u(k)*	5	*r(k), y(k)*
DN_Spline	2	*y(k), u(k)*	5	*r(k), y(k)*

* os centros foram inicialmente distribuídos uniformemente no intervalo [−4,4].

Tabela 7.8 – Desempenho dos modelos de identificação e controle

	Planta 1				
	Identificação		Controle		
			OFF_LINE		ON_LINE
	Erro de treinamento	Erro de teste	Erro de treinamento	Erro de teste	Erro de teste
Modelo I	0.090971	0.155823	----------	----------	0.489099
Modelo II	0.071989	0.124511	----------	----------	0.359806
Modelo III	0.016789	0.040098	----------	----------	0.259876
Modelo IV	0.001471	0.001519	0.078906	0.258978	0.157890
DN_Hermite	0.000054	0.000051	0.016648	0.300631	0.042333
DN_Spline	0.000030	0.000021	0.060217	0.428980	0.064064

- Resultado da identificação e controle utilizando a abordagem DN_Hermite

Resultado da identificação da planta

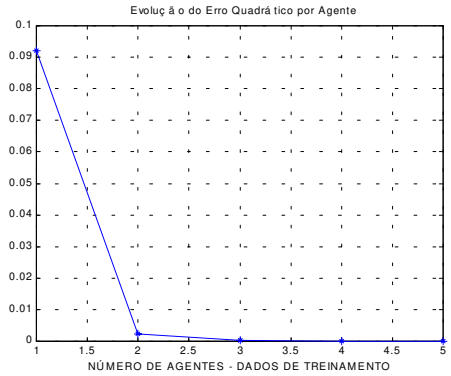


Figura 7.53 - Evolução do erro quadrático durante o processo de construção do dispositivo neurocomputacional híbrido

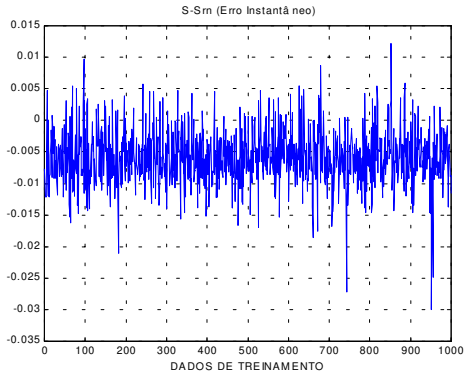


Figura 7.54 Erro instantâneo no conjunto de teste

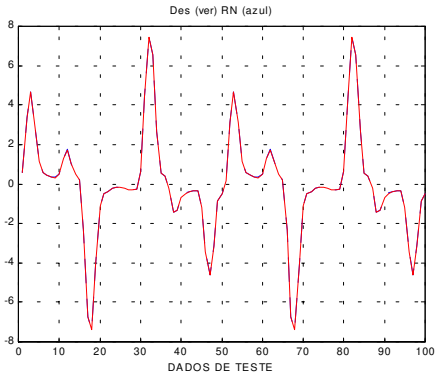


Figura 7.55 - Saída do *DN_Hermite* (RN: azul) × saída da planta (DES: vermelha)

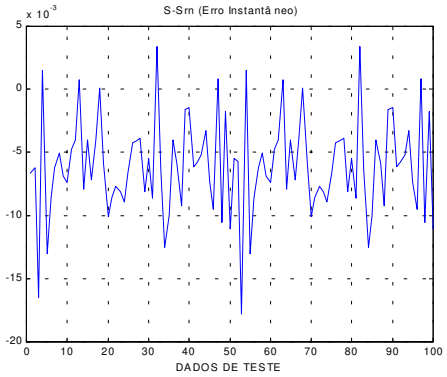


Figura 7.56 - Erro instantâneo: Saída da planta (S) – saída do *DN_Hermite* (Srn)

A figura 7.53 apresenta a evolução do erro quadrático considerando-se até 5 neurônios (agentes), no entanto, os dois últimos (quarto e o quinto) agentes foram retirados por apresentarem uma contribuição muito pequena para a diminuição do erro quadrático no conjunto de treinamento. Após a retirada destes agentes, o *DN_Hermite* sofreu um treinamento de reforço. Tal treinamento é necessário para que os agentes restantes possam ser comunicados da ausência de alguns agentes e com isto, através do processo de

competição e cooperação, irão adaptar-se à nova configuração da rede neural, minimizando o impacto da ausência de alguns agentes.

A figura 7.55 apresenta a saída do *DN_Hermite* e da planta quando não está presente o controlador, isto é, $u(k) = r(k)$. Conforme podemos observar, alcançamos um ótimo desempenho no conjunto de teste.

Resultado de Controle *off_line*

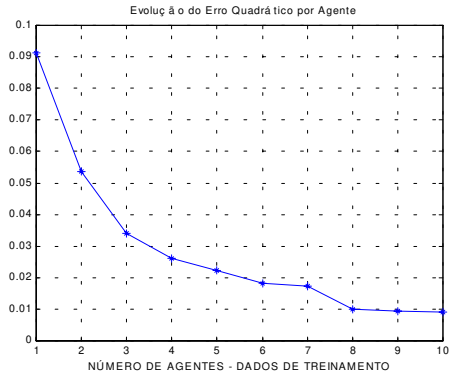


Figura 7.57 - Evolução do erro quadrático durante o processo de construção do dispositivo neurocomputacional híbrido

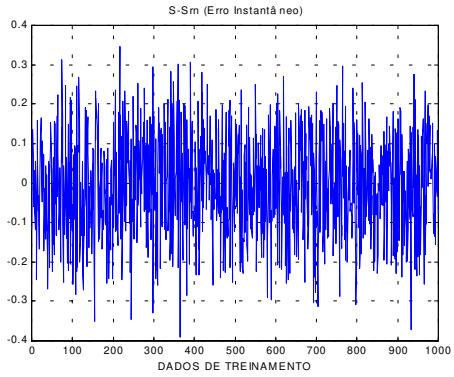


Figura 7.58 - Erro instantâneo no conjunto de teste

A figura 7.57 apresenta 10 agentes, no entanto, somente 5 agentes foram necessários. Isto deve-se ao fato de que após o treinamento é realizado um procedimento de poda, conforme já mencionado. Analisando a figura 7.57 podemos observar que 3 agentes foram retirados devido ao fato de apresentarem baixa contribuição para diminuição do erro quadrático. Os demais agentes foram retirados por produzirem componentes de alta freqüência, ou seja, por serem não-informativos. Estas contribuições para altas freqüências, apesar de ajudarem a diminuição do erro quadrático no conjunto de treinamento, tendem a produzir um aumento do erro quadrático no conjunto de teste.

Várias são as formas de identificar componentes de altas freqüências em processamento de sinais, como por exemplo análise da transformada de Fourier ou densidade espectral de potência do sinal. No entanto, a técnica utilizada nesta dissertação consiste na análise da magnitude da relação entre a norma do vetor c_{ij} ($j = 1,..,P$) do agente i ($i = 2,..,n$) para a norma do vetor c_{1j} ($j = 1,...,P$) do primeiro agente adicionado à rede, ou seja, temos uma norma relativa. Com este procedimento obtemos um valor que mede o

aumento da importância dos c_{ij}, uma vez que sempre o primeiro agente tende a não apresentar componentes de altas freqüências (coeficiente de c_{ij} elevados). Nos *DN_Hermite* estas contribuições podem aparecer devido ao fato de que tais dispositivos neurocomputacionais não apresentam nenhum critério de penalidade na função custo ou validação cruzada como é o caso dos *DN_Spline*.

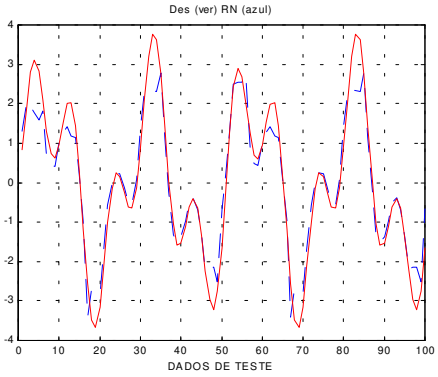


Figura 7.59 - Saída do modelo de referência (vermelha) x saída da planta (azul)

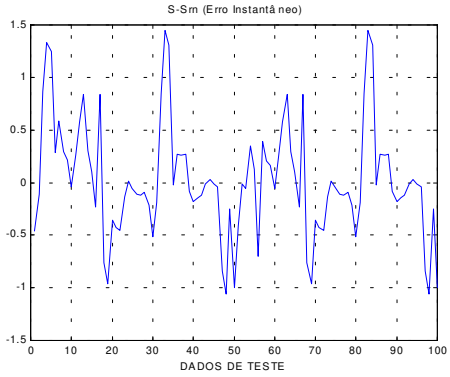


Figura 7.60 - Erro instantâneo (Saída do modelo de referência – saída da planta)

A figura 7.59 mostra a saída da planta e a saída de do modelo de referência, quando o controlador é utilizado. A obtenção deste controlador seguiu os passos descritos no segundo estagio. Note que a saída passa a adquirir a forma do modelo de referência. No entanto, existe algumas regiões com erros, ainda elevados, que serão eliminados ou reduzidos no treinamento *on-line.*

Resultado de controle *on-line*

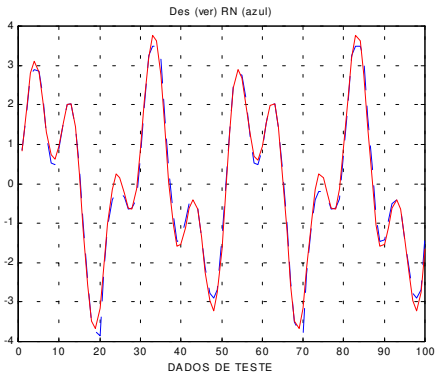


Figura 7.61 - Saída do modelo de referência (vermelha) x saída da planta (azul)

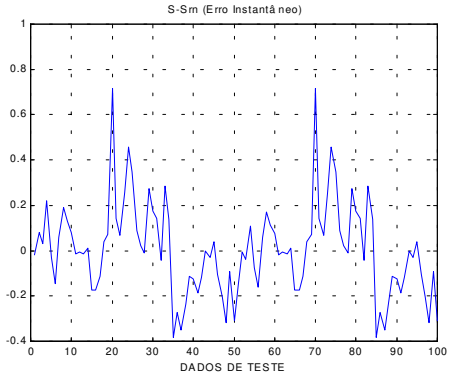


Figura 7.62 - Erro instantâneo (Saída do modelo de referência – saída da planta)

A figura 7.61 ilustra a saída da planta e a saída do modelo de referência na presença do controlador anterior, após ter sofrido também treinamento *on-line*. Como podemos observar a saída da planta realmente acompanha a saída do modelo de referência.

Caso 2

Neste exemplo, a saída da planta é uma combinação não-linear da saídas anteriores e da entrada de controle, conforme apresentado abaixo:

$$y(k+1) = \frac{5y(k)y(k-1)}{1+y^2(k)+y^2(k-1)+y^2(k-2)} + u(k) + 0.8u(k-1) \tag{7.25}$$

A saída do modelo de referencia é descrita por:

$$y_m(k+1) = 0.32 \cdot y_m(k) + 0.64 \cdot y_m(k-1) - 0.5 \cdot y_m(k-2) + r(k) \tag{7.26}$$

O conjunto de treinamento para construção do modelo e do controlador é composto de 1000 pontos gerados a partir do modelo da planta acima, supondo um sinal de entrada randômico *u(k)*, uniformemente distribuído no intervalo [–2,+2], tanto para o modelo de identificação quanto para o controlador, mas independentes entre si. Após concluído o treinamento, tanto o modelo de identificação quanto o sistema de controle com um todo é testado, utilizando o mesmo procedimento adotado na planta anterior (caso 1). Os resultados para vários modelos são apresentada na tabela 7.10

$$r(k) = \sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{2\pi k}{10}\right) \tag{7.27}$$

Tabela 7.9 - Dados dos modelos utilizados (modelo estático com linha de atrasos na entrada)

	Planta 2			
	Identificação		Controle	
	N.º Neurônios	Tipo de entrada	N.º Neurônios	Tipo de entrada
Modelo I	----------	*y(k), y(k-1), y(k-2), u(k),u(k-1)*	----------	*r(k), y(k), y(k-1) y(k-2), u(k-1),*
Modelo II	243*	*y(k), y(k-1), y(k-2), u(k),u(k-1)*	243	*r(k), y(k), y(k-1) y(k-2), u(k-1),*
Modelo III	729*	*y(k), y(k-1), y(k-2), u(k),u(k-1)*	729	*r(k), y(k), y(k-1) y(k-2), u(k-1),*
Modelo IV	10	*y(k), y(k-1), y(k-2), u(k),u(k-1)*	10	*r(k), y(k), y(k-1) y(k-2), u(k-1),*
DN_Hermite	5	*y(k), y(k-1), y(k-2), u(k),u(k-1)*	5	*r(k), y(k), y(k-1) y(k-2), u(k-1),*
DN_Spline	5	*y(k), y(k-1), y(k-2), u(k),u(k-1)*	5	*r(k), y(k), y(k-1) y(k-2), u(k-1),*

*o número elevado de neurônio deve-se ao fato da alta dimensionalidade do espaço de entrada. O número de neurônios foi tomado com sendo 3^n, onde *n* representa o número de entradas.

Tabela 7.10 – Desempenho dos modelos de identificação e controle

	Planta 2				
	Identificação		**Controle**		
			OFF_LINE		ON_LINE
	Erro de treinamento	Erro de teste	Erro de treinamento	Erro de teste	Erro de teste
Modelo I	0.241569	0.321896	----------	----------	0.452567
Modelo II	0.101189	0.221672	----------	----------	0.293156
Modelo III	0.088997	0.108794	----------	----------	0.221894
Modelo IV	0.061907	0.080898	0.045984	0.276876	0.120449
DN_Hermite	0.021432	0.030616	0.040855	0.188258	0.061087
DN_Spline	0.019393	0.022819	0.029257	0.169927	0.042260

- Resultados da identificação e controle utilizando a abordagem DN_Spline

Resultado de identificação

Figura 7.63 - Evolução do erro quadrático durante o processo de construção do dispositivo neurocomputacional híbrido

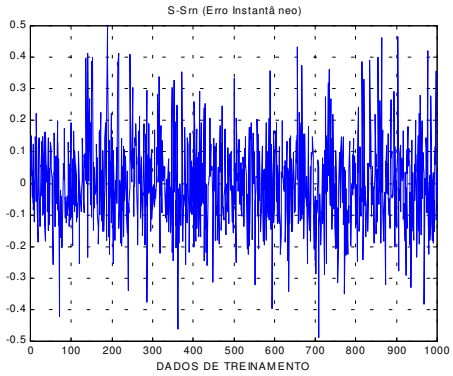


Figura 7.64 Erro instantâneo no conjunto de teste

Ao contrário da planta anterior (caso 1), não houve a necessidade de retirar nenhum agente durante o procedimento de poda realizado. Na figura 7.67 apresentamos a saída da planta e do *DN_Spline* na ausência do controlador.

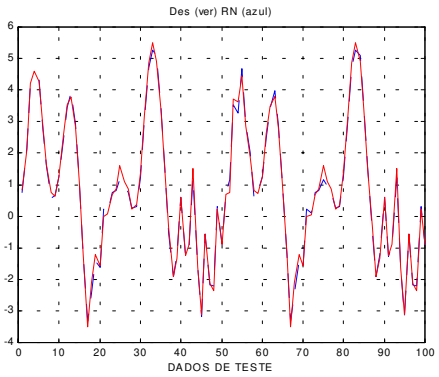


Figura 7.65 - Saída do *DN_Spline* (RN: azul) x saída da planta (DES: vermelha)

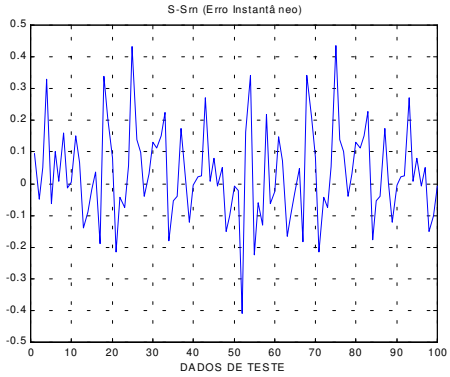


Figura 7.66 - Erro instantâneo: Saída da planta (S) – saída do *DN_Spline* (Srn)

Resultado de Controle *off_line*

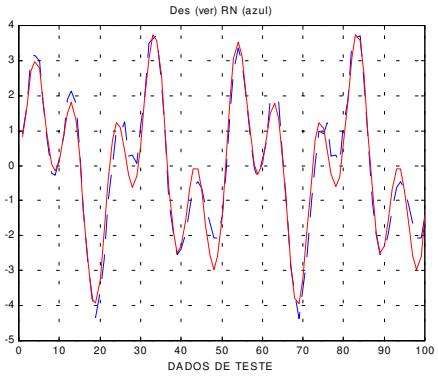


Figura 7.67 - Saída do modelo de referência (vermelha) x saída da planta (azul)

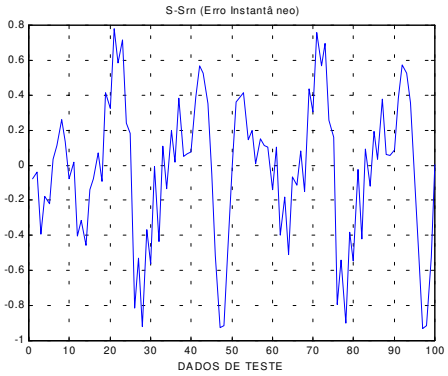


Figura 7.68 - Erro instantâneo (Saída do modelo de referência – saída da planta)

Resultado de controle *on-line*

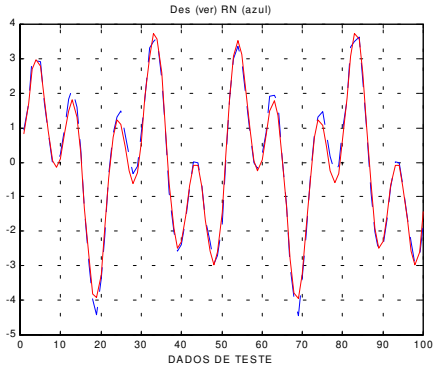


Figura 7.69 - Saída do modelo de referência (vermelha) x saída da planta (azul)

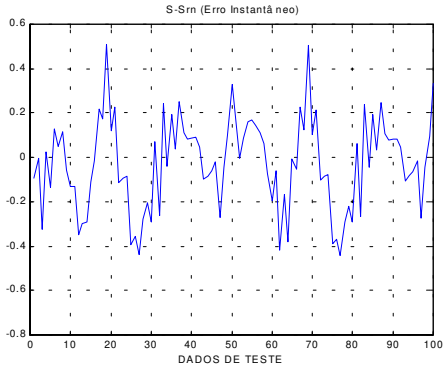


Figura 7.70 - Erro instantâneo (Saída do modelo de referência – saída da planta)

7.4 Análise do dispositivo neurocomputacional híbrido

Para ilustrar como o método de aproximação por busca de projeção realiza a aproximação de funções, considere o primeiro problema de identificação apresentado na seção 7.3.3 (caso 2) utilizando *DN_Hermite*, cujo desempenho é mostrado na tabela 7.5. As direções de projeções escolhidas para cada um dos agentes ou neurônios da camada intermediária e as respectivas funções de ativação são apresentadas, respectivamente, na Tabela 7.11 e nas figuras 7.71 a 7.76.

Tabela 7.11 – As direções de projeção escolhidas para *DN_Hermite*

	Eixos Coordenados		
	u(k)	*y(k)*	*y(k-1)*
1ª direção	-0.041526	-0.154477	-0.353771
2ª direção	0.590152	0.407951	0.000369
3ª direção	-0.511678	0.492766	0.001185
4ª direção	0.205783	-0.479769	-0.004685
5ª direção	-0.293872	-0.643541	-0.002054
6ª direção	-0.831326	0.406370	-0.018132

Os gráficos para o *DN_Hermite* são apresentados a seguir:

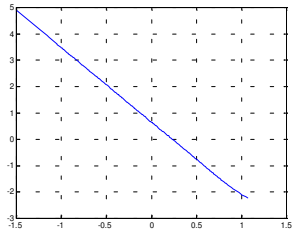


Figura 7.71 – 1ª Função de ativação

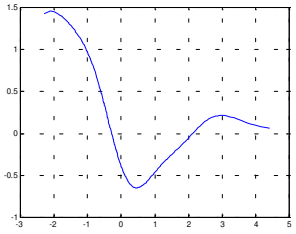


Figura 7.72 – 2ª Função de ativação

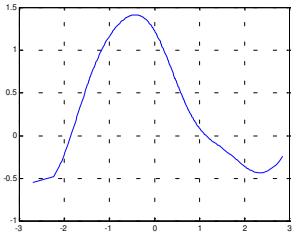


Figura 7.73 – 3ª Função de ativação

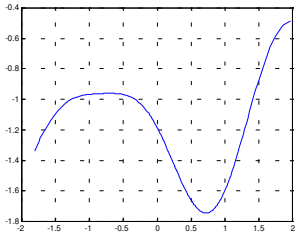


Figura 7.74 – 4ª Função de ativação

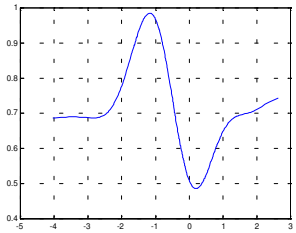


Figura 7.75 – 5ª Função de ativação

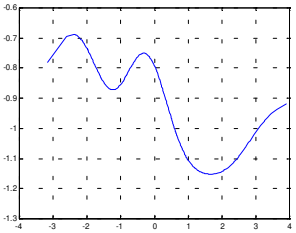


Figura 7.76 – 6ª Função de ativação

Estes resultados suscitam os seguintes comentários:

- todas as soluções obtidas neste capítulo, utilizando dispositivos neurocomputacionais, poderiam ter sido apresentadas incluindo as direções de projeção e o formato das funções de ativação de cada agente neurocomputacional. Estas informações são importantes inclusive para se extrair conhecimento acerca do problema de aplicação, como será abordado na próxima seção. Duas foram as razões que levaram à não

inclusão de informações como as apresentadas acima junto a todos os resultados obtidos: a necessidade de evitar uma extensão ainda maior do texto e a ausência de preocupação em apresentar resultados conclusivos, conforme adiantado no início do capítulo.

- a apresentação da configuração do dispositivo neurocomputacional em termos dos parâmetros e relações funcionais individuais de cada agente neurocomputacional evidenciam mais uma vez que técnicas de treinamento construtivo fornecem um papel mais significativo a cada unidade de processamento (neurônio artificial ou neuroagente), diferente do que ocorre com arquiteturas convencionais de redes neurais artificiais.

- uma análise mais cuidadosa das direções de projeção apresentadas na tabela 7.11, e automaticamente fornecidas pelo processo de treinamento construtivo, permite verificar que cada agente neurocomputacional "observa" os dados disponíveis para treinamento sob um "ponto de vista" distinto. Fica evidente, portanto, no resultado final, o aspecto de cooperação entre neuroagentes, que irão se agregar em uma composição aditiva para fornecer a solução do problema. No entanto, a convergência deste processo de composição está associada a mecanismos de competição, como descrito no capítulo 6.

- o número predominantemente reduzido de neuroagentes que estão presentes nas soluções obtidas ao longo do capítulo também pode ser explicado a partir de uma constatação grosseira (mas que pode ser devidamente refinada) extraída da análise dos resultados da tabela 7.11 e das figuras 7.71 a 7.76: os neuroagentes estão desempenhando não apenas papéis relevantes na produção da solução resultante, mas estes papéis podem ser caracterizados por aspectos de "ortogonalidade". Com isso, dado que os neuroagentes representam elementos de uma composição aditiva, a ortogonalidade de papéis confere parcimônia à solução, ou seja, se consegue obter a solução com um número reduzido de neuroagentes.

7.5 Extração de conhecimento via dispositivo neurocomputacional híbrido

Conforme havíamos comentado no capítulo 6, ao contrário das rede neurais tradicionais, é possível extrair conhecimento dos dispositivos neurocomputacionais híbridos. A título ilustrativo, considere a planta descrita pela equação 7.22. Nesta equação, existem dois termos, um termo é função cúbica da entrada de controle e o outro termo é uma função não-linear da saída atrasada da planta. NARENDRA & PARTHASARATHY (1990) identificaram estes termos utilizando rede neurais tradicionais. No entanto, para realizar tal procedimento estes autores supuseram o conhecimento da equação (7.21), a partir da qual foram gerados dados para identificar tais termos. Com os dispositivos neurocomputacionais híbridos podemos identificar tais termos com apenas o conhecimento dos dados de entrada e saída da planta, não precisando gerar modelos individuais para cada termo. Na verdade, cada agente irá tentar descobrir uma projeção que evidencie aspectos relevantes para a planta. Ao realizar tal tarefa os agentes estarão identificando termos individuais da planta. Portanto, se tais considerações são verdadeiras, deveríamos precisar de dois agentes para aproximar a planta dada pela equação 7.22. Quando analisamos a tabela 7.7, verificamos que, para o dispositivo neurocomputacional híbrido utilizando polinômios de Hermite, foi necessário três agentes. Isto se deve aos fato de que o primeiro termo da planta é uma função difícil de ser aproximada por um único agente com funções básicas utilizando polinômios de Hermite. Com isto há necessidade de mais de um agente para tratar um único termo. No entanto, se analisarmos a tabela 7.7 para o dispositivo neurocomputacional híbrido utilizando *splines*, verificamos que foram necessários somente dois agentes.

Analisando as direções de projeções, conforme apresentado na Tabela 7.12 e 7.13, observamos que as duas direções de projeção obtidas para a abordagem DN_Splines, tendem a ser exatamente uma na direção de $u(k)$ e a outra na direção de $y(k)$. Portanto, as funções de ativação mostradas na figura 7.78 representam corretamente os dois termos da planta. Com isto, podemos notar que a função de ativação de cada agente poderá fornecer uma informação de alto nível a respeito do sistema, o que não pode ser obtido através de rede neurais tradicionais.

Tabela 7.12 – As direções de projeções escolhidas para o *DN_Hermite*

	Eixos Coordenados	
	u(k)	*y(k)*
1ª direção	-0.33997	-0.00000
2ª direção	0.00099	-0.36704
3ª direção	-0.00079	-1.50385

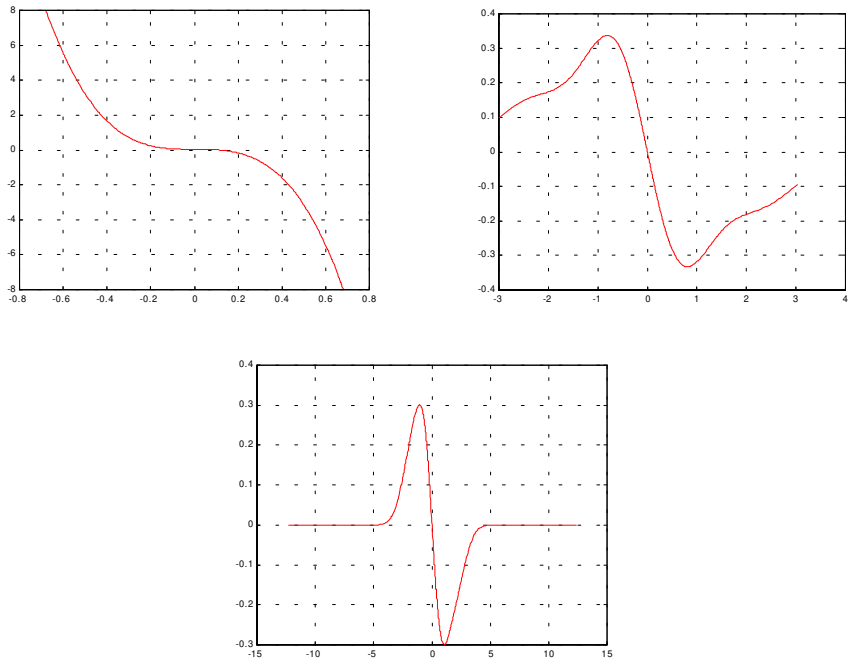


Figura 7.77 – As funções de ativação para *DN_Hermite*

Tabela 7.13 – As direções de projeções escolhidas para o *DN_Spline*

	Eixos coordenados	
	u(k)	*y(k)*
1ª direção	0.99999	-0.00097
2ª direção	-0.00019	0.99999

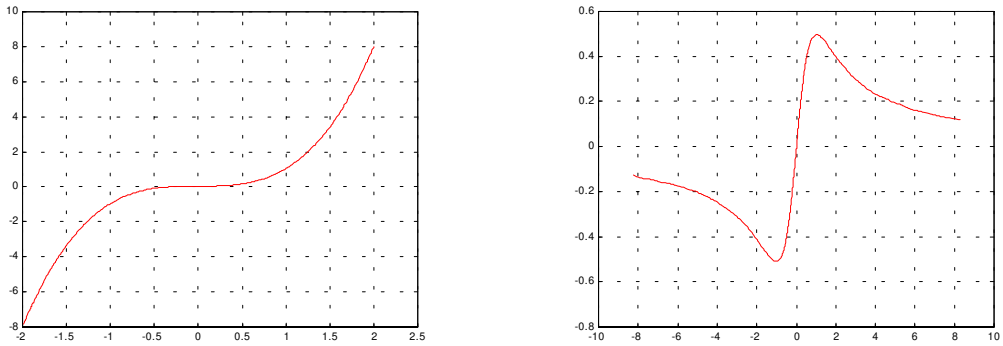


Figura 7.78 – As funções de ativação para *DN_Spline*

7.6 Visão sintética do capítulo

Neste capítulo, apresentamos os resultados referentes à aplicação de modelos lineares, redes neurais tradicionais e dispositivos neurocomputacionais híbridos, estes últimos propostos no capítulo 6, a problemas de aproximação de funções, predição de séries temporais, identificação de sistemas dinâmicos e controle de processos. Os problemas foram escolhidos arbitrariamente, assim como os modelos de identificação e controle, embora os problemas e os modelos tenham sido extraídos de artigos clássicos da literatura afim e da formalização presente nos capítulos anteriores desta dissertação, respectivamente.

No entanto, não foram realizadas análises comparativas com outras abordagens da literatura, visto que o propósito aqui não foi o de produzir resultados conclusivos. Sendo mais específico, todos os experimentos realizados tiveram como objetivo fundamental demonstrar a viabilidade e o potencial do modelo proposto quando aplicado a problemas com não-linearidades significativas.

Finalmente, foi mostrado que, ao contrário da abordagem conexionista tradicional, o modelo proposto nesta dissertação permite a interpretabilidade dos resultados, além de ser parcimonioso no uso de recursos computacionais, o que amplia ainda mais os benefícios advindos de sua aplicação. No entanto, a parcimônia de modelos não-paramétricos em geral custa caro, pois obtê-los via treinamento supervisionado pode chegar a consumir de 10 a 100 vezes mais recursos computacionais quando comparados aos modelos paramétricos (HÄRDLE, 1990). Na apresentação dos resultados deste capítulo, optamos por não dar ênfase aos custos de treinamento (processo de obtenção da solução), e sim ao custo de

implementação da solução obtida, basicamente representada pelo número de neuroagentes, ou neurônios artificiais, utilizados.

A razão para se adotar esta política de apresentação de resultados está no fato de que a parcimônia mencionada acima deve ser adequadamente interpretada, principalmente no contexto de controle de processos, o qual normalmente vai conduzir a dispositivos de controle a serem produzidos em larga escala. Sendo assim, o alto custo de obtenção da solução tem uma contrapartida, representada pela parcimônia na implementação da solução.

Logo, no contexto de controle de processos, o dispositivo de controle em si, o qual será produzido em larga escala, requer sempre menos recursos computacionais para sua implementação e execução, conduzindo, portanto, a uma economia de recursos: o que vai ser produzido em larga escala é o que determina o custo do produto, uma vez que o custo de projeto vai acabar sendo amortizado.

Esta tendência generalizada de se ampliar os custos de projeto de produtos tecnológicos visando produzir dispositivos mais baratos e eficazes está amplamente difundida em sistemas modernos de engenharia. Sob este ponto de vista, esta dissertação acaba alimentando este mecanismo de concepção de solução para problemas atuais de engenharia.

Capítulo 8

Conclusão e perspectivas futuras

8.1 O enfoque da pesquisa revisitado

As ferramentas computacionais propostas durante o desenvolvimento da pesquisa e apresentadas ao longo desta dissertação, dentre outras potencialidades, mostraram-se eficientes no tratamento de dois problemas complexos de engenharia moderna:

1. como resolver um problema de aprendizado de máquina pela geração automática de uma estrutura conexionista, parcimoniosa em termos de uso de recursos computacionais, sem conhecimento a priori de boa parte dos aspectos vinculados à natureza e complexidade do problema?
2. como modelar uma planta (sistema dinâmico a ser controlado), suposta desconhecida e com características não-lineares significativas, empregando recursos matemáticos manipuláveis algebricamente e em computador, de modo a permitir o uso do modelo resultante no projeto de controladores, sem provocar a instabilidade do sistema e ao mesmo tempo otimizando algum critério de desempenho formulado matematicamente?

Quando se toma a iniciativa de aplicar estratégias conexionistas a identificação e controle de processos, verifica-se que estes dois problemas apresentam vários aspectos em comum, de modo que a solução do problema 1 transforma-se em um pré-requisito fundamental para se resolver o problema 2.

Sendo assim, esta dissertação aborda mais intensamente a análise e a síntese de ferramentas destinadas a solução do problema 1, recorrendo para tanto a conceitos avançados de aprendizado, otimização e aproximação de funções. Uma característica peculiar dos métodos de

aprendizado mais eficazes que já foram propostos para o problema 1, todos eles abordados no capítulo 5, é o fato de terem sido concebidos a partir de modelos avançados de regressão não-paramétrica, portanto oriundos da área de estatística. Sendo assim, consideramos como uma das contribuições mais importantes desta dissertação o esforço despendido em formular este processo de aprendizado, envolvendo estruturas de processamento não-paramétricas, recorrendo a conceitos básicos de sistemas multi-agentes e fortemente vinculado à teoria de computação.

Em relação ao problema 2, os objetivos do projeto de pesquisa que culminou com esta dissertação foram menos ambiciosos, tendo sido executadas duas tarefas básicas: descrever como são aplicadas as estruturas conexionistas à identificação e ao controle de processos; e demonstrar, através da análise de resultados de simulação computacional, as potencialidades das ferramentas de solução do problema 1 quando aplicadas a exemplos de casos envolvendo o problema 2.

No entanto, sabemos que um longo caminho ainda deve ser percorrido antes de se testemunhar a incorporação das contribuições aqui apresentadas, e outras previamente existentes, em processos industriais. Conforme enfatizado por NG (1997), apesar de o primeiro controlador baseado em conceitos conexionistas ter sido desenvolvido por WIDROW & SMITH (1963), há quase 40 anos atrás, e sabendo que a comunidade científica tem contribuído ativamente na produção de resultados (só a revista especializada *IEEE Control System Magazine* devotou pelo menos 3 exemplares ao emprego de estratégias conexionistas em controle de processos, nos anos de 1988, 1990 e 1992), ainda 90% das aplicações industriais, envolvendo controle e identificação de sistemas, empregam controladores convencionais, como por exemplo os tradicionais controladores PID (SEBORG, 1994). Além disso, há uma forte tendência à alocação de supervisores humanos a tarefas que envolvem níveis refinados de controle e tomada de decisão, mesmo em situações em que os custos são elevados e o desempenho do operador humano é insatisfatório, em face da crescente complexidade e multiplicidade de objetivos concorrentes.

8.2 Contribuições e resultados obtidos

As contribuições deste trabalho estão basicamente concentradas nos capítulos 5, 6 e 7, sendo que os capítulos anteriores fornecem os subsídios necessários ao posicionamento e à justificativa para as iniciativas adotadas ao longo destes três capítulos.

No capítulo 5, é apresentada uma formalização para modelos não-convencionais de redes neurais artificiais, a qual foi elaborada a partir da agregação de diversos resultados já propostos na literatura, mas que geralmente são apresentados de forma isolada. Na literatura prévia a esta pesquisa, não houve, em essência, um consenso em termos de argumentação para evidenciar as potencialidades e principais linhas de aplicação de abordagens não-paramétricas. O capítulo 5, por sua vez, é o produto de um esforço no sentido de unificar a notação, levantar e associar todos os conceitos multidisciplinares relevantes, além de especificar as reais potencialidades de modelos baseados em projeção e modelos não-paramétricos.

Toda a formalização apresentada no capítulo 6, empregando uma visão de sistemas multi-agentes para o processo de aprendizado construtivo em redes neurais artificiais, representa uma iniciativa inédita no sentido de proporcionar um novo ponto de vista para descrever e analisar os dispositivos neurocomputacionais híbridos, que correspondem ao produto final de um processo de aprendizado construtivo.

O capítulo 7, por sua vez, contém resultados preliminares associados a problemas de aproximação de funções, predição de séries temporais, identificação de sistemas dinâmicos e controle de processos, incluindo comparação de alguns aspectos de desempenho com abordagens alternativas de modelagem. Os resultados apresentados são promissores, mas não podem ser interpretados como conclusivos, inclusive pelo fato de que os problemas de aplicação foram escolhidos arbitrariamente. O propósito básico foi sempre o de indicar o grande potencial dos dispositivos neurocomputacionais e as variadas possibilidades de aplicação.

Ainda no capítulo 7, vale destacar a quantidade reduzida de recursos computacionais necessários para implementar as soluções produzidas pelos dispositivos neurocomputacionais, demonstrando tratar-se de uma abordagem com maior capacidade de conduzir automaticamente a soluções dedicadas, independente da natureza das características requeridas pela aplicação e desde que estas possam ser extraídas pela aplicação de técnicas avançadas de treinamento supervisionado.

8.3 Perspectivas futuras

8.3.1 Modelos conexionistas e teoria de agentes

- espera-se que o novo "ângulo de visão" para modelos não-convencionais de redes neurais artificiais, desenvolvido basicamente ao longo do capítulo 6, permita estender inúmeros aspectos operacionais e conceituais envolvidos, de modo a contribuir no sentido de viabilizar a implementação de uma nova geração de algoritmos e modelos para aprendizado de máquina, dotados de maior autonomia e flexibilidade. Duas são as conseqüências imediatas: maior automatização de todo o processo de aprendizado e maior eficiência no uso dos recursos computacionais disponíveis.
- seguindo a mesma linha adotada no capítulo 5, procurar investir na definição de uma formalização unificada, baseada em sistemas multi-agentes, para todos os modelos conexionistas gerados a partir de algoritmos de aprendizado construtivo, particularmente aqueles baseados em métodos de projeção linear. Um benefício imediato desta iniciativa está no fornecimento de mais subsídios para permitir uma discriminação eficiente dos diversos modelos já propostos.

8.3.2 Dispositivos neurocomputacionais híbridos

- investigação de métodos de projeção não-linear e avaliação de seu potencial de aplicação aos mesmos tipos de problemas considerados neste trabalho. Usando um conceito de teoria de agentes, pode-se afirmar que uma técnica de projeção não-linear vai conduzir a estratégias poderosas de "visualização ou sensoriamento do ambiente" em que se está inserido. Por exemplo, através de projeção não-linear, relações não-lineares no espaço original podem se transformar em relações lineares no espaço de projeção, o que certamente facilitaria sobremaneira a obtenção da solução.
- contribuir no sentido de superar duas limitações inerentemente vinculadas ao processo de construção de dispositivos neurocomputacionais: a dificuldade de lidar com espaços de entrada de elevada dimensão e modelos com múltiplas saídas (nenhum destes casos foi tratado nesta dissertação). A estratégia adotada durante o processo de aprendizado, em que cada unidade básica ou neuroagente tem seus parâmetros ajustados, enquanto todos os demais permanecem com seus parâmetros fixos, parece não ser adequada para estas duas situações. É

intuitivo propor, como uma possível extensão, a possibilidade de que múltiplos neuroagentes passem simultaneamente por sua fase de adaptação.

8.3.3 Aplicação de estruturas conexionistas a controle e identificação de processos

Muitas das mesmas questões teóricas encontradas, por exemplo, no caso de controle adaptativo de sistemas lineares estão presentes nas aplicações das estruturas conexionistas que foram consideradas neste trabalho, mas com um acréscimo de complexidade devido à natureza não-linear dos problemas abordados e ferramentas empregadas em sua solução. Sendo assim, novos resultados são esperados em áreas como:

- processo de definição dos modelos de identificação e controle, incluindo não apenas a estrutura dos modelos, mas também o espaço de entrada a ser considerado (responsável por representar toda a informação disponível para a obtenção da solução), e aspectos técnicos de implementação computacional;
- complexidade dos algoritmos de otimização e aprendizado;
- questões de estabilidade e robustez dos processos de controle;
- aplicação a problemas de identificação e controle extraídos do mundo real e comparações mais refinadas com técnicas alternativas de solução já propostas na literatura;
- extensão (de parte) dos resultados obtidos no sentido de incluir o tratamento de sistemas dinâmicos variantes no tempo;
- análise de convergência local e estabilidade dos algoritmos de treinamento;
- aplicação de redes neurais em conjunto com outras técnicas de identificação e controle de processos, gerando sistemas híbridos de identificação e controle;
- aprofundar o estudo de casos envolvendo dinâmica contínua;
- aprofundar o estudo de casos envolvendo redes neurais recorrentes, de modo a explorar mais eficientemente seu alto potencial de síntese de comportamentos dinâmicos não-lineares via aprendizado.

Apêndice A

Alguns aspectos vinculados à teoria de agentes

A.1 Taxonomia de agentes

A classificação dos tipos de agentes existentes é apontada por diversos autores, baseados na funcionalidade e na forma de atuação (LASHKARI *et al.*, 1994; MAES, 1994; GREEN *et al.*, 1997). A maioria deles adota uma classificação onde os agentes são apresentados como: Agentes de Interface (autônomos ou não), Agentes Colaborativos (arquitetura multi-agentes) e Agentes Móveis. No entanto, esta classificação não é rígida.

Segundo MORREALE (1998), existem cinco características essenciais que definem um agente e o diferenciam de um software convencional. Estas são: *autonomia, reflexibilidade, aprendizado, cooperação e mobilidade*. MORREALE (1998) baseia-se em três dentre as cinco características citadas - autonomia, aprendizado e cooperação - e classifica um agente como sendo:

- *Básico* - se contiver pelo menos uma das três características;
- *Avançado* - se contiver duas das três características;
- Ideal - se contiver as três características.

NWANA & NDUMU (1996) propõem as seguintes classificações para os agentes:

- Os agentes podem ser classificados por sua mobilidade, isto é, sua habilidade para locomover-se em alguma rede. Esta habilidade torna possível a classificação dos agentes em: *agentes estáticos* e *agentes móveis*;
- Os agentes podem ser classificados como *deliberativos* ou *reflexivos*. Agentes deliberativos derivam do paradigma de pensamento que assegura que os agentes possuem um modelo de raciocínio simbólico, e empregam este modelo em

planejamento e negociação com outros agentes para encontrar seus objetivos. Por outro lado, os agentes reflexivos não têm qualquer modelo simbólico explícito de seu ambiente, e agem usando um comportamento estímulo/resposta típico para responder ao estado presente do ambiente no qual encontram-se embutidos (FERBER, 1994). A formalização dos conceitos sobre agentes reflexivos originou-se da pesquisa realizada por BROOKS (1986) e AGRE & CHAPMAN (1987).

- Os agentes podem ser classificados com base em alguns atributos que idealmente deveriam exibir. Da mesma forma que MORREALE (1998), NWANA & NDUMU (1996) apresentaram uma lista mínima de três características que os agentes deveriam exibir: autonomia, aprendizado e cooperação. Autonomia refere-se ao princípio de que os agentes podem operar independentemente, sem a necessidade de serem guiados por seres humanos. Um elemento chave da autonomia é a pró-atividade, isto é, a habilidade para tomar iniciativa. A cooperação com outros agentes é necessária quando se pretende operar com múltiplos agentes. A comunicação requerida para assegurar a cooperação entre os agentes geralmente envolve o uso de mensagens de alto nível. O uso de mensagens de alto nível leva para um baixo custo de comunicação, fácil reimplementabilidade e concorrência. Por último, para os agentes serem verdadeiramente inteligentes, eles deveriam apresentar capacidade de aprendizado quando reagem e/ou interagem com seu ambiente externo, de modo que, a cada instante, seu desempenho melhore cada vez mais. Os autores usaram estas três características, apresentadas na figura A.1, para derivar três tipos de agentes: agente colaborativo, agente de interface e agente verdadeiramente inteligente. Eles enfatizam que estas distinções não são únicas. Por exemplo, no caso do agente colaborativo, há maior ênfase na cooperação e autonomia do que no aprendizado, mas não implica que agentes colaborativos nunca aprendam. Além disso, eles não consideram que a classificação apresentada na figura A.1 abranja todos os agentes existentes, ou seja, podem existir agentes que estejam fora do universo (delimitado pelos círculos) apresentado na figura A.1. Por exemplo, muitos sistemas especialistas têm grande autonomia, mas tipicamente eles não cooperaram ou aprendem. Idealmente, os agentes deveriam possuir todas as três características definidas acima, mas esta é uma aspiração

que está longe de ser atendida. Segundo estes critérios, agentes verdadeiramente inteligentes ainda não existem.

- Os agentes podem também ser classificados pelo seu papel. Por exemplo, agentes coletando informação na *Web*. Essencialmente, tais agentes ajudam a gerenciar a enorme quantidade de informação disponível na Internet. Esta classe de agentes é referida como agentes de Internet ou de informação. Novamente, tal agente de informação pode ser estático ou móvel, deliberativo ou reflexivo, etc.
- agentes podem ser híbridos, isto é, agentes que combinam dois ou mais agentes com filosofias diferentes formando um único agente.
- Os agentes podem ser classificados de acordo com outros atributos secundários em relação àqueles já mencionados acima. Por exemplo, o agente é versátil (isto é, tem muitos objetivos ou é empregado para realizar uma variedade de tarefas)? É um agente benevolente ou malevolente, egoísta ou altruísta? O agente mente astutamente ou ele é sempre verdadeiro? É temporariamente contínuo? Alguns pesquisadores, além dos atributos acima, estão também atribuindo atitudes emocionais aos agentes (BATES, 1994). Alguns agentes são projetados através de atitudes de procedência mental, tais como crença, desejos e intenção.

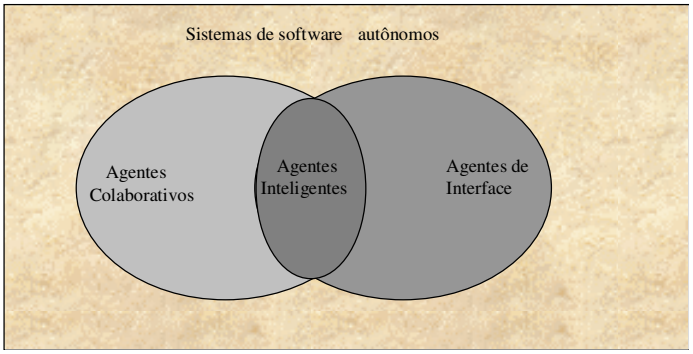


Figura A.1 Uma parte da taxonomia de agentes

A.1.2 Agentes colaborativos

Embora os agentes de interface tenham sucesso em seu aprendizado, algumas das suas possíveis aplicações não são viáveis, devido, principalmente, aos seguintes problemas:

- necessidade de um tempo considerável para acumularem conhecimentos;
- competência restrita a situações similares às encontradas no ambiente do usuário.

A troca de informação, isto é, a colaboração entre os agentes pode solucionar grande parte destes problemas (LASHKARI *et al.*, 1994). Agentes experientes podem ajudar um agente iniciante a aprender com mais rapidez, assim como auxiliá-lo em situações não previstas anteriormente. Enquanto os agentes de interface devem apresentar *habilidade em aprender* e terem *autonomia*, um agente colaborativo deve possuir *autonomia* e capacidade de *cooperação.*

Os agentes colaborativos prezam pela autonomia e cooperação com outros agentes na execução de tarefas para seus proprietários, em ambientes multi-agentes abertos e sujeitos a limitações de tempo. Eles podem aprender, mas este aspecto não é relevante para sua operação, embora alguns tenham capacidade de aprendizado via parametrização ou memorização (por exemplo, um roteiro é fornecido para ser seguido pelo agente). Para coordenar suas atividades, eles podem ter que negociar para alcançar acordos aceitáveis. Eles podem ser benevolentes, racionais, verdadeiros, adotarem algumas combinações destas características ou nenhuma delas.

A motivação para a implementação deste tipo de agente vem principalmente da necessidade de troca de informação em ambientes heterogêneos, gerados pela evolução da tecnologia de redes de computadores, que produz uma grande variedade de sistemas e bases de dados distribuído e interligados.

Segundo MORREALE (1998), um agente ideal para fazer o gerenciamento de uma rede seria aquele que mesclasse a habilidade de cooperar, extraída de agentes colaborativos, com a habilidade de aprender, extraída de agentes de interface.

Segundo COLLIS *et al*. (1998) alguns dos aspectos essenciais de um agente colaborativo são os seguintes:

- Os agentes devem ser capazes de se comunicar – este é atualmente um problema desafiador a ser solucionado, requerendo não somente algum protocolo comum, mas também uma linguagem de comunicação entre agentes.
- Os agentes devem ser capazes de tomar decisões – quando o controle do recurso em questão (por exemplo, um serviço ou quantia de recursos) é destinado a um agente, ele necessita de habilidade suficiente para decidir quando liberar os

recursos para um potencial usuário ou utilizá-los em busca de um objetivo. Portanto, os agentes colaborativos necessitam de procedimentos especiais para determinar a melhor forma de atender a seus objetivos, dado que seus recursos são limitados.

- Os agentes devem ser capazes de cooperar: como um simples agente possui competência, recursos e informações limitadas, ele será capaz de solucionar apenas uma pequena parte do problema. Portanto, um agente colaborativo, ao se defrontar com um problema, pode não ter capacidade suficiente para solucioná-lo sozinho. Sendo assim, ele deve encontrar outros agentes que possuem as habilidades complementares requisitadas, negociar com eles e principalmente delegar a tarefa a eles. Isto sugere um meio de coordenar o comportamento dos diferentes agentes de tal forma que eles possam agir conjuntamente e de uma maneira coerente.

A.1.3 Agentes de interface

Os agentes de interface enfatizam autonomia e aprendizado na execução das tarefas requisitadas por seus proprietários (GREEN *ET AL.*, 1997; NWANA & NDUMU, 1996). MAES (199'4) aponta que a metáfora chave que explica o agente de interface é a de um assistente pessoal que está colaborando com o usuário no mesmo ambiente de trabalho. Note que existe uma diferença fundamental entre colaborar com o usuário e colaborar com outros agentes, como no caso do agente colaborativo.

Um agente de interface não atua simplesmente como uma interface passiva ou uma camada estática entre o nível do usuário e a aplicação. Ele atua em interação com o usuário. O ponto principal (GREEN *et al.*, 1997) que difere um agente inteligente de interface de uma interface inteligente de usuário é que os agentes são pró-ativos e gozam de um certo grau de autonomia. Seu comportamento, conforme já mencionado acima, se assemelha a de um assistente pessoal que coopera com o usuário em uma tarefa. O usuário pode ignorar a participação do agente, se for necessário.

Segundo MAES (1994), três abordagens para a construção de agentes de interface podem ser citadas:

1. A primeira consiste em usar o programa do usuário final como um agente (agente semi-autônomo). Esta abordagem permite que o usuário defina regras que ditam o comportamento do agente. A vantagem, neste caso, é a confiabilidade, pois o usuário tem o controle total das ações, uma vez que ele próprio definiu as regras, portanto tem alta credibilidade. A desvantagem é a carga de trabalho delegada ao usuário final, uma vez que ele tem que detectar a oportunidade de utilizar um agente e preocupar-se com a manutenção das regras definidas ao longo do tempo.
2. A segunda abordagem é dotar o agente de um mecanismo de conhecimento específico sobre o domínio da aplicação e do usuário. Esta última abordagem é a adotada por alguns estudiosos de inteligência artificial e de interfaces inteligentes para usuários. Neste caso, em tempo de execução, o agente usa seus conhecimentos para reconhecer a intenção do usuário e definir uma forma de contribuir. A desvantagem desta abordagem é que o conhecimento atribuído ao agente é fixo e não pode ser adaptado. Há muito trabalho envolvido na definição do mecanismo de conhecimento, e pode levar a um sentimento de perda de controle por parte do usuário, uma vez que o agente poderá ter sido programado por outra pessoa e o usuário não terá noção das limitações e métodos de trabalho do agente.
3. A terceira abordagem, mais robusta, é a apresentada por MAES (1994) e defendida também por outros pesquisadores, tais como LIEBERMAN (1997). Esta abordagem baseia-se em técnicas de aprendizado de máquina. A idéia é fornecer ao agente um mínimo de conhecimento prévio, e a partir da observação das ações do usuário, ele aprende como atuar. Neste caso, algumas condições devem ser respeitadas, tais como: a aplicação deve envolver um comportamento repetitivo e o trabalho deve ser em essência diferente para diferentes usuários. Se estas condições não forem atendidas, não se justifica o uso deste tipo de agente pois, não havendo regularidade de ações, ele não tem meios para aprender (o aprendizado aqui é visto como um processo de condicionamento) e se o trabalho é o mesmo para todos os usuários, a abordagem de conhecimento fixo pode fornecer melhores resultados. Dentre as vantagens desta abordagem

sobre as demais, destacam-se: requer menos esforço do usuário final, alto grau de confiabilidade por parte do usuário, pois o agente desenvolve gradualmente suas habilidades, baseadas no comportamento do próprio usuário, e permite a transferência de *know-how* entre diferentes usuários do ambiente.

Essencialmente, agentes de interface suportam e proporcionam assistência pró-ativa, tipicamente para assistir o usuário a aprender como usar um aplicativo. O agente observa e monitora a ação tomada pelo usuário na interface, encontra novos atalhos, e sugere caminhos melhores para executar a tarefa. Quanto ao aprendizado, tipicamente, o agente de interface aprende a assistir seu usuário nas quatro formas seguintes:

1. observando e imitando o usuário;
2. através de recebimento de realimentação positiva ou negativa do usuário;
3. recebendo instruções explícitas do usuário;
4. consultando outros agentes para receber conselhos.

A.1.4 Agentes móveis

Recentemente, devido à rápida expansão da Internet, e ao aumento do uso de recursos em redes de computadores, tais como aplicações multimídia, um novo paradigma vem surgindo, sendo denominado por agente móvel. Por definição, agentes móveis são entidades de software autônomas, capazes de viajar através da rede e executar tarefas em nome do usuário. A grande diferença entre este conceito e a visão mais simples de agentes é a mobilidade, a qual permite que ele se mova de um ambiente hospedeiro para outro, carregando consigo seu código e seu estado para outra máquina, sendo capaz de terminar sua execução nesta.

Agentes móveis são softwares (programas) capazes de vasculhar grandes áreas da Internet, interagindo com servidores remotos, colecionando informações em nome de seu proprietário e voltando para sua origem, após ter executado o seu conjunto de serviços. Estes serviços podem envolver, por exemplo, a execução de uma reserva de vôo ou o gerenciamento de uma rede de telecomunicações. Agentes móveis são autônomos e cooperam, embora em uma forma diferente daquela verificada no caso dos agentes colaborativos. Por exemplo, eles podem cooperar ou se comunicar com um outro agente

através da localização de alguns de seus objetos internos e métodos conhecidos por outros agentes.

A hipótese chave para explicar a corrida tecnológica pelo desenvolvimento de agentes móveis é a idéia de que, para certas aplicações, eles oferecem muitas vantagens, não encontradas em equivalentes estáticos. Por exemplo, imagine uma situação em que se tem que realizar uma busca (*download)* entre muitas imagens para posterior seleção; seria menos dispendiosos mandar seu agente se deslocar até o repositório destas imagens, fazer uma pesquisa local e somente transferir as imagens desejadas.

Considerando este novo paradigma, LIERBERMAN (1997) apresentou a seguinte definição de agentes inteligentes: Um agente é uma entidade computacional que:

- Atua em nome de outra entidade de uma forma autônoma;
- Executa suas ações em algum nível de pró-atividade e/ou reflexibilidade;
- Exibe algum nível dentre os atributos básicos de aprendizado, cooperação e/ou mobilidade.

As primeiras abordagens (MORREALE, 1998) de agentes móveis consideravam um programa que podia mover-se, sob seu próprio controle, de máquina a máquina, em um ambiente heterogêneo. O primeiro modelo de agentes móveis foi introduzido em 1995 no sistema denominado *AgentTCL* (Dartmouth College - Hanover). Neste modelo, a execução de um agente é suspensa em qualquer ponto e o seu código e estado transportado para outra máquina, onde é reativada e seu processamento é terminado. Esta caracterização foi aprimorada com o tempo, por pesquisadores da área de sistemas distribuídos, dando menos ênfase ao aspecto de código móvel e mais ao desempenho dos agentes para sistemas de tempo-real. As pesquisas atuais concentram-se no aspecto do código, sem entretanto desprezar o comportamento e a facilidade do uso dos agentes móveis em ambientes distribuídos.

Uma área de pesquisa em que os agentes móveis estão sendo utilizados é a de sistemas de banco de dados distribuídos heterogêneos (LEIBNITZ, 1995). Nesta área, um agente poderia ser útil para:

- tornar possível a criação de associação entre dados localizados em diferentes banco de dados.
- tornar acessível uma interface para trabalhar com muitos banco de dados.

A diferença entre um agente móvel e um agente estático reside na forma como ele se comunica com um serviço remoto na rede. O método de chamada remota a procedimento (*RPC - remote programming call*) permite que programas chamem procedimentos localizados em outras máquinas sem que o programador visualize qualquer operação de troca de mensagens ou de entrada e saída (TANENBAUM, 1995). Neste método, um conjunto de dados é enviado para a outra máquina, que os processa e devolve o resultado. Cada troca de mensagens envolve uma solicitação de serviços ou um aviso de reconhecimento. Neste processo, há a necessidade de uma conexão persistente, o que acarreta uma sobrecarga muito grande na comunicação.

Os agentes móveis utilizam uma forma de programação remota (MORREALE, 1998) que traz vantagens sobre a comunicação baseada em *RPC* tradicional. Neste caso, a um *procedimento* são associados os seus dados ou um estado como um objeto, onde tal objeto é um agente móvel que navega através da rede até chegar ao computador de destino, que o executará. Após a execução, o agente pode voltar ao computador de origem ou pode ser apagado.

Para um agente ser executado em diferentes hospedeiros em uma rede, cada hospedeiro deve fornecer suporte de software especializado. Este software permite que o agente tenha acesso aos recursos do hospedeiro que ele precisa para comunicação, computação e navegação. GREEN *et al.* (1997) apresenta a seguinte definição para um ambiente de agentes móveis: "Um ambiente de agentes móveis é um sistema de software distribuído em uma rede de computadores heterogêneos. Sua tarefa fundamental é prover um ambiente no qual os agentes móveis possam ser executados. Este ambiente também deve prover os seguintes serviços:

- suporte relacionado com o próprio ambiente do agente;
- suporte aos serviços do ambiente onde o agente foi construído;
- suporte para acesso a outros sistemas com agentes móveis;
- suporte para permitir acesso a ambientes de software não baseados em agentes."

A tecnologia de agentes móveis vem sendo desenvolvida já há alguns anos, porém apenas recentemente vêm surgindo alguns sistemas práticos. A maioria deles é construída sobre uma plataforma Java, embora alguns não obedeçam a este critério. Como exemplo de

algumas plataformas, temos o sistema *Aglets* desenvolvido pela IBM do Japão, *TCL/tk* pela Sun Microsystems e *Tacoma* (Cornell University).

Uma motivação importante para o desenvolvimento de agentes móveis é justificada pela necessidade de uso destes em ambientes de tempo real, onde latência ou atrasos de qualquer tipo devem ser evitados. GREEN *et al.* (1997) apresenta diversas vantagens do uso de agentes móveis, sendo as citadas abaixo as mais importantes para justificar sua utilização:

- *Eficiência:* Agentes móveis consomem menos recursos de rede, uma vez que tanto a computação como os dados a serem processados são transferidos e processados na máquina de destino, ao invés de somente os dados a processar.
- *Redução do tráfego na rede*: O número de interações dos computadores da rede diminui, uma vez que estas podem ser feitas localmente.
- *Interação autônoma assíncrona:* Agentes móveis podem operar assincronamente e de forma independente do programa que os enviou.
- *Suporte para ambientes heterogêneos:* Como os agentes móveis são independentes do tipo de computador e da rede, eles apresentam grande portabilidade.

A.1.5 Agentes de Informação/Internet

Os agentes de informação executam o papel de gerenciamento, manipulação ou comparação da informação transmitida entre muitos recursos distribuídos. As razões para o desenvolvimento de agentes de informação são pelo menos duas:

- Primeiramente, há simplesmente uma necessidade de se conseguir ferramentas para gerenciar o aumento expressivo de informação na *Web*. Todos os usuários da *We*b deveriam beneficiar-se dos agentes de informação, da mesma forma como estão beneficiando-se de facilitadores de pesquisa como *Spiders*, *Lycos* ou *Webcrawlers*.
- Em segundo lugar, há grandes benefícios financeiros a serem conseguidos com o emprego de agentes de informação. Por exemplo, a Netscape Corporation cresceu da relativa obscuridade para uma companhia com capital de bilhões de

dólares em um tempo muito curto, simplesmente oferecendo um *browsing* com recursos genéricos.

O principal problema dos agentes de informação é manter atualizadas as classificações realizadas em um ambiente extremamente dinâmico como a *Web*. Por esta razão, e pelas similaridades apresentadas, é possível que a maioria dos futuros agentes de informação sejam variações de agentes móveis. É possível prever que eles serão capazes de navegar no ambiente *Web* e armazenar sua topologia em bancos de dados, ou seja, na sua própria página de acesso à Internet.

Alguns autores acreditam que os agentes de informação são essencialmente (ou similares aos) agentes de interface ou aos agentes móveis, dependendo do agente de informação ser estático ou móvel, respectivamente.

A.1.6 Agentes Híbridos

O debate a respeito de qual destes agentes é o mais adequado para cada aplicação não apresenta perspectivas de convergir para um consenso. Visto que cada tipo de agente tem suas próprias vantagens e desvantagem, todo projeto consistente visa maximizar as vantagens e minimizar as desvantagens das técnicas mais relevantes para seu propósito particular. Freqüentemente, um caminho para fazer isto é adotar uma abordagem híbrida, como a feita por MAES (1991), que traz algumas das vantagens dos paradigmas reflexivo e deliberativo. Em tais casos, o componente reflexivo, que deverá tomar precedência sobre o deliberativo, traz os seguintes benefícios: robustez, tempo rápido de resposta e adaptabilidade. No passo seguinte, a parte deliberativa trata os resultados relacionados com os objetivos em questão. Desta forma, os agentes híbridos referem-se àqueles agentes cuja constituição é uma combinação de duas ou mais filosofias de agentes.

A.1.7 Agentes heterogêneos

Sistemas de agentes heterogêneos referem-se a uma integração estabelecida com pelo menos dois ou mais agentes que pertençam a duas ou mais classes de agentes diferentes. Um sistema de agentes heterogêneos pode também conter agentes híbridos. GENESERETH & KETCHPEL (1994) declaram que a motivação para o uso de sistemas de

agentes heterogêneos, essencialmente, é que existe uma grande diversidade de produtos de software. Embora estes programas trabalhem isolados, há uma crescente demanda por interoperabilidade e de alguma forma essa interoperabilidade agrega um "valor adicional" que não seria possível de outro modo. De fato, foi criado um novo domínio baseado em teoria de agentes no contexto de engenharia de software, para facilitar a interoperação de agentes de software heterogêneos.

Um requerimento importante para interoperabilidade de agentes heterogêneos é a linguagem de comunicação de agentes, através da qual diferentes agentes podem comunicar-se entre si. GENESERETH & KETCHPEL (1994) observam que a engenharia de software é freqüentemente comparada à programação orientada a objeto, na qual um agente, como um objeto, fornece uma interface baseada em mensagens para sua estrutura de dados interna e para seus métodos. Entretanto, eles observam que há uma distinção importante:

- em programação orientada a objetos, o significado da mensagem pode diferir de objeto para objeto;
- em engenharia de software baseada em agentes, os agentes usam uma linguagem comum com uma semântica independente do agente.

A.2 Agentes e conhecimento

O componente associado ao conhecimento dos agentes é algo fundamental na determinação do seu comportamento. É este conhecimento que habilita os agentes a realizarem tarefas como:

- ensinar para as pessoas novas técnicas de programação – agente tutor;
- aprender sobre o calendário da programação habitual dos seus usuários humanos – agente aprendiz;
- usar o senso comum a um determinado grupo de usuários – agente com senso comum;
- derivar ações conhecidas através da percepção do ambiente físico a sua volta – agente físico.

A.2.1 Agentes tutores

SELKER (1984) considera agentes como programas de computador que simulam um relacionamento. Segundo ele, pode haver dois tipos de agentes, a saber:

Um *agente estilo assistente* que constrói um relacionamento com o usuário através de uma interface privada. Usando esta interface, o agente pode entender as necessidades do usuário para executar tarefas anteriormente intratáveis ou de difícil execução, com macros criada pelo computador.

Por outro lado, um *agente estilo consultivo* constrói um relacionamento com o usuário com objetivo explícito de educar o usuário. Por exemplo, SELKER (1984) construiu um agente tutor com estilo consultivo chamado *Cognitive Adaptive Computer Help* (*COACH*) que ajuda os usuários a aprender a programar na linguagem de programação *Lisp*.

A.2.2 Agentes aprendizes

MITCHELL *et al.* (1984), pesquisadores da Universidade de Carnegie Mellon (*CMU*), acreditam que o aprendizado de máquina desempenhará importante papel no futuro de assistentes pessoais de software. Eles imaginam um futuro em que o conhecimento é baseado em assistentes, que "operam através de uma rede, como uma secretária, provendo serviços tais como: pagamento de contas, acordos de viajem, pedidos de compra e localização de informações em bibliotecas eletrônicas" (MITCHELL *et al.,* 1984).

O sucesso destes agentes dependerá do conhecimento e aprendizado a respeito dos hábitos e objetivos de um usuário particular. Os pesquisadores da *CMU* construíram um administrador de calendário chamado *Calendar Apprentice* (*CAP*), que aprende o escalonamento das atividades habituais preferencias do usuário através da experiência.

A.2.3 Agentes com senso comum

LENAT *et al*. (1990) acreditam que os agentes necessitam de alguma base comum de conhecimento compartilhada para a finalidade de comunicação. Segundo ele, os últimos 20 anos têm presenciado numerosas implementações bem-sucedidas de sistemas baseados em conhecimento. Ao lado deste sucesso, observa-se também fracassos recorrentes,

principalmente porque estes sistemas não compartilham conhecimento e especialidades, e não trabalham unidos em sinergia. Em outras palavras, estes sistemas apresentam desempenho insatisfatório diante de situações não previstas.

De acordo com LENAT *et al*. (1990), o principal empecilho para a realização de agentes com comportamento satisfatório está na escassez de conhecimento no qual eles se baseiam para operar. Ele argumenta que as pessoas não deveriam trabalhar arduamente para produzir algoritmos, estruturas de dados e arquiteturas inteligentes se houver um rico banco de dados de conhecimento disponível para ser explorado.

LENAT é pioneiro na tentativa de agrupar uma grande base de conhecimento (na ordem de 10 milhões de axiomas) cobrindo o conhecimento de consenso humano (LENAT *et al*., 1990; GUHA, 1994). Na sua opinião, atualmente há duas metodologias de construção de agentes de software, que são as seguintes:

- Na primeira, a competência emerge de um grande número de agentes relativamente simples, integrados por alguma arquitetura cuidadosamente planejada. Um exemplo deste paradigma é o *SOAR* (LAIRD *et al*., 1987), cujo precursor foi o sistema de produção *OPS5* (BROWNSTON *et al*., 1985);
- Na segunda, a competência emerge de um agregado de sistemas processando uma grande quantidade de conhecimento útil. Para tarefas de mundo real, esta envolve grande quantidade de conhecimento, denominado conhecimento de senso comum. Neste paradigma, a arquitetura não é importante. O arquétipo deste paradigma é o projeto *Cyc*, e seu precursor foi o conceito de sistema especialista, muito difundido nos anos 70.

O projeto *Cyc* objetiva testar a segunda metodologia de modelagem para agentes de software. Muitas das partes constituintes do conhecimento de senso comum incluem noções simples de tempo, espaço, causalidade, e eventos; capacidades humanas, limitações, objetivos, estratégias de tomada de decisão, emoção; familiaridade com arte, história, literatura, assuntos correntes, e assim por diante.

A.2.4 Agentes físicos

BROOKS (1990, 1991) acredita numa abordagem de inteligência incremental, que depende estritamente de interfaces robóticas para percepção e ação no ambiente. Ele propôs a "hipótese dos fundamentos físicos" que declara que, para construir um sistema que é inteligente, é necessário ter sua representação baseada diretamente no mundo físico. Ele observa que o mundo real é seu melhor modelo. Em outras palavras, o mundo real é sempre atual e contém todos os detalhes a serem conhecidos.

A noção tradicional de sistema inteligente, influenciada pelos pesquisadores de Inteligência Artificial, tem sido aquela baseada no sistema central, com módulos de percepção como entrada e módulo de ação como saída. A metodologia tradicional decompõe a inteligência em unidades funcionais, que combinadas fornecem o comportamento global do sistema.

BROOKS (1990, 1991) argumenta que "a inteligência humana é complexa demais e muito pouco compreendida para ser decomposta corretamente em partes. Mesmo se as partes são conhecidas, ainda não se conhece a interface correta para elas". Ele prefere uma decomposição alternativa de sistemas inteligentes em direção a módulos gerando comportamento, cada um dos quais individualmente conectando percepção à ação, sem passar por um processador de informação central. A vantagem desta abordagem é que ela oferece um caminho incremental muito simples para sistemas autônomos inteligentes complexos. Além disso, a coexistência e cooperação entre estes conjuntos de módulos de geração de comportamento conduz a um estágio para a geração de comportamentos cada vez mais complexos.

A.3 Características de um agente inteligente

Conforme já mencionado, cada autor considera algumas características necessárias para seu agente em particular. Abaixo, relacionamos as principais características encontradas na literatura:

- *Independência* - um agente deve continuar funcionando mesmo depois que seu agenciador não estiver mais presente;

- *Adaptabilidade* - um agente deve poder adaptar-se a múltiplos ambientes, envolvendo múltiplas plataformas e sistemas;
- *Rastreabilidade* - apesar de independentes, os agentes devem poder ser monitorados;
- *Robustez* - agentes devem ser capazes de lidar com erros, recursos escassos e dados incompletos;
- *Auto-Gerenciabilidade* - agentes devem ser capazes de realizar a gestão de seu próprio ciclo de vida, ou seja, iniciar e cessar suas ações de acordo com critérios próprios;
- *Focalização nos interesses do usuário* - um agente deve atuar no intuito de considerar os interesses do usuário, antes de mais nada;
- *Reflexibilidade* - um agente deve responder prontamente às mudanças no ambiente;
- *Autonomia* - um agente deve exercer um controle sobre suas próprias ações;
- *Pró-Ativismo* - um agente deve ter propósitos, ou seja, deve ser orientado a metas;
- *Continuidade* - um agente deve ser um processo que roda continuamente, ou seja, tem um ciclo de vida ininterrupto;
- *Comunicabilidade* - um agente deve poder comunicar-se com outros agentes, incluindo seres humanos;
- *Aprendizagem* - um agente deve utilizar suas experiências prévias para aprender e adaptar-se às mudanças no ambiente;
- *Flexibilidade e Mobilidade* - um agente deve ser capaz de deslocar-se de um ambiente para outro e mover-se dentro de um mesmo ambiente;
- *Personalidade* - um agente deve ter individualidade, ou seja, capacidade de diferenciar-se de outros agentes com propriedades comuns;
- *Emocionabilidade* - um agente deve poder exibir "estados emocionais" que caracterizem seu estado diante das metas que visa cumprir e do estado atual do ambiente;
- *Credibilidade (Believability)* - um agente deve poder proporcionar a ilusão de ser um ser humano, com o qual o usuário se comunica.

A.3.1 Requisitos Mínimos

Com o intuito de diferenciar agentes de objetos, ou mesmo de simples programas computacionais, muitos pesquisadores definem alguns requisitos mínimos necessários a um agente, a saber:

- ***Ambiente*** - todo agente deve ter um ambiente para que possa perceber e agir sobre o mesmo.
- ***Sensores e Atuadores*** – necessários para que o agente possa coletar dados do ambiente e atuar sobre ele.
- ***Ciclo de Vida*** - todo agente deve possuir um ciclo de vida, no qual passos de percepção e ação são executados continuamente.
- ***Objetivo(s)*** – todo agente deve ser criado com algum objetivo, o qual gerencia o comportamento do agente

A.4 Tipos de agentes inteligentes

Vários são os nomes e classificações dadas para agentes inteligentes. A classificação apresentada nesta dissertação não é única e a nomenclatura utilizada pode variar, dependendo do autor.

A.4.1 Agentes reflexivos

Possuem um processo único de percepção-ação (reflexo). Ação é uma função *f* direta das entradas dos sensores, conforme apresentado na figura A.2. Esta função *f* pode ser qualquer, por exemplo: conjunto de regras condição-ação (fuzzy ou binárias), rede neural não-recorrente, ou uma função matemática.

Para atingir os objetivos, deve-se fazer uma boa escolha para a função *f,* de modo que os objetivos estejam implícitos. O projeto da função *f* pode ser realizado através das seguintes abordagens:

- heurística - algoritmo heurístico (conhecimento implícito), lógica fuzzy - (conhecimento explícito);
- aprendizagem - indutiva (redes neurais), evolutiva (algoritmos genéticos);

- via algoritmos de otimização (estimativa de parâmetros);
- via mecanismos de busca alternativos.

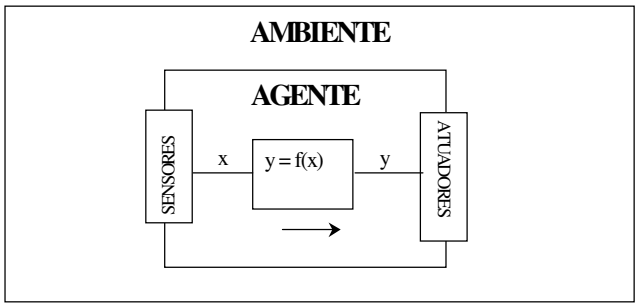


Figura A.2 - Agente Reflexivo

A.4.2 Agentes Comportamentais

Possuem processos de percepção e ação. A ação é um processo independente, controlado pelo módulo de percepção. Além disso, possuem um conjunto pré-definido de comportamentos, os quais são selecionados dependendo da percepção, conforme ilustrado na figura A.3. Desta forma, apresenta uma dinâmica de comportamento geralmente sofisticada, mas com limites em sua aplicabilidade (incapacidade de afastar-se do seu objetivo e retornar posteriormente). Depende de uma especificação adequada do conjunto de comportamentos disponíveis.

O mecanismo de ação pode corresponder a diversos mecanismos reflexivos em paralelo, sendo que apenas um deles está operando em um determinado instante.

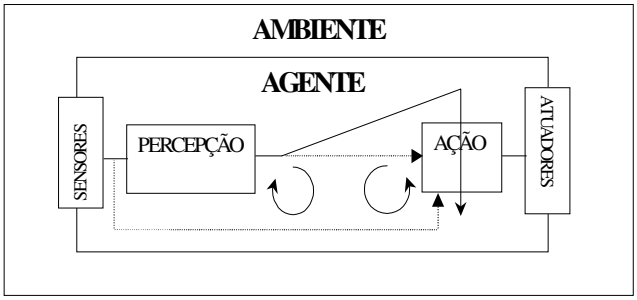


Figura A.3 - Agente Comportamental

A.4.3 Agentes planejadores

Possuem processos independentes de percepção e ação, conforme apresentado na figura A.4. Existe um modelo do ambiente (modelo do mundo)**.** A ação não é gerada diretamente pela percepção, mas por um mecanismo de geração de comportamento**.**

O modelo do ambiente pode ser conhecido a priori, e o mecanismo de percepção apenas "situa" o agente no contexto, ou pode ser aprendido por meio da interação com o ambiente através de mecanismos de percepção, que conduzem à elaboração de uma representação das partes que o constituem, e situa o agente em seu interior. O modelo do mundo pode ser constituído por duas partes: física do mundo (interação dinâmica) e história do mundo (Passado e Presente, Modelo Episódico ou Memória Episódica).

O módulo de geração de comportamento utiliza o modelo do ambiente para fazer predições de como o ambiente se comportará, ao menos localmente, diante de possíveis ações tomadas pelo agente. Funciona, neste caso, como um gerador de predições (preditor).

O conjunto de ações a serem tomadas pelo agente é chamado de plano. Quando o agente necessitar de múltiplos planos diferentes ele terá um gerador de planos. Para avaliar estes planos, o módulo de geração de comportamento utilizará o gerador de predições para determinar o resultado da realização de vários planos diferentes, determinados pelo gerador de planos. O plano que melhor satisfizer os objetivos do sistema será efetivamente executado. Três situações poderão ocorrer: objetivo não é atingido, objetivo é atingido, e objetivo é atingido com o melhor desempenho.

Uma medida de utilidade dá uma "nota" para um plano e, além de informar se um objetivo é cumprido, avalia o quanto ele está sendo cumprido. Permite a seleção de planos quando existirem múltiplos planos cumprindo um mesmo objetivo, com desempenhos diferentes.

Durante a execução do plano, alguns problemas poderão surgir, como por exemplo:

- Tempo de execução do plano é muito demorado;
- Mudanças no modelo do ambiente, devido a mudanças do próprio ambiente ou devido a mudanças no conhecimento sobre ele;
- Planos podem se tornar obsoletos diante de novas condições: eles não mais satisfazem os objetivos;

- Planos podem diminuir de qualidade diante de novas condições: eles podem não ser mais a melhor opção.

Para solucionar tais problemas, normalmente realiza-se o replanejamento, isto é, aborta-se a execução do plano e retornar-se ao processo de planejamento.

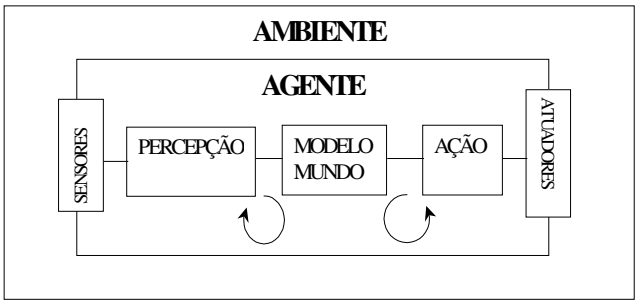


Figura A.4 - Agente Planejador

A.4.4 Agentes Emocionais

A função de utilidade ou medida de utilidade, conforme definida na seção anterior pode ser insuficiente para a avaliação dos planos. Surge então a necessidade do emprego de um sistema de valores baseado em emoções. A figura A.5 mostra a estrutura de um agente emocional.

As emoções podem ser:

- avaliações internas que medem se os objetivos do agente estão sendo cumpridos a contento;
- mecanismos de influência no planejamento de futuras ações.

Além disso, estas emoções podem ser:

- externalizadas por meio de atuações na aparência do agente diante do ambiente;
- utilizadas como uma forma de comunicação entre agentes.

Diferentes tipos de emoções podem ser modeladas: medo, desejo, dor, alegria. As emoções podem ser atribuídas a estados atuais ou a previsões de estados futuros.

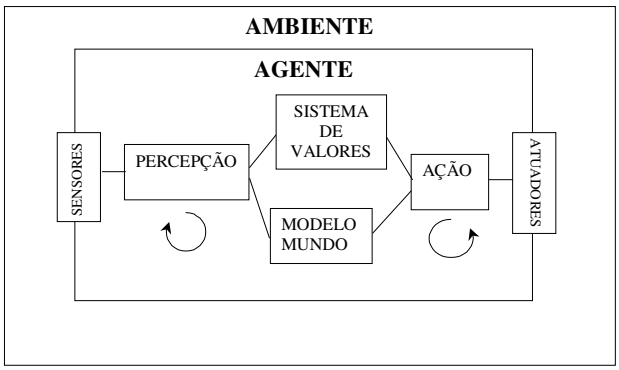


Figura A.5 - Agente Emocional

A.4.5 Agentes comunicativos

Possuem um canal de comunicação direta. Desta forma, necessitam de uma linguagem de agentes. Representam as unidades básicas em Sistemas Multi-Agentes.

A especialização de tarefas é realizada através de uma Coordenação Central (Delegação Subordinada) ou uma Coordenação Distribuída (Delegação Participativa). A coordenação central (Delegação Subordinada) pode ser dividida em: coordenadores e executores.

A coordenação distribuída (Delegação Participativa) pode ser realizada em função dos seguintes critérios:

- Colaboração, cooperação e competição;
- Localização e vizinhança determinam o comportamento;
- Comportamento Emergente;
- Papéis Comportamentais - todos os agentes são iguais, mas dependendo da sua posição e situação, assumem papéis distintos.

A figura A.6 mostra a estrutura de um agente comunicativo.

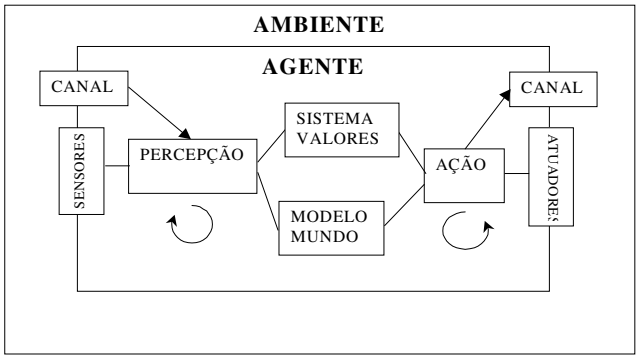


Figura A.6 Agente Comunicativo

A.4.6 Agentes semióticos

Não necessitam de canal de comunicação, pois o próprio ambiente funciona como canal. Assim, o mecanismo de percepção e ação é mais sofisticado em relação aos agentes anteriores. Sua concepção é baseada em alguns conceitos de semiótica computacional (GUDWIN, 1996). SOUZA E SILVA (1998) apresenta uma visão teórica deste agente.

A figura A.7 apresenta a estrutura básica de um agente semiótico.

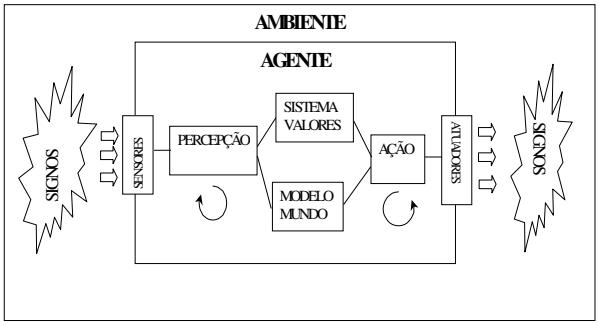


Figura A.7 Agente Semiótico

A.5 Sistemas Multi-agentes

Em sistema de software multi-agentes (NWANA & NDUMU, 1996), os agentes geralmente cooperam e/ou competem uns com os outros para executar uma tarefa específica. Tipicamente, estes agentes estão distribuídos topologicamente através de uma

rede de área local ou não, com os agentes individuais processando uma quantidade limitada de informações ou dados, com limitado poder de processamento e capacidade de solução de problemas. Na situação de cooperação ou mesmo na competição, um grupo de agentes pode necessitar colaborar com os outros para executar tarefas mais complexas ou mesmo disputar por recursos escassos. A razão da cooperação reside no fato de que cada agente possui limitada competência, informação ou recursos disponíveis, e assim, unindo suas habilidades de maneira coerente, são capazes de solucionar problemas de alta complexidade, o que talvez não fosse possível com apenas um agente (NDUMU *et al*., 1999). Geralmente, tal colaboração e competição requer um controle sofisticado da comunicação e das trocas de mensagens de requisição e informação entre os agentes, com a finalidade de assegurar seu funcionamento coerente.

Inteligência Artificial Distribuída é uma sub-área da Inteligência Artificial que investiga modelos de conhecimento (GREEN *et al.*, 1997), assim como técnicas que permitem que agentes computacionais participem em sociedades compostas por computadores e pessoas. Uma sociedade assim é denominada Sistema Multi-Agentes (*MAS – Multi-Agents System*). Estes sistemas multi-agentes são compostos de agentes individuais que trabalham juntos para resolver problemas complexos, como por exemplo em um sistema de apoio à tomada de decisão, em que um diagnóstico final pode ser gerado, baseado no diagnóstico individual feito por iniciativa de cada agente. Tipicamente, estes agentes, apesar de estarem distribuídos pela rede, são estáticos, isto é, permanecem em uma só máquina. Seu projeto geralmente é baseado nas características de coordenação, negociação e comunicação.

GREEN *et al.* (1997) distinguem dois tipos de sistemas multi-agentes baseado no grau de cooperação entre os agentes:

- Sistemas Cooperativos Multi-Agentes (*CMAS – Cooperative Multi-Agents Systems*): compreende sistemas em que os agentes que interagem foram programados pelo mesmo projetista, visando portanto o melhor aproveitamento do sistema, não se preocupando demasiadamente com o desempenho de cada agente em particular.

- Sistemas Multi-Agentes com Interesses Próprios (*SMAS - Self-Interested Multi-Agents System*): consiste de sistemas em que os agentes foram programados por projetistas diferentes, visando primordialmente não o benefício da cooperação, mas os benefícios derivados de sua atuação individual.

O problema neste segundo tipo de sistema é que os agentes nem sempre conseguem exercitar a propriedade de benevolência, isto é, a situação de total cooperação, pois podem ocorrer conflitos entre objetivos (objetivo individual × objetivo do grupo). Um exemplo disto é o agendamento de reuniões. Cada agente tentará esquematizar a reunião no melhor horário particular. O problema em otimizar o desempenho de sistemas com *SMAS* é denominado Solução do Problema da Distribuição (GREEN *et al.*, 1997), e envolve comunicação, coordenação e negociação entre os agentes.

A colaboração e a comunicação entre os agentes podem ser realizadas de diferentes formas. LASHKARI *et al.* (1994) apresentam duas formas de colaboração baseadas na comunicação:

- *Comunicação* "*baseada em impasse*": é invocada quando um agente em particular tem experiência insuficiente para fazer uma previsão correta. Neste caso, ele pede socorro a outro agente e solicita o procedimento padrão para situações similares;
- *Comunicação exploratória*: é o tipo de comunicação realizada pelo agente para explorar, isto é, descobrir novos agentes que façam previsões mais acertadas do que os até então consultados.

Atualmente, a estratégia de competição entre os agentes num sistema multi-agentes é muito pouco explorada pelos pesquisadores de sistemas multi-agentes. No entanto, em muitos casos a utilização desta estratégia é justificável, principalmente quando a quantidade de recursos disponíveis para todo o sistema é limitada. Através da utilização de cooperação e competição entre os agentes, podem emergir fenômenos interessantes, tal como a auto-organização. Nesta dissertação, estas duas estratégias são exploradas no treinamento de dispositivos neurocomputacionais híbridos.

A.5.1 Linguagens de comunicação entre agentes

Para os agentes colaborarem e se comunicarem entre si, eles devem adotar uma linguagem comum, assim como respeitar um protocolo padrão Caso contrário, a troca de informações pode ser prejudicada pela heterogeneidade de linguagem. Sem dúvida, uma linguagem comum é a base para a troca de informações entre os agentes e, conseqüentemente, para um bom desempenho do sistema multi-agentes.

Segundo GENESERCH & KETCHPEL (1994), existem duas abordagens populares para o projeto desta linguagem: a abordagem *procedural* e a abordagem *declarativa*. A primeira abordagem é baseada na troca de diretivas procedurais. Algumas linguagens, tais como *TCL*, *Apple events* e *Telescript,* são baseadas nesta abordagem. Apesar destas linguagens serem eficientemente executadas, elas têm a desvantagem dos *procedimentos* serem unidirecionais, sendo que muitas vezes os agentes precisam compartilhar informações em ambas as direções.

A segunda abordagem é baseada no conceito de que a comunicação pode ser modelada como troca de comandos de declaração. Como exemplo temos o padrão *ACL* (*Agent Communication Language*), que é composta por três partes: o vocabulário, uma linguagem interna denominada *KIF* (*Knowledge Interchange Format*) e uma linguagem externa chamada *KQML* (*Knowledge Query and Manipulation Language*). Uma visão mais abrangente destas linguagens pode ser encontrada em GENESERCH & KETCHPEL (1994).

Referências Bibliográficas

AGARWAL, M. A systematic classification of neural-network based control. *IEEE Control Systems Magazine*, vol. 17, no. 2, pp. 75-93, 1997.

AGRE, P.E., CHAPMAN, D. Pengi: An Implementation of a Theory of Activity. *Proc 6th National Conf. on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, pp. 268-272, 1987.

AGRE, P.E., ROSENSCHEIN, S.J. (Guest editors) Computational Research on Interaction and Agency. *Artificial Intelligence*, Special Volumes 72 & 73, 1995.

ALBUS, J.S. Outline for a Theory of Intelligence. *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 3, May/June 1991.

ANDERSON, C.W. Q-learning with hidden-unit restarting. In *Advances in Neural Information Processing System* vol. 5. San Mateo, CA: Morgan Kaufmann, 1993.

ANGELINE, P. J., FOGEL, D. B. An Evolutionary Program for the Identification of Dynamical Systems. In *Proceedings of SPIE* (Volume 3077): Application and Science of Artificial Neural Networks III, S. Rogers (ed.), SPIE-The International Society for Optical Engineering, Bellingham, WA. pp. 409-417, 1997

ANTASAKLIS, P.J., Ed. Special issue on neural networks in control systems. *IEEE Control Systems Magazine*, vol. 10, no. 3, April 1990.

ASH, T. Dynamic node creation in backpropagation networks. *Connection Science*, vol. 1, no 4, pp. 365-375, 1989.

ÅSTROM, K.J., WITTENMARK B. **Adaptive Control**. Reading, MA: Addison-Wesley, 1994.

BALDI, P.F, HORNIK, K. Learning in Linear Neural Networks: A survey. *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 837-858, 1995

BARRON, A.R., BARRON, R.L. Statistical learning networks: a unifying view. In Wegman, E. J., Gantz, D.T, Miller, J.J. (Eds.). **Computing Science and Statistics**, 1988.

BARTO, A. R. Connectionist learning for control: An overview. In T. Miller, R.S. Sutton, P.J. Werbos (eds) **Neural Networks for Control**. Cambridge: M.I.T. Press, 1990.

BAKSHI, B.R., UTOJO, U. A common framework for the unification of neural and statistical modeling methods. *Analytica Chimica Acta* ,1998.

BAKSHI, B. R., UTOJO, U. Unification of neural and statistical modeling methods that combine inputs by linear projection. *Computers and Chemical Engineering*, vol. 22, no. 12, pp. 1859-1878, 1998. Published by Elsevier Science, Ltda.

BATES, J. The role of Emotion in Believable Characters. *Comunications of the ACM*, vol. 37, no. 7, pp. 122-125, 1994.

BAUM, E., HAUSSLER, D. What size nets gives valid generalization? Neural Computation, vol. 1, no. 1, pp. 151-160, 1989.

BERTHIER, N. E., SINGH, S. P., BARTO, A. G., HOUK, J. C. Distributed representation of limb motor programs in array of adjustable pattern generators. *Journal of Cognitive Neuroscience* vol. 5, no. 1, pp. 56-78, 1993.

BERTRAND, D.J.S.R. Neural Networks controllers for the X29 aircraft. In *IEEE/INNS Int. Conf. on Neural Networks*, Baltimore, MD, 1992.

BEYER, U., ŚMIEJA, F.J. Lerning from examples, agents teams and the concept of reflection. *Gesellschaft für Mathematik und Datenverarbeitung*, St. Augustin, Germany, Tech. Rep. 766, July 1993.

BILLINGS, S. A. Identification of Nonlinear systems – a survey. *IEE Proceedings*, vol. 126, no. 6, pp.272-305, 1980.

BILLINGS, S.A., VOON, W.S.F. A prediction-error and stepwise-regression estimation algorithm for non-linear systems. *International Journal of Control*, vol. 44, no. 1, pp. 803-822, 1986.

BILLINGS, S.A., CHEN, S. Neural network and system identification. In K. Warwick et al.(Eds.), **Neural networks for systems and control. London**, Peter Peregrinus, pp. 181-205, 1992.

BILLINGS, S.A., ZHENG, G.L., Radial Basis Function Networks Configuration Using Genetic Algorithms. *Neural Networks*, vol. 8, no. 6, pp. 877-890, 1995.

BINGULAC, S., VANLANDINGHAM, H.F. Algorithms for Computer-Aided. *Design of Multivariable Control Systems*. New York, Marcel Dekker, Inc., 1993.

BISHOP, C. Exact Calculation of the Hessian Matrix for the Multilayer Perceptron. *Neural Computation*, vol. 4, no. 4, pp. 494-501, 1992.

BROOKS, R. A. A Robust Layered Control System for a Mobile Robot. IEEE Journal Robotics and Automation, vol. 2, no. 1, pp. 14-23, 1986.

BROOKS, R.A. Elephants don't play chess. **In Designing Autonomus Agents**, Pattie Maes Ed. pp. 3-15, MIT Press, Cambridge, Mass., 1990.

BROOKS, R. A. Intelligence without Representation, Artificial Intelligence 47, pp. 139-159, 1991.

BROOMHEAD, D.S., LOWE, D. Multivariable functional interpolation and adaptive networks. *Complex Systems*., vol. 2, pp. 321-355, 1988

BROWNSTON, L., FARRELL, R., KANT, E., MARTIN, N. Programming Expert Systems in OPS5: **An Introduction to Rule-Based Programming**. Reading, Mass., Addison-Wesley, 1985.

BRUSTOLONI, J.C. Autonomous Agents: Characterization and Requirements. *Carnegie Mellon Technical Report CMU-CS-91-204*, Pittsburgh, Carnegie Mellon University, 1991.

CABRERA, J.B.D., NARENDRA, K.S. Linearization of nonlinear control systems: a tutorial, Part I: state variables accessible. Technical report no. 9310, Center for Systems Science, Yale University, New Haven, CT, October, 1993.

CARMON, A. Applying self-tuning control to plant. *Control and Instrumentation*, vol. 18, pp. 81-83, 1986.

CHAUVIN, Y. A backpropagation algorithm with optimal use of hidden units. In *Advances in Neural Information Processing Systems*, vol. 1, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 519-526.

CHEN, S. & BILLINGS, S. A. Recursive prediction error parameter estimation for non-linear models. *International Journal of Control*, vol. 49, no. 2, pp. 569-594, 1989.

CHEN, S., BILLINGS, S.A., GRANT, P.M. Nonlinear system identification using neural network. *International Journal of Control*, vol. 51, no. 6, pp. 1191-1214, 1990.

CHEN, S., BILLINGS, A. Neural networks for nonlinear dynamic system modelling identification. *International Journal of Control*, vol. 56, no. 2, pp. 319-346, 1992.

CHEN, S., WIGGER, J.A fast orthogonal least square algorithm for efficient subset model selection. *IEEE Transactions on Signal Processing*, vol. 43, no. 7, pp. 1713-1715, 1995.

CHEN, V. C., PAO, Y. H. Learning control with neural networks. In *Proc. of 1989 IEEE Conf. on Robotics & Automation*, pp. 1448-1453, 1989.

CHO, H.-J., OH, S.-Y., CHOI, D.-H. Hybrid evolutionary programming technique using subpopulations with completely different evolution mechanisms. Electronics Letters, vol. 34, no. 10, pp. 964-966, 1998.

CHOI, J.Y, VANLANDINGHAM, H. F., BINGULAC, S. A constructive approach for nonlinear system identification using multilayer perceptrons. *IEEE Transactions on Systems, Man, and Cybernetic* - part b, vol. 26, no. 2, april 1996.

CHOI, J.Y., TRAN, M., HUYUNH, P., VANDINGHAM, H. F. Optimal control of nonlinear systems using neural networks. In *Proc. 1992 Southeastern Symp. System Theory*. North Carolina A&T University, Greensboo.

CHOI, J.Y., VANLANDINGHAM, H. F. Nonlinear system identification using neural networks. *International Fuzzy Systems Association (IFSA)*, Seoul, Korea, July 1993.

CLARKE, D.W., MOHTADI, C., TUFFS, P.S. Generalized predictive control-i. the basic algorithm. *Automática* vol. 23, pp. 137-148, 1987.

CLEEREMANS, A., SERVAN-SCHREIBER, D. MCCLELLAND, J. Finite state automata and simple recurrent networks. *Neural Computation*, vol. 1, no. 3, pp. 372-381, 1989.

COLLIS, J.C, NDUMU, D.T., NWANA, H.S., LEE, L. C. The ZEUS agent building toolkit. *BT Technology Journal*, vol. 3, no. 16, 1998.

CYBENKO, G. Continuous valued neural networks with two hidden layers are sufficient. *Technical Report.* Department of computer science, Tuffs University. Medford, MA, 1989a.

CYBENKO, G. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signal and Systems*, vol. 2, pp. 303-314, 1989b.

DAHMEN, W., MICCHELLI, C.A. Some remarks on ridge functions. *Approximation Theory and its Applications*, vol. 3, no. 2-3, pp. 139-143,1987.

DE JONG, S., FAREBROTHER, R. W. Extending the relationship between ridge regression and continuum regression. C*hemometrics and Intelligent Laboratory Systems*, vol. 25, no 2, pp. 179-181, 1994.

DIAGONIS P. Discussion, *in* Huber P.J. Projection pursuit. *The Annals of Statistics*, vol. 13, no. 2, pp. 494-496, 1985.

DOS SANTOS, E.P., VON ZUBEN, F.J. Efficient Second-Order Learning Algorithms for Discrete-Time Recurrent Neural Networks. *in* L.R. Medsker and L.C. Jain (eds.) **Recurrent Neural Networks: Design and Applications**, CRC Press, chapter 3, pp. 47-75, 2000.

DOYLE, J.C., GLOVER, K. KHARGONEKAR, P.P., FRANCIS, B. A. State-Space Solutions to Standard H_2 and H_∞ Control Problems. *IEEE Transactions on Automatic Control*, vol. 34, no. 8, pp. 831-847, 1989.

ELMAN, J.L. Finding structure in time. *Cognitive Science*, vol. 14, no. 2, pp. 179-211, 1990.

FABRI, S., KADIRKAMANATHAN, V. Dynamic Structure Neural Networks for stable adaptive control of nonlinear systems. *IEEE Transactions on System, Man, and Cybernetic*, vol. 7, no. 5, Sept 1996.

FAHLMAN, S. E, LEBIERE, C. The cascaded-correlation learning architecture. In *Advance in Neural Information Processing Systems*, vol. 2, pp. 524-532. Morgan Kaufmann, Los Altos, CA, 1990.

FAHLMAN, S. E., LEBIERE, C. The cascade-correlation learning architectures. In *Advance in Neural Information Processing Systems*, vol. 2, D. S. Touretzky, Ed. Los Altos, CA: Morgan Kaufmann, pp.524-532, 1990

FAHLMAN, S.E., LEBIERE, C. The cascade-correlation learning architecture, in *Advances in Neural Information Processing System* vol. 2, D. Touretzy, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 524-532.

FARLOW, S. J. The GMDH-algorithm. In S.J. Farlow (ed.). **Self-Organizing Methods in Modelling: GMDH Type Algorithms**. Marcel Dekker, New York, pp. 87-104, 1984.

FERBER, J. Simulating with Reactive Agents. In Hillebrand, E. & Stender, J. (Eds.) **Many Agent Simulation and Artificial Life**, Amsterdam, IOS Press, 8-28, 1994.

FOGEL, D.B. Evolutionary programming for neurocontrol. *IEEE expert,* pp. 24-27, June 1995.

FONSECA, C. M., MENDES, E. M., FLEMING, P. J., BILLINGS, S. A. Non-linear model term selection with genetic algorithms. *Proceedings of IEE/IEEE Workshop on Natural Algorithms in Signal Processing,* vol. 2, pp. 271-278, Essex, 1993.

FRANK, I.E. A nonlinear PLS model. *Chemom. Intel. Lab. Systems*, vol. 8, pp. 109-119, 1990.

FRANK, I.E. Modern nonlinear regression methods. *Chemom. Intell. Lab. System*, vol. 27, pp. 1-9, 1995.

FRANKLIN, S, GRAESSER, A. Is it an Agent, or just a Program? - A Taxonomy for Autonomous Agents. *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages,* Springer-Verlag, 1996.

FRANKLIN, S. **Artificial Minds.** Cambridge, MA: MIT Press, September 1995.

FRIEDMAN, J. H, STUTZLE, W. Projection Pursuit Regression. *Journal of the American Statistical Association (JASA)*, vol. 76, no. 376, pp. 817-823, 1981.

FRIEDMAN, J. H, TUKEY, J. A projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Transactions on Computers*, vol. 23, no. 9, pp. 881-890, 1974.

FRIEDMAN, J.H. A variable span smoother. Technical Report no. 5, Dept. of Statistics, Stanford University, Stanford, CA, 1984.

FU, F.C., FARISON, J.B. On the volterra-series functional identification of nonlinear discrete-time system. *International Journal Control*, vol. 18, no. 6, pp. 1281-1289, 1973.

FUNG, C., BILLINGS, S.A., LUO, W. On line supervised adaptive training using radial basis function neural networks. *Neural Networks*, vol. 9, no. 9, pp. 1597-1617, 1996.

GEMAN, S., BIENENSTOCK, E., DOURSAT, R. Neural networks and the bias/variance dilemma. *Neural Computation*, vol. 4, pp. 1-58, 1992.

GENESERETH, M.R., KETCHPEL, S.P. Software Agents. *Comunications of the ACM*, vol 7, no. 37, pp. 48-53, 1994.

GILBERT, D., APARICIO, M., ATKINSON, B., BRADY, S., CICCARINO, J, GROSOF, B., O'CONNOR, P., OSISEK,D., PRITKO, S., SPAGNA, R., WILSON, L. Ibm intelligent agent strategy. White paper, 1995. http://activist.gpl.ibm.com.81/WhitePaper/ptc2.htm

GILES, C., MILLER, C., CHEN, D., CHEN, H., SUN, G., LEE, Y. Leaning and extracting finite state automata with second-order recurrent neural network. *Neural Computation*, vol. 4, no. 3, pp. 393-405, 1992.

GOMI, H., KAWATO, M. Neural networks for a closep-loop system using feedback-error-control. *Neural Networks*, vol. 6, no. 7, pp 933-946, 1993.

GOMM, J.B., EVANS, J.T., WILLIAMS, D. Development and performance of a neural-network predictive controller. *Control Engineering Practice*, vol. 5, no. 1, pp. 49-59, 1997.

GONÇALVES, R., VON ZUBEN, F.J., GOMIDE, F. Using constructive learning in embedded systems engineering. *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 1, pp. 251-255, May 1998.

GOODWIN, G. C., SIN, K. S. **Adaptative filtering prediction and control**. New Jersey, Prentice-Hall, Inc.1984.

GORINEVKY, D.M. Modelling of direct motor program learning in fast human arm motions. *Biological Cybernetics*, vol. 69, pp. 219-228, 1993.

GREEN, S., HURST L., NANGLE, B., CUNNINGLAM, P., SOMERS, F., EVANS, R.. Software Agents: A Review. *Technical report*. Trinity Collega, Dublin, Ireland, May 1997. http://www.cs.tcd.ie/research_groups/aig/iag/pubreview.ps.gz

GROSSBERG, S. A theory of visual coding, memory and development. In *Formal Theories of Visual Perception*. Wiley, New York, 1978.

GROSSBERG, S. Contour enchancement, short term memory and constancies in reverberating neural networks. *Studies in Aplied Mathematics*, vol. 52, pp. 217-257, 1973.

GUDWIN, R. Notas de aula da disciplina sobre Agentes. *Departamento de Engenharia Computação Industrial, Faculdade de Engenharia Elétrica*, Unicamp, 1999.

GUHA, R. V., LENAT, D. B. Enabling agents to work together. *Comunication of the ACM*, vol. 37, no. 7, pp. 126-142, 1994.

HAARIO, H, TAAVITSAINEN, V.-M. Nonlinear data analysis. II. Example of new link functions and optimization aspects. *Chemom. Intell. Lab. Systems*, vol. 23, pp. 51-64, 1994.

HABER, H., UNBEHAUEN, H. Structure identification of nonlinear dynamic system – A survey on input/output approaches. *Automatica*, vol. 26, no. 4, pp. 651-677, 1990.

HANSON, S.J., PRATT, L.Y. Comparing biases for minimal network construction with backpropagation. In *Advances in Neural Information Processing Systems*, vol. 1, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 177-185.

HAPPEL, B.L.M., MURRE, J.M.J. Design and evolution of modular neural network architectures. *Neural Networks*, vol. 7, pp. 985-1004, 1994.

HÄRDLE, W. **Applied Nonparametric Regression**. Cambridge University Press, 1990.

HARTMAN, E., KEELER, J.D. Predicting the future: Advantages of semilocal units. *Neural Computation*, vol. 3, pp. 566-578, 1991.

HASSIBI, B., STORK, D. G. Second order derivatives for network pruning: optimal brain surgeon. In *Advances in Neural Information Processing Systems* vol. 5, pp. 164-171. Morgam, Kaufmann, San Mateo, CA, 1993.

HASTIE, T.J, TIBSHIRANI, R.J. Generalized additive models. In *Monographs on Statistics Applied Probability* 43. New York, Chapman and Hall, 1990.

HAYES-ROTH, B. An Architecture for Adaptive Intelligent Systems. *Artificial Intelligence: Special Issue on Agents and Interactivity*, pp. 72, 329-365, 1995.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. 2nd edition, Prentice-Hall, 1999.

HECHT NIELSEN, R. Theory of the backpropagation neural networks. *In Proc. 1st Int. Joint Conf. Neural Networks, Washington*, D.C., San Diego, 1989.

HECHT-NIELSEN, R. **Neurocomputing**. Addison-Wesley Publishing Company, 1991.

HERMANS, B. Intelligent Software Agents on the Internet. In *Journal First-Monday*, 1997. http://www.firstmonday.dk/issues/issue2-3/ch-123

HIROSE, Y.,YAMASHITA, K., HIJIYA, S. Backpropagation algorithm which varies the number of hidden units. *Neural Networks*, vol. 4, pp. 61-66, 1991.

HOEHFELD, M., FAHLMAN, S.E. Learning with limited numerical precision using the cascade-correlation algorithm. *School Comput. Sci.*, Carnegie Mellon Univ. Tech Rep. CMU-CS-91-130, May 1991

HOLCOMB, T.R., MORARI, M. Local training for radial basis function networks: towards solving the hidden unit problem. *Proc. American Control Conf.*, pp. 2331-2336, 1991.

HOLCOMB, T. R., MORARI, M. PLS/Neural Networks. *Computers in Chemical Engineering*, vol 16, no. 4, pp. 393-411, 1992.

HOPFIELD, J.J. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences of the U.S.A.*, vol. 79, pp. 2554-2558, 1982.

HORNIK, K. Some new results on neural network approximation. *Neural Networks*, vol. 6, no. 8, pp. 1069-1072, 1993.

HORNIK, K., STINCHCOMBE, M., WHITE, H. Universal approximation of na unknown function and its derivatives using multilayer feedforward networks. *Neural Networks*, vol. 3, pp. 551-560, 1990.

HORNIK, STINCHCOMBE, M., WHITE, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, vol. 2, pp. 359-366, 1989.

HRYCERJ, T. A modular architecture for efficient learnnig. In *Proc. in Joint Conf. Neural Networks*, vol. 1, 1990, pp. 557-567.
http://activist.gpl.ibm.com.81/WhitePaper/ptc2.htm

HUBER, P.J. Projection pusuit (with Discussion). *The Annals of Statistics*, vol. 13, no. 2, pp. 435-475, 1985.

HUNT, K.J., SBARBARO, D., ZBIKOWSKI, R.,GAWTHROP, P.J. Neural Networks for Control Systems – A Survey. Automática, vol. 28, no. 6, pp. 1083-1112, 1992.

HWANG, J. N., LI, H., MARTIN, D., SCHIMERT, J. The learning parsimony of projection pursuit and backpropagation networks. In *Conf. Record 25th Asilomar Cong. Signals, Syst., Comput.*, vol. 1, Pacific Grove, Ca, Nov. 1991, pp. 491-495.

HWANG, J.N., LAY, S.R., MAECHLER, M., MARTIN, D., SCHIMERT, J. Regression modeling in backpropagation and project pursuit learning. *IEEE Trans. Neural Networks*, vol. 5, pp. 342-353, May 1994.

HWANG, J.N., LEWIS, P.S. From nonlinear optimization to neural network learning. *In Proc. 24th Asilomar Conference on Signals Systems Computation*. Pacific Grove, CA, pp. 985-989, november 1990.

HWANG, J.N., LI, H., MAECHLER, M., MARTIN, R. D., SCHIMERT, J. A comparison of projection pursuit and neural network regression modeling. In *Advance in Neural Information Processing Systems* vol. 4, J. E. Moody, S. J. Hanson, and R. P. Lipmann, Eds. San Mateo, CA: Morgan Kaufmann, pp. 1159-1166, 1992.

HWANG, J.N., YOU, S.S., LAY, S.R., JOU, I.C. What's wrong with cascaded correlation learning network? A projection pursuit learning perspective. *Technical report*, Department of Electrical Engineering, FT-10, University of Washington, Seattle, WA 98195, 1993.

IGLESIAS, P.A., GLOVER, K. State-space approach to discrete-time H_∞ control. *International Journal of Control*, vol. 54, pp. 1031-1073, 1991.

ISIDORI, A. **Nonlinear Control Systems**, Springer-Verlag, 1989.

IYODA, E.M. **Inteligência Computacional no Projeto Automático de Redes Neurais Híbridas e Redes Neurofuzzy Heterogêneas**. Tese de Mestrado, Faculdade de Engenharia Elétrica e de Computação (FEEC/Unicamp), Janeiro de 2000.

JOE, K., MORI, Y, MIYAKE, S. Construction of large-scale neural network: Simulation of handwriten Japanese character recognition on NCUBE, **Concurrency, Practice, Experience**, vol. 2, no. 2, pp. 79-107, June 1990.

JONES, L.K. A Simple Lemma on Greedy Approximation in Hilbert Space and Convergence Rates for Projection Pursuit Regression and Neural Network Training. *The Annals of Statistics*, vol. 20, no. 1, pp. 608-613, 1992.

JORDAN, M., RUMELHART, D.E. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, pp. 307-354, 1992.

KADIRKAMANATHAN, V., LIU, G.P. Robust identification with neural using multiobjetive criteria. *Preprints of the 5th IFAC Symposium on Adaptive Systems in Control and Signal Processing*, Budapest, Hungary, pp. 237-242, 1995.

KARAKASOGLU, A., SADHARSANAN, S.L., SUNDARESHAN, M.K. Identification and decentralized adaptive control using neural networks with application to robotics manipulators. *IEEE Transactions Neural Networks*, vol. 4, pp. 919-930, 1993

KARNIN, E. D. A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, vol. 1, n.° 2, pp. 239-242, june, 1990.

KAUTZ, H., SELMAN, B., COEN, M. Bottom-up Design of Software Agents. *Communications of the ACM*, vol. 37, no 7, pp. 143-146, 1994.

KAWATO, M. Adaptation and learning in control of voluntary movement by the central nervous system. *Advanced Robotics*, vol. 3, no. 3, pp. 229-249, 1989.

KAWATO, M., FURUKAWA, K., SUZUKI, R. Forward models: Supervised learning with a distal teacher. *Biological Cybernetics*, pp. 169-185, 1987.

KOCHENBURGER, R. J. A Frequency Response Method for Analysis and Synthesis of Conctat Servomechanisms. *Transactions of IAEE*, vol. 69, no. 1, pp. 270-284, 1950.

KOHONEN, T. Self-organized formulation of topologically correct feature maps. *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.

KOHONEN, T. **Self-Organizing Maps (2nd Ed.)**(Springer Series in Information Sciences, 30). Springer Verlag, June, 1997.

KOLLIAS, S., ANASTASSIOUM, D. An adaptive least square algorithm for the efficient training of artificial neural networks. *IEEE Trans. Circuits Syst*., vol. 36, no. 8, pp. 1092-1101, Aug. 1989.

KORENBERG, M., BILLINGS, S. A., LIU, Y. P., MCLLROY, P. J. Orthogonal parameter estimation algorithm for non-linear stochastic systems. *International Journal of Control*, vol. 48, no. 1, pp. 193-210, 1988.

KURKOVÁ, V. Kolmogorov's theorem and multilayer neural networks. *Neural Networks*, vol. 5, pp. 501-506, 1992.

KUNG, S.Y., HU, Y.H. A Frobenius approximation reduction method (FARM) for determining optimal number of hidden units. In *Proc. Int. Joint Conf. Neural Networks*, Seattlem WA, vol. II, 1991, pp. 163-168.

KUSCHEWSKI, J.G., HUI, S., ZAK, S.H. Application of feedforward neural networks to dynamical system identification and control. *IEEE Transactions on Control System Technology*, vol. 1, no. 1, pp. 37-49, 1993.

KWOK, T.Y, YEUNG, D.Y. Use of Bias Term in Projection Pursuit Learning Improves Approximation and Convergence Properties. *IEEE Transactions on Neural* 1996.

LAIRD, J., NEWELL, A., ROSENBLOOM, P. SOAR: An Architecture for general intelligence. *Artificial Intelligence*, vol. 33, no. 1, pp. 1-64, 1987.

LANDAU, Y. D. **Adaptative Control: The Model Reference Approach**. New York, Marcel Dekker, 1979.

LAPEDES, A., FABER, R. How neural nets work. In **Neural Information Processing Systems**, Anderson (Ed.), pp. 442-456. New York, American Institute of Physics. Paris, France, 1987.

LASHKARI, Y., METRAL, M. , MAES, P. Collaborative Interface Agents. In *Proceedings of the National Conference on Artificial Intelligence*, 1994. *http://agents.www.media.mit.edu/groups/agents/publications/aaai-ymp/aaai.html*

LECUN, I. **Modeles connexionnistes de l'apprentissage**. PhD. Thesis. Universitie Paris VI. Paris, France, 1987.

LECUN, Y., DENKER, J. S., SOLLA, S. A. Optimal brain damages. In D. S. Touretzky. Editor, in *Advances in Neural Information Processing Systems, vol. 2,* pp. 598-605, Morgan Kaufmann, San Mateo, CA, 1990.

LEE, S., RHEE, M.K. A gaussian potencial function network with hierarchically self-organizing learning. *Neural Networks*, vol. 4, pp. 207-224, 1991.

LEIBNITZ The application of the Mobile agents in the decentralized heterogeneous database systems, 1995.

LENAT, D.B., GUHA, R.V., PITTMAN, K., PRATT, D., SHEPHERD, M. Cyc: Toward programs with common sense. *Comunication of the ACM*, no. 33, vol. 8, pp. 30-49, 1990.

LENNY, F. What's an Agent, Anyway? A Sociological Case Study, MIT Media Lab.,1993.

LEONTARITIS, I.J.& BILLINGS, S. A Input-output parametric models for non-linear systems. Part I: deterministic non-linear systems. Part II: stochastic non-linear systems, *International Journal Control,* vol. 41, no 2, pp. 304-344, 1985.

LEONTARITIS, I.J., BILLINGS, S. A. Model selection and validation methods for non-linear systems. *International Journal of Control*, vol. 45, no. 3, pp. 311-341, 1987.

LEVIN, A., NARENDRA, K.S. Control of nonlinear dynamical systems using Neural Networks: Controllability and stabilization. *IEEE Trans. Neural Networks*, vol. 4, no. 2, pp. 192-206. March 1993

LEVIN, A., NARENDRA, K.S. Control of nonlinear systems using neural networks. Part II – Inaccessible states, *Technical Report*, Center for Systems Science, Yale University, New Haven, july, 1992.

LEVIN, A.U., NARENDRA, K.S. Stabilization of nonlinear dynamical systems using neural networks. In IEEE/INNS Int. *Conf. on Neural Networks*, Baltimore, MD. 1992.

LIEBERMAN, H. Letizia: An Agent that assists Web Browsing. *Proceedings of the International Joint Conference on Artificial Intelligence*, Montreal, Canada, August,1995.

LIERBERMAN, H. Autonomous Interface Agents. M.I.T. *Proceedings of the ACM Conference on Computers and Human Interface*, CHI-97, Atlanta, Georgia, March 1997
http://lieber.www.media.mit.edu/people/lieber/letizia/AIA/AIA.html

LIU, G. P., KADIRKAMANATHAN, V., BILLINGS, S.A. Variable neural networks for adaptive control of nonlinear systems. *IEEE Transactions on Systems, Man and Cybernetics*, vol. 4, 1998a.

LIU, G.P., KADIRMANATHAN, V., BILLINGS, S. A. On-line identification of nonlinear systems using Volterra polynomial basis function neural networks. *Neural Networks*, vol. 11, 1645-1657, 1998b.

LIU, G.P., KADIRKAMANATHAN, V., BILLINGS, S.A. Nonlinear predictive control using neural networks. *International Journal of Control* (in press), 1996.

LJUNG, L., GLAD, T. Modelling of dynamic systems. *Information and systems science series*. Englewood Cliffs, NJ, Prentice Hall, 1994.

LOBER, A. , WANGEN, L.E, KOWALSKI, B.R. A theoretical foundation for the PLS algorithm. *Journal Chemometrics*, vol. 1, pp. 19-31, 1987.

LUCASISUS, C.B., KATERMAN, G. Towards solving subset selection problems with the aid of the genetic algorithm. In. R. Manner and B. Manderick, (Eds.), **Paralell Problem Solving from Nature**, vol. 2. Amsterdam, Elsevier Science Publishers, 1992.

LUENBERGER, D. G. **Linear and Nonlinear Programming**, Reading, Massachustes, Addison-Wesley Publishing Company, 1989.

LUO, W., BILLINGS, S.A. Adaptive model selection and estimation for nonlinear systems using a sliding data window. *Signal Processing*, vol. 46, pp. 179-202, 1995.

LUO, W., BILLINGS, S.A. Structure selective updating for nonlinear models and radial basis function neural networks. *International Journal of Control and Signal Processing* (in press), 1998.

LUO, W., BILLINGS, S.A., TSANG, K.M. On line structure detection and parameter estimation with exponencial windowing for nonlinear systems. *European Journal of Control*, vol. 2, pp. 291-304, 1996.

MACKAY, D.J.C. Bayesian interpolation. *Neural Computation*, vol. 4, no. 3, pp. 415-447, 1992.

MAECHLER, M., MARTIN, D., SCHIMERT, J., CSOPPENSZKY, M., HWANG, J.N. Projetion pursuit learning networks for regression. In *Proc. 2nd Int. IEEE Conf. Tools Artificial Intell.*, Herndon, VA, Nov. 1990, pp. 350-358.

MAES, P. Agents that reduce Work and information overload. *Communications of the ACM*, vol. 37, no. 7, july 1994.

MAES, P. Artificial Life Meets Entertainment: Life like Autonomous Agents. *Comunications of the ACM*, vol. 38, no. 11, pp. 108-114, 1995, http://pattie.www.media.mit.edu/people/pattie/CACM-94/CACM-94.p5.html

MAES, P. Situated Agents Can Have Goals, In Maes, P. (ed) **Designing Autonomus Agents: Theory and Pratice from Biology to Engineering and Back**, London, The MIT press, 49-70, 1991.

MARTENS, H., NAES, T. Multivariate Calibration. Wiley, New York, 1989.

MARTIN, E. B., MORRIS, A. J., ZHANG, J. Artificial neural networks and multivariable statistics. In (Ed.) Bulsari, A. B. (Ed.) **Neural Networks for Chemical Eng**ineers, 1995.

MCCARTHY, J. A basis for mathematical theory of computation, pp. 33-70. In P. Braffort and D Herschberg, editors, *Computer programming and formal systems*, pp. 33-70, North Holland, 1963.

MCCELLAND, J.L., RUMELHART, D. E., **Parallel Distributed Processing**, Cambridge, MA, MIT Press, 1986, vol. 1.

MCCULLOCH, W. S., PITTS, W. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, vol. 5, 115-133, 1943.

MCDONNELL, J.R., WAGEN, D. Evolving Recurrent Perceptrons for Time-Series Modeling. *IEEE Transactions Neural Networks*, vol. 5, no. 1, january, 1994

METRAL, M., LASHKARI, Y., MAES, P. Maximus, MIT Media Lab, 1993-1994. *http://agents.www.media.mit.edu/groups/agents/projects/*

MILLER, W.T., SUTTON, R.S., WERBOS, P.J. **Neural Networks for Control**. MIT Press. Cambridge, Massachusettes, 1990.

MILLS, P.M., ZOMAYA, A.Y., TADE, M. O. Adaptive model-based control using neural networks. International Journal Control, vol. 60, pp. 1163-1192, 1994.

MINSKY, M.L. Steps towards artificial intelligence. *Proceedings of the Institute of Radio Engineers*, vol. 49, pp. 8-30, 1961.

MINSKY, M.L. A Conversation with Marvin Minsky about Agents. *Communications of the ACM,* vol. 37, no. 7, July 1994.

MINSKY, M.L., PAPERT, S.A. **Perceptrons**. MIT Press. Cambridge, MA, 1969.

MITCHELL, T., CARUANA, R., FREITAG, D., MCDERMOTT, J., ZABORWSKI, D. Experience with a learning personal assistant. *Comunication of the ACM*, vol 7, no. 37, pp. 80-91, 1994.

MOCLOONE, S., BROWN, M.D., IRWIN, G.W., LIGHTBODY, G. A hybrid linear/nonlinear training algorithm for feedforward neural networks. *IEEE Transactions on Neural Networks*, vol. 9, no. 4, pp. 669-684, July, 1998.

MOODY, J., YARVIN, N. Networks with learned unit response functions. In *Advances in Neural Information Processing Systems*, vol. 4, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. San Mateo, CA: Morgan Kaufmann, 1992, pp. 1048-1055.

MORREALE, P. Agents on the Move. *IEEE Spectrum* - April 1998

MOZER, M. C., SMOLENSKY, P. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In D. S. Touretzky, Editor, in *Advances in Neural Information Processing Systems*, vol. 1, pp. 107-115, Morgan Kaufmann, San Mateo, CA, 1989.

MÜHLENBEIN, H. Limitations of multilayer perceptrons – Step towards genetic neural networks, *Parallel Computation.*, vol. 14, no. 3, pp. 249-260, 1990.

MUKHOPADHYAY, S & NARENDRA, K. S. Disturbance rejection in nonlinear systems using neural networks. *Technical report* no. 9114. Center for Systems Science, Yale University. New Haven, CT, 1991.

MURCH, R., JOHNSON, T. **Intelligent Software Agents**. Prentice Hall,1999.

MUSAVI, M.T., AHMED, W. CHAN, K.H., FARIS, K.B., HUMMELS, D.M. On the training of radial basics function classifiers. *Neural Networks*, vol. 5, pp. 595-603, 1992.

NARENDRA, K.S. Adaptive control of dynamical systems. In. D. A. White, D. Sofdge (Eds.), **Handbook of Intelligent Control: Neural adaptive and fuzzy approaches**. New York, Van Nostrand, pp. 143-183, 1996.

NARENDRA K.S., PARTHASARATHY, K. A diagrammatic representation of backpropagation. *Technical Report 8815*, Center for Systems Science, Department of Electrical Engineering, Yale university, New Haven, CT, 1988a.

NARENDRA K.S., PARTHASARATHY, K. Neural networks and dynamical systems. Part I: *A* gradient approach to Hopfield network. *Technical Report 8820*, Center for Systems Science, Department of Electrical Engineering, Yale university, New Haven, CT, 1988b.

NARENDRA, K. S., PARTHASARATHY, K. Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990.

NARENDRA, K. S., PARTHASARATHY, K. Neural networks in dynamical system. *SPIE Intelligent Control and Adaptive Systems*, vol. 1196, 1989

NARENDRA, K.S., PARTHASARATHY, K, Neural networks and dynamical systems. *International Jounal of Approximate Reasoning*, 1992.

NARENDRA, K.S., ANNASWAMY, A.M. **Stable Adaptive Systems**. Englewood Cliffs, NJ, Prentice-Hall, 1989.

NARENDRA, K.S., MUKHOPADHYAY, S. Adaptive Control of Nonlinear Multivariable Systems Using Neural Networks. Neural Networks, vol. 7, no. 5, pp. 737-752, 1994.

NARENDRA, K.S., MUKHOPADHYAY, S. Intelligent Control using Neural Networks. (Tech. Rep no. 9307), Yale University, Center for Systems Science. In M. M. Gupta and K. M. Sinha (Eds.). **Intelligent Control Systems - Theory and Application**, IEE Press, pp. 151-186, 1996.

NARENDRA, K.S., PARTHASARATHY, K. Gradient methods for the optimization of dynamical systems containing neural networks, *IEEE Transactions o Neural Networks* vol. 2, pp. 252-262, 1991.

NARENDRA, K.S., PARTHASARATHY, K. Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27, 1990.

NDUMU , D.T., TAH, J. M .H. Agents in Computer-Assited Collaborative Design. Intelligent Systems Research, BT Laboratories. *BT Technology Journal*, 1999.

NDUMU, D. T, NWANA, H. S., LYNDON, C. LEE & HAYDN R. HAYNES. The visualisation and debugging of distributed multi-agents. *BT Technology Journal*, 1999.

NEGROPONTE, N., KAY, A. Software Agents - MIT Media Lab - *Autonomous Agent Group*,1984 http://agents.www.media.mit.edu/groups/ag...papers/newtesis/susection2_5.html

NERRAND, O., ROUSSEL-RAGOT, P., URBANI, D., PERSONNAZ, L., DREYFUS, G. Training recurrent neural networks: Why and how? An illustration in Process Modelling. *IEEE Transactions on Neural Networks* vol. 5, pp178-184, 1994.

NEWELL, A., SIMON, H.A. **Human Problem Solving**. Prentice-Hall, 1972.

NG, G.W. **Application of Neural Networks to Adaptive Control of Nonlinear Systems**. John Wiley & Sons Inc., 1997.

NG, G. W., COOK, P A. On-line adaptive control of non-linear plants using neural networks with application to liquid level control. *International Journal of Control and Signal Processing*, vol. 12, pp. 13-28, 1998.

NGUYEN, D.H., WIDROW, B. Neural networks for Self-learning Control Systems. *IEEE Control Systems Magazine*, vol. 10, pp. 18-23, 1990.

NORMAN, D. How Might People Interact with Agents. *Communications of the ACM*, vol. 37, no. 7, july 1994.

NWANA, H., NDUMU, D. An Introduction to Agent Technology. *BT Technology Journal*, vol.14, no. 4, 1996.

OGATA, K. **Modern Control Engineering**. Prentice Hall, 3rd. edition, 1996.

ORTEGA, J. G., CAMACHO, E. F. Mobile robot navigation in a partially structured static environment, using neural predictive control. *Control Engineering Practice*, vol. 4, no. 12, pp. 1669-1679, 1996.

PALANISWAMI, M., ARTTIKIOUZEL Y., MARKS II R. J., FOGEL, D., FUKUDA T. (editors) **Computational Intelligence – A Dynamic System Perspective**. IEE Press, 1995.

PARK, J., SANDBERG, I. W. Universal approximation using radial-basis function networks, *Neural Computation*, vol. 3, pp. 246-257, 1991.

PARK, M.G., CHO, N.Z. Self-tuning control of a nuclear-reactor using a gaussiana function neural-networks. *Nuclear Technology*, vol. 110, pp. 285-293, 1995.

PARKER, D. B. Learning logic. *Center for Comput. Research Economics and Management Sci.*, Massachusetts *Inst. Technol., Technical Report TR-47*, *MIT*, Apr. 1985.

PEARLMUTTER, B. A. Fast Exact Multiplication by the Hessian. *Neural Computation*, vol. 6, no. 1, pp. 147-160, 1994.

PHAM, K.M. The NeurOagent: A Neural Multi-agent approach for modelling, distributed processing and Learning. **Intelligent Hybrid Systems**, Edited By S. Goonatilake and S. Khebbal, 1995.

PHILLIPS, S. M., MÜLLER-DOTT, C. Identification and Control. *International Journal of Analog Integrated Cicuits and Signal Processing*, vol 2, no. 4, pp. 353-363, 1992.

PIOVOSO, M.J., OWENS, A.J. Sensor data analysis using artificial neural networks. In (Eds.) Arkun, Y. and Ray W.H. **Chemical Process Control CPC IV**, CACHE, Austin, TX, 1986.

POGGIO, T., GIROSI, F. Networks for approximation and learning. *Proc. IEEE*, vol. 78, no. 9, Sept. 1990.

POLYCARPOU, M., LOANNOU, P. Identification and control of systems using neural networks models: Design and stability analysis. *Dept. Electrical Engineering Systems, Univ. Southern California, Los Angeles, Tech. Rep. 91-09-01,* Sept. 1991.

PSALTIS, D., SIDERIS, A., YAMAMURA, A. A. A multilayered neural network controller. *IEEE Control Systems Magazine*, vol. 8, pp. 17-21, 1988.

PSLATIS, D., SIDERIS, A, YAMAMURA, A. Neural controllers. Proc. IEEE Int. Conf. on Neural Networks, vol. 4, pp. 21-24, 1987.

PYLYSHYN, Z. W. **Computation and Cognition – Toward a Foundation for Cognitive Science**. The MIT Press, 1984.

QUIN, S.J., MCAVOY, T. J. Nonlinear PLS Modeling Using Neural Networks. Comput. *Chem. Engng*, vol. 16, n° 4, pp. 379-391, 1992.

REED, R. Prunning algoritms -- A survey. *IEEE Trans Neural Networks*, vol. 4, n.° 5, pp. 740-747, September 1993.

REILLY, D.L., SCOFIELD, C., ELBAUM, COOPER, L. N. Learning system architectures composed of multiple learning modules. *In Proc. IEEE 1st Int. Conf. Neural Networks*, pp. 495-503, 1987.

RIVALS, I. & PERSONNAZ, L. Black-Box Modelling with state-space neural networks. i*n* R. Zbikowski , K. J. Hunt (eds.) World Scientific Series in Robotics and Intelligent Systems - **Neural Adaptive Control Technology**, vol. 15, pp.237-264, Singapore, World Scientific Publishing Co. Pte. Ltd, 1996.

RONCO, E. **Apprentissage a complexite progressive dans les systems connexionnisties**. Master's thesis. Institut National Polytechnique de Grenoble. Grenoble, France,1994.

RONCO, E., GAWTHOP, P. J., Modular neural neural networks: a state of the art. Technical Report CSC-95026. Centre for System and Control. Faculty of Mechanical Engineering, University of Glasgow, UK, 1995. Available at http://www.mech.gla.ac.uk/~ericr/pub/surveyMNM.ps.

RONCO, E., GOLLEE, H., GAWTHROP, P. J. Modular neural network and self decomposition. Technical Report CSC-96012. Centre for System and Control, University. of Glasgow, UK, 1996. Available at www.mech.gla.ac.uk/~ericr/pub/mnsd.ps.Z.

ROOSEN, C. B., HASTIE, T.J. Automatic Smoothing Spline Projection Pursuit. *Journal of Computational and Graphical Statistics*, vol. 3, no. 3, pp. 235-248, 1994.

ROSENBLATT, F. Perceptron simulation experiments. In *Proceedings of the Institute of Radio Engineers (IRE)*, vol. 48, pp. 301-309, 1960.

RUMELHART, D.E., HINTON, G.E., WILLIAMS, S. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, J. L. McCelland and D. E. Rumelhart, Eds. Cambidge, MA, MIT Press, 1986, vol. 1, pp. 318-361.

RUMLELHART, D. E., MCCLELLAND, J.L. **Parallel Distributed Processing**, vol. 1, MIT Press, Cambridge, 1986.

RUSNAK, A., FIKAR, M., NAJIM, K., MESZAROS, A. Generalized predictive control based on neural networks. *Neural Processing Letters*, vol. 4, pp. 107-112, 1996.

RUSSELL, S.J., NORVIG, P. **Artificial Intelligence: A Modern Approach,** Englewood Cliffs, NJ, Prentice Hall, 1995

SADEGH, N. A perceptron networks for functional identification and control of nonlinear systems. *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 982-988, 1993.

SANNER, R.M., AKIN, D.L. Neuromorphic pitch attitude regulation of an underwater telerobot. *IEEE Control Systems Magazine*, vol.10, pp. 62-67, 1990.

SANNER, R.M., SLOTINE, J.J.E. Gaussian networks for direct adaptive control. *IEEE Transactions. Neural Networks*, vol. 3, Nov. 1992.

SASTRY, S., BODSON, M. Adaptative Control: Stability, Convergence and Robustness. Englewood Cliffs, NJ, Prentice-Hall, 1989

SCHETZEN, M. Nonlinear system modeling based on the Wiener theory. *Proceedings of the IEEE*, vol. 69, pp. 1557-1572, 1981.

SCHMIDT, R.A. **Motor Control and Learning**. Human Kinetics Publishers. Champaign, Illinois, 1982

SCHWARZENBACH, J., GILL, K.F. **System Modelling and Control**. Edward Arnold, 1992.

SEBORG, D.E. A perspective on advanced strategies for process control. *Modeling, Identification and Control*, vol. 15, no. 3, pp. 179-189, 1994.

SELFRIDGE, O. G. Pandemonium: A paradigm for learning. *In Proc. Symp. Held Physical Lab.*: *Mechanisation Thought Processing*, Londom, HMSO, nov. 1958, pp. 511-517.

SELKER, T. COACH: A teaching agent that learns. *Comunication of the ACM*, vol 7, no. 37, pp. 92-99, 1994

SHANNON, D. F. Recent advances in numerical techniques for large-scale optimization. In *Neural Networks for Robotics and Control*, W. T. Miller, R. Sutton, and P. J. Werbos, Eds. Cambridge, MA, MIT Press, 1990.

SJÖBERG, J. **Non-Linear Systems Identification with Neural Networks**, Ph.D. Thesis, Departament of Electrical Engineering, Linköping University , Sweden, 1995.

SJÖBERG, J., ZHANG, Q., LJUNG, L., BENEVENISTE, A., DELYON, B., GLORENNESC, P.-Y., HJALMARSSON, H., JUDITSKY, A. Nonlinear Black-Box Modelling in System Identification: a Unified Overview. *Automatica*, vol. 31, no. 12, pp. 1691-1724, 1995.

SJØGAARD, S. **A conceptual approach to generalization in dynamic neural networks**. Ph.S. Dissertation. Comput., Sei. Dep., Aarhus Univ. Aahus C., Denmark, 1991.

SLOTINE, J.J.E. The robust control of robot manipulators. The International Jounal of Robotics Research, vol. 4, pp. 49-64, 1985.

ŚMIEJA, F.J. Multiple network systems (MIMOS) modules: Task division and module discrimination. In *Proc. 8th AISB Conf, Artificial Intelligent*, Leeds, April 16-19, 1991. Also available as GMD Tech. Rep. 638.

ŚMIEJA, F.J. The Pandemonium System of Reflective Agents. *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. no. 97-106, Jan 1996

ŚMIEJA, F.J., MÜHLENBEIN, H. Reflective modular neural networks systems. *Gesellschaft für Mathematik und Datenverarbeitung*, St. Augustin, Germany, Tech. Rep. 633, Feb 1992.

SMITH, D.C., CYPHER, A., SPOHRER, J. KidSim: Programming Agents Without a Programming Language. *Communications of the ACM*, 37, 7, 55-67, 1994

SMYTH, P. On stochastic complexity and admissible models for neural network classifiers. In R. P Lippmann, J. Moody and D. S. Touretzky (Eds.), Advances in neural informational processing systems vol. 3. San Mateo, CA, Morgan Kaufmann, 1991.

SONTAG, E. Some Topics in Neural Networks and Control. *Technical Report LS93-02*, Departament of Mathematics Rutgers University, 1993.

SOUZA E SILVA, M.E. **De agentes racionais a agentes semióticos: um estudo sobre a aplicação da Semiótica na concepção de sistemas inteligentes**. Tese de Mestrado. Centro de Ciências e Tecnologia – CCT, UFPB, 1998

STECK, J. E., ROKHSAZ, K., SHUE, S. P. Linear and neural networks feedback for flight control decoupling. *IEEE Control Systems Magazine*, vol. 16, no. 4, pp. 22-30, 1996.

STONE, M., BROOKS, R.J. Continuum regression: cross-validated sequentially constructed prediction embracing ordinary least squares, partial least squares and principal components regression. *Journal of the Royal Statistical Society,* vol. 52 B, nº 2, pp. 237-269, 1990.

SUNDBERG, R. Continuum regression and ridge regression. *Journal of the Royal Statistical Society*., vol. (55) B, n.° 3, 653-659, 1993.

SUTTON, R.S., BARTO, A.G. **Reinforcement Learning: An Introductio**n (Adaptive Computation and Machine Learning). MIT Press, March 1998.

SZILAS, N., RONCO, E. Action for learning in non-symbolic systems. In *European Conference on Cognitive Science*, *Fuzzy Sets and Systems,* vol. 18, pp. 329-346, 1995.

TANENBAUM, A. **Sistemas Operacionais Modernos**. Prentice-Hall 1995

TEMENG, K.O., SCHNELLE, P.D., MCAVOY, T.J. Model-predictive control of an industrial packed-bed reactor using neural networks. *Journal Process Control*, vol. 5, pp. 19-27, 1995.

TENORIO, M. F., LEE, W. T. Self-organizing neural nets for the identification problem. In *Advances in Neural Information Processing System* vol. 1, D. Touretzky, Ed. San Mateo, CA, Morgan Kaufmann, 1989.

TIKHONOV, A.N., ARSENIN, V.Y. **Solutions of Ill-posed Problems**. Washington, D.C., W. H. Winston, 1977.

TOASTES, F. M. **Control theory in biology and experimental psychology**. London, 1975.

TZIRKEL-HANCOCK, E. Stable control of nonlinear systems using neural networks. Ph.D. dissertation, Cambridge Univ. Eng. Dept., Cambridge, U.K., June 1992.

VANLANDINGHAM, H.F., BINGULAC, S., TRAN, M. A comparison of conventional and neural network approaches to system identification. *C-TAT,* vol. 9, no. 1, 1993a.

VANLANDINGHAM, H.F., CHOI, J.Y., TRAN, M. System identification with multilayer perceptrons. In *Proc IEEE Systems, Man, and Cybernetic* Conf. LeTouquet France, Oct. 1993b.

VAPNIK, V. **The Nature of Statistical Learning Theory**. Springer, 1995.

VENUGOPAL, K.P., PANDYA, A.S., SUDHAKAR, R. A recurrent neural network controller and learning algorithm for the on-line learning control of autonomous underwater vehicles. *Neural Networks*, vol. 7, no. 5, pp. 833-846, 1994.

VIDYASAGAR, M. **Nonlinear Systems Analysis**. New Jersey: Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 1993.

VIRDHAGRISWARAN, S. Defining mobile agent technology. *In. Crystaliz*, 1995.

VON ZUBEN, F. J. **Modelos paramétricos e não-paramétricos de redes neurais artificiais e aplicações**. Tese de Doutorado, Departamento de Engenharia de Computação e Automação Industrial, Faculdade de Engenharia Elétrica, Unicamp, Fevereiro, 1996.

WAHBA, G. Smoothing Noisy Data with Spline Functions. *Numerische Mathematik*, vol. 24, no. 5, pp. 383-393, 1975.

WANG, S., YEH, H. Self-Adaptive Neural Architecture for Control Applications. *Proceedings of the IEEE International Conference on Neural Networks*, vol. 3, no. 1, pp. 309-314, 1990.

WATROUS, R. L. Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization. In *Proc. 1987 IEEE Int. Conf. Neural Networks*: vol II, San Diego, CA, pp. 619-627, june 1987.

WEIGNED, A.S., RUMELHART, D. E., HUBERMAM, B.A. Generalization by weight-elimination with application to forescating. In *Advances in Neural Information Processing System* vol. 3. R. Lippmann, J. Moody, and D. Touretzky, Eds. San Mateo, CA, Morgan Kaufmann, 1991, pp. 875-882.

WERBOS, P. J. Supervised Learning: Can it escape its local minimum? *In First World Congress on Neural Networks*, July, 1993. INNS Press/ Erlbaum.

WERBOS, P.J. **Beyond Regression: New Tools for Prediction and Analysis in the Behavior Sciences.** PhD. Thesis. Harvard University, Committee on Applied Mathematics, 1974.

WEXELBLAT, A., MAES, P. Footprints: History-Rich Tools for Information Foraging, CHI'99. *Proceedings, ACM Press*, 1999. http://agents.www.media.mit.edu/groups/agents/paper/aaai-ymp/aaai.html

WIDROW, B., HOFF, M.E. Adaptive switching circuits. *WESCON Conv. Rec*. pp. 96-140,1960.

WIDROW, B. & SMITH, F.W. Pattern-recognizing control systems. *In Computer and Information Sciences Symposium Proceedings*, Spartan, Washington, DC, 1963.

WIDROW, B., STEARNS, D. **Adaptive Signal Processing**. Prentice-Hall. Englewood Cliffs, 1985.

WIENER, N. **Cybernetics**, The MIT Press, 1948.

WISE, B. M, RICKER, N. L. Identification of finite impulse response models with continuum regression. *Journal Chemom*. vol. 7, pp. 1-14, 1993.

WOLD, S. Nonlinear partial least squares modelling II. Spline inner relation. *Chemom. Intell. Lab. Systems*, vol. 14, pp. 71-84, 1992.

WOOLDRIDGE, M., JENNINGS, N. R. Agent Theories, Architectures, and Languages: a Survey. In Wooldridge and Jennings Eds., **Intelligent Agents**, Berlin, Springer-Verlag, pp. 1-22, 1995.

YEUNG, D.Y. Constructive neural networks as estimator of Bayesian discriminant functions. *Patern Recognition*, vol. 26, n.°1, pp. 189-204, 1993.

ZBIKOWSKI, R.W. **Recurrent Neural Networks: Some Control Aspects**. Ph.D Thesis, Faculty of Engineering, Glasgow University, 1994.

ZHANG, Y., SEN, P., HEARN, G.E. An on-line trained adaptive neural controller. *IEEE Control Systems Magazine*, vol. 15, pp. 67-75, 1995.

ZURADA, J.M., MARKS II, R.J., ROBINSON, C.J. (editors) **Computational Intelligence** – Imitating Life. IEEE Press, 1994.

Índice Remissivo de Autores