Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica

Departamento de Engenharia de Computação e Automação Industrial

Representação e aquisição de regras em sistemas conexionistas

Autor: Engº Alexandre Ricardo Soares Romariz

Orientador: Prof. Dr. Márcio Luiz de Andrade Netto

Dissertação apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas, como requisito parcial para a obtenção do grau de **Mestre em Engenharia Elétrica.**

Campinas, fevereiro de 1995

A meus pais, José e Dôra.

Agradecimentos:

É boa e justa a tradição de, concluído o trabalho, deixar registrada uma tímida noção da dívida do autor com inúmeras pessoas.

Durante toda a pesquisa, sempre pude contar com o auxílio do Professor Márcio, que soube indicar caminhos, apontar problemas e possíveis soluções, mantendo-se sempre aberto a idéias novas. O clima de liberdade e descontração que foi estabelecido entre nós contribuiu em muito para o progresso do trabalho.

Muito antes de deixar Brasília e buscar a pós-graduação em Campinas, foi necessário crescer. E eu cresci em meio a uma família enorme e feliz. A convivência com estas muitas pessoas fundamenta meus mais importantes sentimentos e crenças. Nunca serão, portanto, exagerados os agradecimentos a meus pais, José e Dôra, minha tia Aida, meus irmão Rose, Marilene, José, Fátima, Marize, Luiz e João (sim, para minha sorte são muitos).

E, à medida em que cresci, foram surgindo outras pessoas. Não são ligadas a mim por laços de família, mas este erro foi logo corrigido com outros laços. Aqui não posso citar nomes de todos os que me acompanharam nos colégios, na UnB e em Campinas. Que os represente meu amigo Roberto, que não satisfeito em me aturar como colega desde 87, por dois anos e meio fez minha também a sua casa em Campinas.

Este trabalho não seria possível sem o apoio de numerosas instituições. Devo agradecer à CAPES pelo financiamento e às muitas universidades em todo o mundo que mantêm acesso público a seus trabalhos pela rede Internet, em particular aos administradores do repositório *Neuroprose* (Ohio State University) e do repositório *Machine-Learning Databases* (University of California at Irvine). Por último, um agradecimento à Unicamp, nas pessoas de seus professores, funcionários e alunos.

Só mesmo com o auxilío de tanta gente posso buscar confiança e novas realizações.

"Pois não devíamos estar sossegados, ao rés da terra, com modestos rumos, e não vêm cordas que nos enlaçam, que nos suspendem, que nos deixam numa altura de onde a terra, que é o nosso destino, torna-se o nosso abismo?"

Cecília Meireles

Jogos Circences (crônica)

Resumo

Este trabalho trata da representação de conhecimento estruturado (na forma de regras) em sistemas conexionistas. Primeiramente, é feito um estudo sobre redes conexionistas modulares, nas quais grupos de neurônios podem ser associados a antecedentes e conseqüentes de regras. Em seguida, mostram-se formas pelas quais estas redes são associadas a conceitos de lógica nebulosa, nos chamados sistemas neuronebulosos.

Um algoritmo de aquisição incremental de regras é proposto para tais sistemas. Nele, promove-se alteração estrutural e não apenas adaptação de parâmetros da rede. Novas regras vão sendo adicionadas para lidar com padrões ainda não cobertos pelas regras existentes. O erro decorrente da aplicação de uma regra é usado como indicador da função de pertinência da mesma. Obtém-se assim um procedimento automático de partição do espaço de entrada, que valoriza a inteligibilidade das operações da rede e o aproveitamento efetivo de sugestões fornecidas ao sistema, conforme comprovam os resultados da aplicação do algoritmo em problemas típicos de aproximação de funções e reconhecimento de padrões.

Palavras-chave: Redes neurais; representação de conhecimento; lógica nebulosa; partição do espaço de entrada.

Abstract

This work addresses the problem of representing structured knowledge (as a set of rules) in connectionist systems. First, modular connectionist networks are studied. In this kind of network, groups of neurons may be associated with rule antecedents or consequents. Next, we show some ways by which these networks are associated with fuzzy logic concepts (neuro-fuzzy systems).

An algorithm for incremental rule acquisition is proposed for these systems. Structural modification as well as parameters adaptation are considered. New rules are added periodically to deal with patterns which are not yet covered by the existing rules. The error that results from the application of each rule is used as an indication for membership function construction. By doing so, we build input space partition automatically, emphasizing network operation inteligibility and effective use of hints given to the system. This is confirmed by results obtained through application of this algorithm in machine learning benchmarks.

Keywords: Neural networks; knowledge representation; fuzzy logic; input space partition.

Conteúdo

Conteúdo (continuação)

Lista de figuras

Lista de Tabelas e Quadros

Capítulo 1

Introdução

O presente trabalho visa promover aquisição de conhecimento estruturado a partir do treinamento de uma rede neural. Estes sistemas de inspiração biológica, formados pela interconexão de processadores simples, têm causado grande impacto na área de Inteligência Artificial, em especial a partir de meados da década de 80 (Rumelhart *et al.* 1986), tendo obtido sucesso em numerosas aplicações (Hecht-Nielsen 1991). Sua arquitetura interna, no entanto, favorece uma representação de conhecimento de difícil interpretação.

A capacidade de manipular conhecimento estruturado (na forma de regras, por exemplo) representa uma série de vantagens para sistemas que lidam com problemas complexos, ao estabelecer um canal para comunicação de alto nível com o mundo exterior ou sistemas semelhantes. No caso específico de redes neurais, a falta de representação clara de conhecimento faz com que a exibição de exemplos da operação desejada seja praticamente sua única via de aquisição de informações. Conhecimento parcial sobre o problema tratado não pode ser traduzido facilmente na representação interna das redes.

Outra motivação para a busca de formas de representação de conhecimento nos chamados sistemas conexionistas está ligada à principal proposta da pesquisa em conexionismo: explicar como o comportamento inteligente e consciente pode emergir de conjuntos de processadores que isoladamente manipulam informação não-estruturada (Smolensky 1988).

Nossa pesquisa levou-nos naturalmente ao estudo da lógica nebulosa (Zadeh 1965). Sob este nome é conhecida uma série de formalismos (relações, regras) adequados para lidar com conhecimento vago e impreciso. Estão assim colocados entre o processamento estruturado dos sistemas baseados em regras e o processamento não-estruturado das redes neurais (Kosko 1992), e são candidatos naturais a servirem de ponte entre estes sistemas.

De fato, há desde o início da década uma intensa pesquisa em sistemas que procuram combinar arquitetura de redes neurais e conceitos da teoria de conjuntos nebulosos, nas chamadas redes neuronebulosas (Lin&Lee 1991, Jang 1992, Takagi 1993). Procura-se aliar a capacidade de aprendizado a partir de exemplos com a representação de conhecimento possibilitada pelo emprego de lógica nebulosa.

No entanto, de forma contrária aos nossos objetivos, uma definição inicial de estrutura é arbitrada no início do treinamento destes sistemas híbridos. Tal estrutura, em geral, é obtida por análise da distribuição dos dados disponíveis ou por tentativa e erro, e raramente representa conhecimento sobre o domínio do problema. O treinamento visa o ajuste global de parâmetros para otimizar o desempenho do sistema, o que costuma produzir uma complexa interação entre os elementos que representariam regras, e nos afasta da proposta inicial de inteligibilidade do conhecimento adquirido.

Buscou-se então alternativas para a definição da base de regras, e um novo algoritmo de partição do espaço de entrada foi proposto. Nele, as regras são desenvolvidas uma de cada vez. Uma nova regra é desenvolvida para lidar com regiões não cobertas pelas regras já existentes. Ela é otimizada individualmente, para que represente uma aproximação do comportamento desejado sobre uma região do espaço de entrada, como se espera de uma regra. Uma avaliação heurística do desempenho da regra otimizada é efetuada para que se determine sua região definitiva de aplicação. O algoritmo foi implementado em *software* no Laboratório de Engenharia de Computação e Automação e foi testado em problemas de aproximação de funções e problemas clássicos de aprendizado em máquina, com resultados animadores em termos de clareza das bases de regras resultantes e de aplicação de conhecimento prévio (na forma de regras) diretamente sobre a estrutura da rede.

Há várias linhas de pesquisa divulgadas na literatura que relacionam-se com este trabalho. Em primeiro lugar, como afirmamos acima, a construção de sistemas híbridos unindo redes neurais e lógica nebulosa tem sido intensamente pesquisada nos anos 90 (Lin&Lee 1991, Jang 1992, Takagi 1993 entre vários outros). Há um forte relacionamento entre alguns destes sistemas e as Redes de Funções Radiais de Base (RBFN), propostas inicialmente com inspiração biológica e matemática (Jang & Sun 1992). Pesquisas em arquiteturas conexionistas modulares (Jacobs *et al.* 1990) também originadas com outros propósitos, acabam por desenvolver

estruturas semelhantes. Deve-se notar também que a questão central do trabalho, um procedimento construtivo para definição de regras, envolve aprendizado estrutural, um tema de grande interesse em redes neurais (Hunt *et al.* 1992, Barto 1989). Finalmente, a idéia de aquisição incremental de regras foi inicialmente inspirada no trabalho de McMillan *et al.* (1992), ainda que estes autores tenham trabalhado com regras não nebulosas e com isto não tenham tratado da questão da definição de funções de pertinência.

Esta dissertação está organizada da seguinte forma: o capítulo 2 apresenta uma breve revisão da proposta conexionista no âmbito de inteligência artificial. Apresenta-se também o problema de representação de conhecimento nas redes neurais e descrevem-se arquiteturas modulares que visam dar estrutura à representação interna das redes.

No capítulo 3 temos os fundamentos da teoria de conjuntos nebulosos, com ênfase no estudo dos chamados controladores nebulosos, aplicação de maior emprego prático e com mais claro relacionamento com redes neurais. Aspectos deste relacionamento são então destacados, e no capítulo 4 as várias propostas de redes neuronebulosas são apresentadas. Discute-se ainda a questão da representação de conhecimento nestes sistemas.

Apresentados todos os conceitos, estaremos em condições de descrever o algoritmo proposto, o que é feito no capítulo 5. O capítulo 6 trata dos experimentos realizados e resultados obtidos na validação das idéias propostas. Finalmente, no capítulo 7 é feita uma análise dos objetivos alcançados e das limitações do sistema, sendo apresentados vários aspectos que inspiram pesquisa futura.

Capítulo 2

Conexionismo e Inteligência Artificial

2.1. Origens

A automação do processamento de dados tem sido buscada ao longo da história, em esforços que exploram ao limite o conhecimento tecnológico de cada época. O desenvolvimento alcançado pela eletrônica na década de 40 trouxe finalmente viabilidade técnica a este ideal. O conceito de máquina a programa armazenado (von Neumann 1945) e sua efetiva implementação abriu caminho para a era dos computadores digitais. Sucessivos avanços tecnológicos e o conseqüente aprefeiçoamento destas máquinas resultaram em uma capacidade computacional inimaginável há meio século.

O princípio básico de funcionamento dos computadores digitais não se distanciou do proposto na década de 40 (Hayes 1988). Há diferenças muito acentuadas entre este princípio e a maneira humana de raciocinar, diferenças que não foram atenuadas por mudanças na tecnologia empregada.

Para que um computador possa ser usado na solução de uma classe específica de problemas, um algoritmo geral de solução para esta deve ser definido. A atuação do computador limita-se à execução repetitiva dos passos do algoritmo, com rapidez e precisão. Não há paralelo possível com a capacidade humana de buscar criativamente soluções para problemas novos, de aprender com os próprios erros, ou de buscar uma solução aproximada para um problema complexo, com uso razoável de tempo e recursos. Além disso, as linguagens compreendidas pelos computadores estão muito longe da flexibilidade e robustez da linguagem natural.

Tudo isso contribui para que classes de problemas, tratadas com facilidade por animais, representem um grande desafio para os mais poderosos computadores. Como

exemplos, podemos citar a interpretação de imagens, o controle motor e a tomada de decisões.

A Inteligência Artificial (IA) reúne técnicas que buscam integrar aos sistemas de processamento características típicas do comportamento inteligente. A dificuldade em encontrar uma definição consensual de IA e definir claramente seus limites reside no problema da definição do próprio conceito de inteligência (Hunt 1975). Pode-se dizer que a capacidade de solucionar problemas relacionados à percepção, interpretação, inferência e tomada de decisões, com elevado grau de auto-adaptação, define inteligência no escopo de IA.

As questões levantadas pelas pesquisas em Inteligência Artificial despertam interesse sob a ótica de vários ramos do conhecimento. A fascinante tentativa de construir autômatos com comportamento inteligente relaciona-se diretamente com a elaboração de modelos e hipóteses sobre este comportamento, uma questão cujo interesse está registrado desde a antigüidade. Sendo assim, IA torna-se um campo de pesquisa multidisciplinar, resultado de intensa cooperação entre engenharia, matemátca, física, neurologia, psicologia e filosofia.

A arquitetura clássica de computadores valoriza a manipulação rápida e eficiente de símbolos numéricos e lógicos. Boa parte da pesquisa em IA concentra-se no estudo da lógica e de estruturas complexas de dados, capazes de representar o conhecimento em uma forma tratável por computadores. Estes estudos foram aplicados com sucesso em problemas de dificuldade crescente nos anos 60 e 70 (Nilsson 1980). Algoritmos sofisticados de busca de soluções revelaram-se "inteligentes" o suficiente para evitar a explosão combinatória da pesquisa caso a caso. A manipulação conveniente de fórmulas da lógica de predicados de primeira ordem obteve sucesso na questão da demonstração automática de teoremas e permitiu a definição das chamadas linguagens lógicas, que até certo ponto substituem a descrição passo a passo do algoritmo de solução pela definição lógica do problema. Finalmente, a definição de estruturas para manipulação de dados ("frames", redes de inferência) evoluiu ao ponto de tornar possível representar convenientemente o conhecimento humano em domínios específicos, nos chamados

sistemas especialistas, que obtiveram enorme sucesso em problemas que vão da deteção de falhas (diagnose) até a prospecção de minerais.

O sucesso da abordagem simbólica de IA não impediu que a década de 80 visse ressurgir o interesse por um paradigma diferente de computação: o conexionismo. A idéia principal por trás desta abordagem é a de que computações complexas podem ser obtidas pela combinação de muitos processadores simples altamente interconectados. Este paradigma segue uma inspiração biológica, ao aproveitar idéias emanantes do conhecimento que se tem da estrutura do cérebro e de modelos para os neurônios.

Substitui-se, nesta abordagem, a arquitetura baseada em um processador central por um sistema de Processamento Paralelo e Distribuído, um dos nomes pelo qual os sistemas conexionistas são conhecidos. Características importantes dos processos cognitivos, como a capacidade de considerar simultaneamente múltiplas restrições ou de combinar múltiplas fontes de conhecimento são representadas com naturalidade nesta arquitetura (McClelland *et al.* 1986). Tais características contribuem, por exemplo, para a rapidez e robustez do processo humano de reconhecimento de padrões.

Outra característica do comportamento inteligente que se torna mais facilmente representada nesta abordagem é a capacidade de generalização. Os estímulos complexos recebidos do mundo exterior por um sistema inteligente nunca se repetem exatamente. Torna-se fundamental para estes sistemas o princípio de que estímulos semelhantes devem levar a comportamento semelhante. Ao substituírmos o processamento de símbolos discretos (do tipo *presente* ou *não presente*) por padrões de saída em processadores (funções contínuas), torna-se possível reconhecer proximidade entre estímulos e daí generalizar a experiência existente (Kosko 1992).

O conexionismo enfatiza também um novo ponto de vista para o processo de computação: o ponto de vista de *sistema dinâmico* (Kosko 1992). Computar uma solução corresponde a acompanhar a dinâmica natural de um sistema que evolui com o tempo até seu ponto de equilíbrio, onde as restrições impostas estarão, dentro do possível, obedecidas. Sob esta ótica, a água de um rio, ao correr em direção ao mar, está resolvendo o "problema" de minimizar sua energia potencial gravitacional, e o que diferencia o rio (ou outro sistema dinâmico qualquer) de um sistema cognitivo é a

complexidade deste último (Smolensky 1988). "Programar" um sistema dinâmico cognitvo é definir sua dinâmica de forma tal que sua evolução natural corresponda à solução do problema em questão. Propor um problema é definir o estado inicial do sistema.

Finalmente, o conexionismo nos propõe uma mudança no nível de análise do comportamento inteligente. Os modelos propostos trabalham com representações distribuídas (Hinton *et al.* 1986), nas quais a cada conceito simbólico corresponde um padrão de ativação em um certo conjunto de unidades processadoras. O sinal de saída de um processador isolado não costuma representar um conceito claro, pelo que a abordagem conexionista é conhecida também como subsimbólica. O ideal conexionista é demonstrar como os processos cognitivos conscientes e a manipulação de conceitos simbólicos emergem do nível subsimbólico (Smolensky 1988).

Pode parecer que a questão da verdadeira implementação dos processos cognitivos é de menor importância. Seja qual for esta implementação, ela deve poder ser reproduzida pela máquina a programa armazenado (de fato, a simulação em computadores digitais é a forma mais comum de estudo dos sistemas conexionistas). Nossa atenção deveria concentrar-se, portanto, em um nível superior de análise (Fodor & Pilyshin 1988). Mas a idéia conexionista é a de que princípios importantes para a compreensão do comportamento cognitivo só são visíveis no nível subsimbólico. Muito mais do que uma questão de implementação, esta abordagem valoriza aspectos não contemplados no paradigma simbólico. Estes novos aspectos talvez venham a nos revelar detalhes antes insondáveis da inteligência humana.

2.2. Histórico

Os sistemas conexionistas têm uma história tão ou mais antiga do que a dos computadores digitais. Já nos trabalhos de James (1890) encontramos os princípios fundamentais do que são hoje o modelo de neurônio e o princípio da associação. Mas a primeira análise matemática das potencialidades de redes de elementos processadores inspirados em neurônios data da década de 40 (McCulloch & Pitts 1943). A demonstração de que associações destes neurônios artificiais podem implementar qualquer função lógica

finita foi o primeiro sucesso teórico do conexionismo. John von Neumann, ao propor a máquina a programa armazenado, cita este trabalho e procura associar as operações de sua máquina aos neurônios de McCulloch e Pitts (von Neumann 1945).

A primeira onda de entusiasmo com redes neurais surgiu com o Perceptron de Rosenblatt (Rosenblatt 1958). Este sistema consegue aprender a classificação de padrões a partir de exemplos. Pela primeira vez via-se um modelo de aprendizado e percepção com resultados concretos. Ao entusiasmo seguiu-se uma grande crise com a descoberta das limitações do Perceptron. Tornou-se clássico o caso da função *ou exclusivo*, que apesar de sua aparente simplicidade não pode ser ensinada a este sistema. A extensão do Perceptron para superar estas limitações envolvia uma metodologia matemática desenvolvida na década de 70 e que só viria a ser empregada nos anos 80. O trabalho de Minsky & Papert (1969) onde todas estas dificuldades são apontadas, acompanhadas da declaração de uma crença pessoal (e errônea) dos autores de que a extensão do modelo seria inútil, marca o ocaso das pesquisas em redes neurais na década de 70.

No início dos anos 80, o físico Hopfield traz de volta o interesse na área ao propor um sistema dinâmico baseado em neurônios que desempenha a função de memória associativa (Hopfield 1982). Barto *et al.* (1983) também utilizam neurônios artificiais com aprendizado associativo para tarefas complexas de controle. Kohonen (1984) estuda a capacidade de auto-organização destes sistemas. Mas o ressurgimento do interesse em conexionismo a que nos referimos anteriormente ocorreu com o uso do algoritmo de treinamento por retropropagação de erro (Rumelhart *et al.* 1986b) que permitiu a extensão do Perceptron de Rosenblatt para várias camadas de neurônios e assim superou as dificuldades daquele modelo. Observou-se então uma explosão de aplicações de redes neurais nos mais variados campos (Hecht-Nielsen 1991): reconhecimento de padrões (com aplicações industriais, militares, em medicina e finanças), previsão de séries temporais (com aplicações em economia e finanças), identificação e controle de sistemas dinâmicos complexos, compressão de dados, etc.

Temas recentes da pesquisa em redes neurais incluem métodos numéricos mais eficientes para treinamento (Riedmiller 1994, Cohn 1994, Röbel 1994, Tresp *et al.* 1993b), algoritmos genéticos aplicados ao treinamento, arquiteturas dedicadas ao controle de sistemas dinâmicos (Zbikowski *et al.* 1994), arquiteturas modulares (Jacobs *et al.* 1990, Jordan & Jacobs 1993, Cacciatore & Nowlan 1994) e sistemas híbridos redes neurais / lógica nebulosa (Lin & Lee 1992, Keller *et al.* 1992, Takagi 1993, Pedrycz 1994). Estes últimos dois temas inserem-se no contexto de procurar tornar inteligíveis as operações internas das redes neurais. Este tambem é o tema do nosso trabalho, e este problema será discutido em maior detalhe mais a frente.

2.3. Modelo de neurônio e o Perceptron multicamadas

As diferentes arquiteturas de redes neurais praticamente só têm em comum o fato de utilizarem um grande número de processadores simples densamente interconectados. A topologia da rede varia bastante, como também varia o algoritmo de adaptação de parâmetros (aprendizado). Nesta seção, descreveremos o modelo de neurônio artificial e arquitetura de rede mais comumente utilizados (Rumelhart *et al.* 1986b).

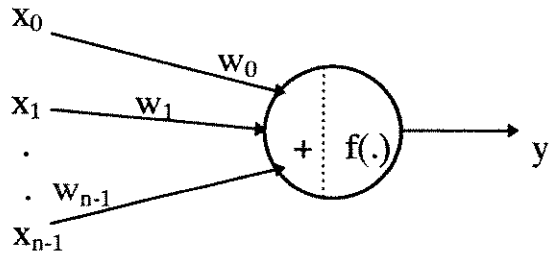


Figura 2.1: Modelo de neurônio artificial

O neurônio esquematizado na Figura 2.1 anterior recebe um vetor de entrada $\mathbf{x}=(x_0,x_1,...,x_{n-1})$ cujos componentes podem representar entradas externas ou valores de saída de outros neurônios, e produz uma saída y, eventualmente transmitida a outros neurônios. O cálculo do valor de saída costuma envolver duas etapas: primeiramente, é efetuada uma soma dos valores de entrada. O vetor de parâmetros ajustáveis $\mathbf{w}=(w_0,w_1,...,w_{n-1})$ define o peso de cada entrada nesta soma. A saída y é então calculada aplicando-se à soma uma função de ativação f, geralmente não-linear. A equação que descreve o comportamento do neurônio é, portanto:

$$y = f\left(\sum_{i=0}^{n-1} x_i . w_i\right) \qquad (2\text{-}1)$$

É comum que o valor de x_0 seja uma constante, fazendo com que o peso w_0 correspondente atue como um sinal de polarização do neurônio.

Em termos de analogia ao neurônio biológico, podemos associar a soma de sinais de entrada ao processo, efetuado pela capacitância da membrana do neurônio, de integração temporal dos pulsos elétricos recebidos. Ainda nesta analogia, o valor de saída representa a freqüência dos pulsos gerados pelo neuônio.

O modelo apresentado para o neurônio pode causar estranheza pela sua simplicidade quando comparado ao que se conhece da intrincada troca de sinais elétrico-químicos no sistema nervoso. Há uma série de aspectos conhecidos da fisiologia do cérebro que não estão representados nas chamadas redes neurais artificiais (Smolensky 1988). Estes modelos não lidam com a organização espacial dos neurônios e das interconexões, e não prevêem a existência de vários tipos de sinais entre processadores.

Se a inspiração biológica é um dos fundamentos do conexionismo, é de se estranhar a ausência, nos modelos pesquisados, de análogos a estruturas conhecidas do sistema nervoso natural. Mas um equilíbrio deve ser buscado entre a plausibilidade biológica dos modelos e sua tratabilidade matemática. Conhecimentos de neurologia (mais abundantes no campo da morfologia do que da fisiologia) são usados como inspiradores de princípios gerais que os modelos estudados procuram seguir (Smolenksy 1988). Alguns dos mais importantes são os seguintes (Rumelhart & McClelland 1986): as interconexões entre os neurônios formam uma rede densa, a comunicação entre eles é feita pelo envio de sinais excitatórios e inibitórios, e o aprendizado envolve alterações nas conexões.

Como dissemos, há uma grande variedade de topologias e dinâmicas para redes neurais. A seguir, descreveremos a arquitetura empregada na maioria das aplicações. A arquitetura e o algoritmo de treinamento do sistema desenvolvido neste trabalho serão apresentados em detalhes posteriormente.

O perceptron multicamadas (Rumelhart *et al.* 1986b) usa uma topologia de camadas de neurônios, com conexões unilaterais de uma camada para a seguinte, mas sem conexões dentro de uma mesma camada (arquitetura "feed forward"). Um exemplo de perceptron com duas camadas é mostrado abaixo:

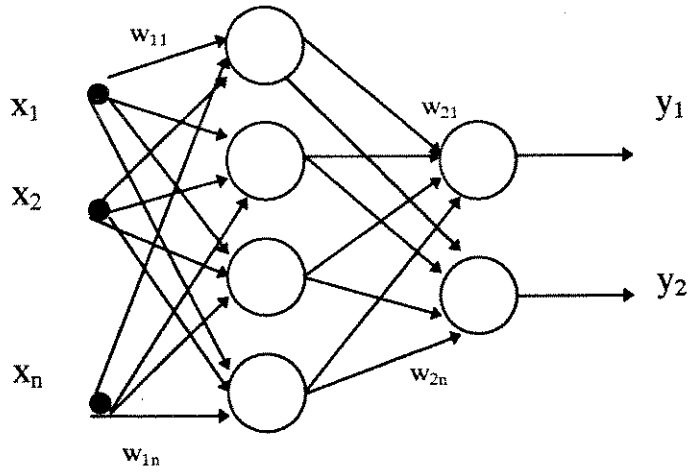


Figura 2.2: Perceptron multicamadas

O número de camadas deve ser definido por tentativa e erro (geralmente duas ou três), bem como o número de neurônios em cada camada (exceto na camada de saída, na qual o número de neurônios coresponde ao número de saídas do sistema). A função de ativação dos neurônios costuma ser do tipo sigmóide ou tangente hiperbólica (Figura 2.3). Estruturas como estas são aproximadores universais: dada uma função contínua qualquer definida em um domínio compacto, existe uma rede neural com uma matriz de conexão entre camadas que a aproxima com erro arbitrariamente pequeno. A seguir, fixaremos atenção no problema de aprendizado, estudando de que forma os exemplos do comportamento desejado para o sistema devem ser usados para ajustar o valor das interconexões da rede.

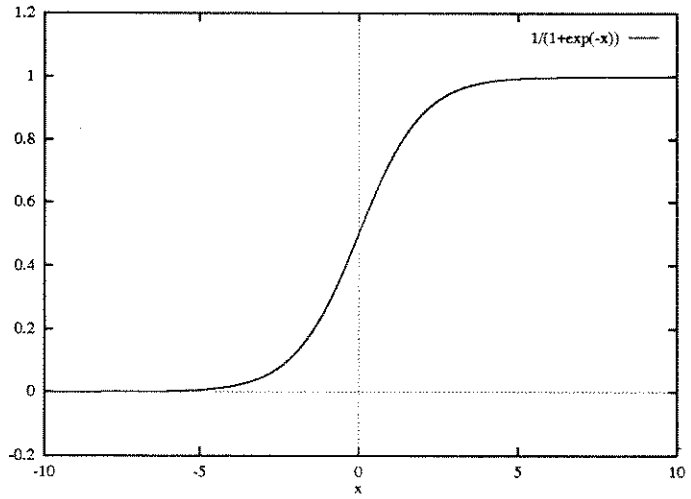


Figura 2.3: Função de transferência típica de neurônio artificial

2.4. Aprendizado

A questão do aprendizado é central na abordagem conexionista. Aprendizado a partir de exemplos era o aspecto mais atraente na proposta do Perceptron (Rosenblatt 1958). Esta característica marca também a mais forte diferença entre conexionismo e computação simbólica: ensinar ao invés de programar, apresentar exemplos ao invés de

algoritmos. É o aprendizado que dá a esta agremiação de processadores simples uma elevada capacidade computacional.

Aprendizado, no âmbito de redes neurais, significa mudança nos valores de parâmetros (Kosko 1992). Um certo algoritmo guia esta mudança com objetivo de melhorar a performance do sistema em um problema específico. Áreas de pesquisa como controle automático e processamento de sinais usam o termo *adaptação* com o mesmo significado. As técnicas de adaptação usadas nestes sistemas muitas vezes só diferem em nomenclatura das empregadas em redes neurais.

O gênero de aprendizado no qual estamos interessados chama-se treinamento supervisionado. Nele, dispomos de um conjunto de exemplos da operação desejada do sistema, chamado arquivo de treinamento. Nestas condições, o aprendizado pode ser visto como um procedimento para obter-se um estimador de uma função dado um conjunto de pontos (Turney 1990, Wahba 1994, Girosi *et al.* 1994, Kosko 1992). Vários trabalhos recentes abordam o problema de treinamento supervisionado e capacidade de generalização em redes neurais, apoiados em conceitos já há muito empregados em matemática e estatística. Consideramos conveniente apresentar aqui alguns destes conceitos, que nos permitem compreender melhor certos problemas encontrados no treinamento.

Estamos interessados em estimar a relação existente entre um vetor $\mathbf{x} \in \mathbf{X}$ (representando as variáveis de entrada) e uma variável de saída $y \in Y$. Como em Niyogi & Girosi (1994), vamos supor que um conjunto

$$D = \left\{(\mathbf{x}i, yi) \in X \times Y\right\}_{i=1}^{m}$$

foi obtido amostrando-se o espaço $\mathbf{X} \times Y$ de acordo com uma distribuição $P(\mathbf{x},y)$ desconhecida. O relacionamento entre $\mathbf{x}$ e y não é totalmente determinístico, ou porque há ruído afetando a obtenção dos dados, ou porque y depende de fatores não modelados no vetor de entrada $\mathbf{x}$. O problema de aprendizado a partir de exemplos é o de construir o melhor estimador possível $f(\mathbf{x})$ dado D.

A medida de adequação de f(**x**) é normalmente o erro quadrático médio do estimador:

$$I[f] = E[(y - f(\mathbf{x}))^2] = \int_{X \times Y} P(\mathbf{x}, y)\ (y - f(\mathbf{x}))^2 d\mathbf{x}\, dy$$

denominado risco esperado para f. Este risco pode ser decomposto (conforme demonstrado em Niyogi & Girosi 1994) da seguinte forma:

$$I[f] = E[(f0(\mathbf{x}) - f(\mathbf{x}))^2] + E[(y - f0(\mathbf{x}))^2]$$

onde $f0(\mathbf{x}) = \int_Y y\, P(y|\mathbf{x})\, dy$ é a função de regressão (valor médio de y para cada **x**) e é evidentemente o estimador que minimiza I[f] (o risco $I[f_0]$ é decorrente apenas do caráter não determinístico do mapeamento **x**→y).

O problema pode então ser redefinido como o de encontrar a função de regressão $f_0[\mathbf{x}]$ dado D. O fato de $f_0(\mathbf{x})$ minimizar I[f] não nos ajuda muito, porque na prática não podemos calcular I[f] (afinal, não conhecemos P(**x**,y)). O conjunto D só nos permite calcular uma aproximação de I[f] que chamaremos risco empírico:

$$Iemp[f] = \frac{1}{m} \sum_{i=1}^{m} (yi - f(\mathbf{x}i))^2$$

Sob condições bastante gerais (Niyogi & Girosi 1994), $I_{emp}[f]$ converge em probabilidade para I[f] com m→∞. Esta convergência, no entanto, não garante que o mínimo de $I_{emp}[f]$ esteja próximo do mínimo de I[f]. Em outras palavras, a minimização de $I_{emp}[f]$ não garante um estimador de baixo risco I[f]. Esta é a origem do chamado "overtraining" (o fato de que, a partir de certo ponto, estender o treinamento da rede resulta em menor erro no arquivo de treinamento, mas pior desempenho em pontos não apresentados) (Wang *et al.* 1993, Turney 1993).

O problema de minimização de $I_{emp}[f]$

$$\min_{f \in \mathcal{F}} (Iemp[f])$$

é claramente mal posto, no sentido de que admite infinitas soluções, se aplicado a uma classe de funções $\mathcal{F}$ muito grande (Niyogi & Girosi 1994). Todas as funções $f \in \mathcal{F}$ que interpolam os pontos de D, por exemplo, têm $I_{emp}[f]=0$. Na teoria da regularização, este problema é estudado fixando-se a condição adicional de que a função f seja suave (Girosi *et al.* 1994). Passamos então a querer minimizar o funcional

$$H[f]=I_{emp}[f]+\lambda\phi[f] \qquad (2\text{-}2)$$

onde $\phi[f]$ é um funcional que traduz a noção de suavidade e $\lambda \geq 0$ é o parâmetro de regularização ($\lambda=0$ nos dá um problema de interpolação, e com $\lambda \rightarrow \infty$ o problema torna-se o de ajustar uma reta por mínimos quadrados) (Wahba 1994).

Podemos também impor a restrição de que f seja uma função parametrizada, isto é, a solução deve pertencer a uma classe

$$Hn = \left\{ \sum_{\alpha=1}^{n} C\alpha \, H(\mathbf{x}; \mathbf{w}\alpha) \right\}$$

H_n pode ser, por exemplo, a classe dos polinômios de grau n, de funções implementáveis em um Perceptron com n unidades,etc. Na prática, em algoritmos computacionais de aproximação de funções, esta restrição está sempre presente, porque as funções f devem ser armazenadas e manipuladas de alguma forma.

Chamando $\hat{f}_{n,m}$ a função $f \in H_n$ que minimiza I_{emp}, é interessante investigar a adequação de $\hat{f}_{n,m}$ aos nossos propósitos em função de n e m. É razoável supor que $\hat{f}_{n,m}$ torne-se melhor estimador com o aumento de m (porque com isto $I_{emp}[f]$ deve ser melhor estimativa de I[f]). Já a questão de n é mais complexa. Um n maior significa maior capacidade de representação para a classe H_n, e por isso esperamos um melhor estimador. Mas um aumento de n com m fixo deteriora a estimativa $I_{emp}[f]$ (exatamente pela maior complexidade de H_n) e piora a questão do "overtraining". De fato, há vários estudos demonstrando que entre estimadores com mesmo $I_{emp}[f]$, o mais simples (menor n) costuma exibir melhores resultados de generalização (Wang *et al.* 1993). Esta questão também está diretamente relacionada ao dilema "bias"- variância estudado em estatística (Meir 1994).

A quantidade $E\left[\left(f_0 - \hat{f}_{n,m}\right)^2\right]$ é chamada erro de generalização. Em Niyogi & Girosi (1994) divide-se este erro em dois componentes: o erro de aproximação, que reflete a incapacidade da classe H_n em representar perfeitamente a função de regressão f_0, e o erro de estimação, decorrente da minimização do risco empírico $I_{emp}[f]$ ao invés do risco $I[f]$. Como dito anteriormente, o erro de aproximação decresce com n, e o de estimação decresce com m e aumenta com n. Assim, para uma quantidade fixa de dados, há um nível ideal para a complexidade do modelo H_n. Para o caso particular de redes cuja saída é uma soma de gaussianas de mesma variância, os autores obtém um resultado do qual conclui-se que o valor ótimo de n é proporcional à potência (1/3) de m.

A minimização do risco empírico $I_{emp}[f]$ é feita por um método de gradiente. A cada ponto do arquivo de treinamento, modificam-se os parâmetros no sentido oposto ao indicado pelo gradiente do erro de saída com relação aos parâmetros atuais. Em termos de equações, se a apresentação do k-ésimo ponto $(\mathbf{x}_k, f(\mathbf{x}_k))$ do arquivo de treinamento produziu a saída $\hat{f}_{\mathbf{w}}(\mathbf{x}_k)$ com o vetor atual de parâmetros $\mathbf{w}$, o erro quadrático com relação à saída desejada é

$$e_k = \left(f(\mathbf{x}_k) - \hat{f}_{\mathbf{w}}(\mathbf{x}_k)\right)^2$$

A modificação no vetor $\mathbf{w}$ segue então a equação

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \frac{1}{2}\varepsilon \nabla e_k$$

ε é uma constante que controla a amplitude da mudança de parâmetros em cada ponto, por vezes chamada constante de aprendizado. É mais um valor a ser determinado por tentativa e erro. Um valor muito pequeno resulta em um treinamento muito demorado. Valores muito elevados fazem-nos correr o risco de "pular"o mínimo desejado, num processo oscilatório que impede a convergênia do algoritmo.

Para o caso de um neurônio com função de ativação identidade, temos (eq. 2-1)

$$\hat{f}_{\mathbf{w}}(\mathbf{x_k}) = \sum_{i=0}^{n-1} w_i . x_{k_i}$$

o que nos dá

$$\nabla e_k = \left[\frac{\partial e_k}{\partial w_0},\ldots,\frac{\partial e_k}{\partial w_{n-1}}\right] = \left[-2e_k \frac{\partial \hat{f}_{\mathbf{w}}(\mathbf{x_k})}{\partial w_0},\ldots,-2e_k \frac{\partial \hat{f}_{\mathbf{w}}(\mathbf{x_k})}{\partial w_{n-1}}\right] =$$

$$= -2e_k\left[x_{k_0},\ldots,x_{k_{n-1}}\right] = -2e_k . \mathbf{x_k}$$

e, portanto,

$$\mathbf{w_{k+1}} = \mathbf{w_k} + \varepsilon . e_k \mathbf{x_k} \qquad (2\text{-}3)$$

Esta é a equação do algoritmo LMS, extensamente utilizado em processamento adaptativo de sinais. Na nomenclatura de redes neurais, a mesma equação costuma ser chamada Regra Delta (Widrow & Lehr 1990). Observa-se que tal regra só pode ser diretamente aplicada à camada de saída de uma rede neural. Se o valor de saída do neurônio não é saída do sistema (caso das camadas interediárias), não se conhece seu erro e_k. A solução deste problema é conseguida pela chamada Regra Delta Generalizada ou Retropropagação de Erro (Rumelhart *et al.* 1986b).

2.5. Natureza da representação distribuída

Vimos na seção 2.1 que a abordagem conexionista propõe um novo nível de análise dos processos cognitivos, no qual não são tratados conceitos simbólicos. A arquitetura radicalmente nova que resulta da idéia de representação distribuída (Hinton *et al.* 1986) torna extremamente difícil compreender em termos simbólicos as operações internas de uma rede neural (Dyer 1988).

Imaginemos um Perceptron multicamadas como o da Figura 2.2. Imaginemos ainda que tenhamos aplicado treinamento supervisionado ao sistema, conseguindo bom desempenho em algum problema complexo (podemos tomar como exemplo Nguyen & Widrow 1989, onde redes como esta são aplicadas com sucesso no problema de controlar a ação de estacionamento de um caminhão). Estamos de posse de um sistema cujo desempenho supera o de tentativas anteriores de desenvolver algoritmos para a solução do mesmo problema. A rede de fato adquiriu conhecimento implícito nos exemplos fornecidos. Este conhecimento, infelizmente, não pode ser resgatado em uma forma elaborada ou compreensível.

Tal conhecimento está, de alguma forma, codificado nas matrizes de interconexão. Os componentes desta matriz estão sem dúvida relacionados à ação de excitação ou inibição de um neurônio sobre outro. Mas qual o significado do valor de saída dos neurônios das camadas intermediárias? Seu número é em geral definido por tentativa e erro. A eles não é atribuído significado algum em termos de conceitos do domínio do problema. O que temos é uma caixa preta que aprendeu a solucionar um problema complexo.

A princípio este é o resultado direto da abordagem adotada: se vamos conduzr o processamento no nível subsimbólico, não vamos dispor de descrições simbólicas com facilidade. Mas, como afirmamos anteriormente, a proposta conexionista deve ser capaz de mostrar como os conceitos simbólicos são manipulados, partindo-se de uma implementação subsimbólica de processamento (Smolensky 1988). Não atentar para esta questão é afastar-se da maneira humana de raciocinar, uma das inspirações desta proposta.

Quando adquirimos conhecimento, elaboramos também alguma maneira de descrevê-lo. Mesmo quando o conhecimento adquirido é de caráter individual ou intuitivo, caso em que uma descrição completa, formal e precisa no nível conceitual não é possível (Smolensky 1988), uma descrição aproximada é conseguida, geralmente empregando conceitos vagos ou mal definidos.

A distância entre as abordagens simbólica e subsimbólica precisa ser diminuída para que possamos aproveitar os progressos obtidos com ambas (Minsky 1988, McMillan *et al.* 1992). Esta distância cria dificuldades para que redes neurais sejam empregadas nos níveis de processamento mais elevados de um sistema inteligente, ficando restritas a tarefas como percepção e controle (Fodor & Pylyshyn 1988). Acredita-se que tarefas de mais alto nível exijam a manipulação de análogos às estruturas simbólicas (Dyer 1988).

Em termos mais práticos também há enormes vantagens na aproximação entre as abordagens. A técnica de treinamento supervisionado descrita na seção anterior, apesar do sucesso alcançado nas aplicações, apresenta vários problemas: é um método de gradiente, e pode por isso convergir para mínimos locais de valor muito elevado na superfície de erro (Kosko 1988); apresenta grande sensibilidade à ordem dos exemplos apresentados, criando dificuldades para o treinamento "on line" (Jacobs *et al.* 1990); não permite outra forma de passagem de conhecimento ao sistema além da exibição de exemplos. É raro conseguir traduzir o conhecimento existente em termos de valores ideais para as interconexões, e mesmo quando isto é possível nem sempre o sistema, ao seguir o gradiente de erro, aproveita a sugestão (Omlin & Giles 1992). A possibilidade de trocar conhecimento de alto nível com um sistema conexionista não apenas pode facilitar o treinamento como ainda permitiria a modularização do projeto, essencial para a construção de sistemas complexos (Waltz & Feldman 1988). Se as operações dos módulos são bem compreendidas, sua integração é facilitada.

MacLennan (1991) esboça um sistema matemático para interpretação do conhecimento conexionista. Chamado *simulacrum* pelo autor, este sistema lida com "imagens", um conceito análogo ao de fórmulas no cálculo simbólico. Estas imagens formam espaços métricos conexos. O mapeamento entre espaços de imagens é contínuo, mas pode, em alguns casos, ser representado aproximadamente por regras discretas.

Outros autores buscam a interpretação das operações de uma rede neural pelo caminho da estruturação do sistema. Se grupos de neurônios puderem ser identificados com estruturas de função conhecida, estaremos caminhando no sentido de compreender o funcionamento do sistema como um todo. Descreveremos a seguir uma arquitetura estruturada de rede e veremos como sua estrutura leva à interpretação do sistema como um conjunto de regras.

2.6. Arquiteturas conexionistas modulares

Jacobs *et al.* (1990) propõem um sistema conexionista modular, citando como vantagem, além de uma interpretação mais fácil das operações, uma maior velocidade de aprendizado pela decomposição de tarefas. A figura 2.4 abaixo mostra um esquema da arquitetura proposta :

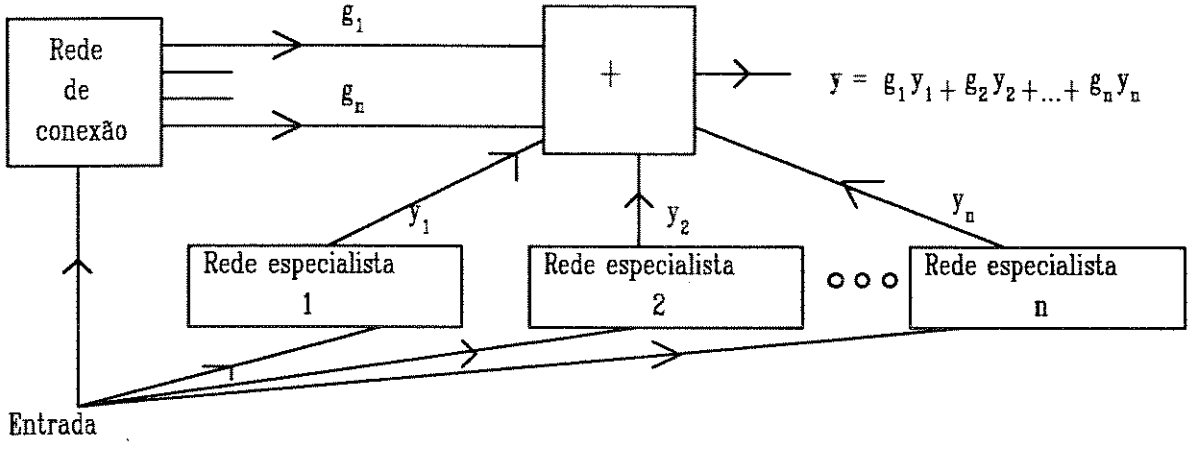


Figura 2.4: Arquitetura modular de Jacobs *et al.* (1990)

Nesta arquitetura, o vetor de entrada é apresentado ao mesmo tempo a um conjunto de n redes especialistas ("expert networks") e a uma rede de conexão ("gating network"). A rede de conexão computa valores que definem o peso da saída de cada rede especialista na saída global do sistema.

A idéia por trás desta estrutura é a de que, seguindo um algoritmo de treinamento específico, a rede de conexão seja capaz de indicar, dado um vetor de entrada, qual a rede especialista mais apropriada para gerar a resposta desejada. A intenção dos autores foi criar um sistema que busca a decomposição natural do problema apresentado, com sub-redes se especializando em domínios diferentes.

O algoritmo de treinamento proposto, tanto para a rede de conexão como para as redes especialistas, é o mesmo apresentado anteriormente. O que diferencia a atualização de pesos nos dois tipos de redes é a definição da função de erro. O treinamento das redes especialistas procura minimizar o erro quadrático total do sistema com relação ao vetor de saída. Para a rede de conexão, precisamos definir valores desejados para a saída g_i. Estes valores são definidos da seguinte forma: a cada nova iteração, é feita uma comparação do erro global atual do sistema com uma média histórica de erro. Se o erro atual é considerado significativamente menor, a rede especialista que melhor aproximou a saída desejada deve ser incentivada. Neste caso, o valor desejado na saída do neurônio correspondente na rede de conexão é 1, e dos demais é 0. Se o erro atual não é significativamente menor, nenhuma rede especialista deve ser favorecida, e o valor desejado na saída de todos os neurônios da rede de conexão é 1/n. A função de erro adotada para esta rede inclui termos adicionais que favorecem a tomada de valores binários pelos neurônios e um limite na soma dos mesmos, que força a competição entre as redes especialistas.

Um dos resultados interessantes conseguidos pelos autores foi a atenuação do chamado "crosstalk" espacial, uma degradação da performance da rede que ocorre quando os padrões de treinamento vêm consecutivamente da mesma região do espaço de entrada. Cacciatore & Nowlan (1994) apresentam uma extensão desta arquitetura, dedicada ao controle de sistemas dinâmicos, com bons resultados no controle de sistemas não-lineares com múltiplos modos de operação.

É fácil observar uma estrutura de regras por trás da arquitetura acima. A rede de conexão representa as condições destas regras, e cada rede especialista produz a ação correspondente. Seguindo este raciocínio, McMillan *et al.*(1992) propõem a arquitetura "rulenet". Esta rede neural, aplicada ao problema de identificar funções de transformação

de "strings", consegue não apenas identificar o "string" de saída em cada caso, como também exibir naturalmente em sua estrutura as regras condição-ação que geram o mapeamento. A principal diferença introduzida em relação ao trabalho original de Jacobs et. al. é o emprego de uma técnica que os autores denominaram Jogo Conexionista do Cientista ("Connectionist Scientist Game") . Nela, inicia-se o treinamento com apenas uma rede especialista (uma regra). Novas redes especialistas vão sendo periodicamente adicionadas para lidar com os padrões ainda não cobertos. O nome dado refere-se ao processo, presente na atividade científica, de criar e confirmar hipóteses induzidas pelos fenômenos estudados.

Os trabalhos discutidos acima representam passos na direção de uma melhor compreensão das operações internas de redes neurais. Outro passo importante é buscar um formalismo mais flexível para representar estas operações. A chamada lógica nebulosa parece servir bem a este propósito, como procuraremos deixar claro no capítulo seguinte.

Capítulo 3

Conjuntos nebulosos

3.1. Origem e definições

A lógica sempre foi estudada em Inteligência Artificial como um formalismo para representação do conhecimento. A representação obtida é, no entanto, inflexível demais para lidar com conceitos do mundo real sem exigir um nível elevado de simplificação. Na lógica clássica lidamos com um mundo de clareza, rigor e precisão no qual conceitos reais não se encaixam com facilidade.

A habilidade humana para manipular conceitos imprecisos é vista como a principal diferença entre inteligência humana e artificial (Zadeh 1977). A lógica nebulosa pretende formalizar a habilidade humana de chegar a decisões razoáveis baseadas em dados imprecisos e qualitativos. Com ela procura-se tratar a imprecisão e o caráter vago dos processos mentais humanos (Gupta 1977).

Um artigo de Zadeh(1965), que propõe uma extensão da teoria clássica de conjuntos, costuma ser citado como ponto de partida das pesquisas sobre lógica nebulosa, ainda que estas relacionem-se com as lógicas multivaloradas, cuja origem é bem mais antiga (Rescher 1969). Na apresentação que se segue, estudaremos conceitos de maior emprego em aplicações práticas. No entanto, devemos ressaltar que este assunto inspira diferentes abordagens. Sempre que necessário, citaremos os caminhos alternativos, que fogem ao escopo do nosso trabalho. Os conceitos serão apresentados, quando for possível, como extensão de seus equivalentes da teoria clássica, o que nos parece facilitar a compreensão.

Um conjunto nebuloso (Zadeh 1965) é caracterizado por uma função de pertinência que toma valores no intervalo [0,1]. Em outras palavras, é um conjunto no qual o conceito clássico de pertinência (sim/não) é substituído pela idéia de grau de pertinência. Desta maneira, podemos caracterizar de forma mais razoável certos conjuntos nos quais a fixação de limites nítidos de pertinência é muito artificial. Como exemplo, podemos citar o conjunto das pessoas altas em uma população, ou o conjunto dos números pequenos em um intervalo.

Operações

A extensão das operações de união e interseção para conjuntos nebulosos é feita usando-se as chamadas normas triangulares (interseção) e conormas triangulares (união). Explicitamente, se A e B são conjuntos nebulosos em um universo de discurso U e $\mu_A(x)$ é o valor da função de pertinência do conjunto A no ponto $x \in U$, temos

$$\mu_{A \cup B}(x) = \mu_A(x) \oplus \mu_B(x)$$
$$\mu_{A \cap B}(x) = \mu_A(x) * \mu_B(x)$$

onde * representa alguma norma triangular e $\oplus$ alguma conorma. O quadro abaixo apresenta algumas destas operações.

Normas Triangulares (x*y)	Conormas Triangulares (x⊕y)
min(x,y)	max(x,y)
x.y	x+y-x.y (soma algébrica)
max(0,x+y-1) (Produto limitado)	min(1,x+y) (soma limitada)
$\begin{cases} x, \text{ se } y = 1 \\ y, \text{ se } x = 1 \\ 0, \text{ se } x,y < 1 \end{cases}$ (Produto drástico)	$\begin{cases} x, \text{ se } y = 0 \\ y, \text{ se } x = 0 \\ 1, \text{ se } x,y > 0 \end{cases}$
	max[min(x.1-y),min(1-x,y)] (soma disjunta)

Quadro 3.1: Algumas normas e conormas triangulares

É fácil notar que estas operações concordam com a união e a interseção clássicas quando as funções de pertinência tomam valores 0 e 1. Mais do que isto, estas operações satisfazem dois requisitos fundamentais para união e interseção (Gaines 1977) :

$$\mu_{A \cup B}(x) \geq \max(\mu_A(x), \mu_B(x)) \forall x \in U$$
$$\mu_{A \cap B}(x) \leq \min(\mu_A(x), \mu_B(x)) \forall x \in U$$

Portanto, a união e interseção de conjuntos nebulosos podem ser realizadas de várias maneiras, e a escolha de uma delas é normalmente guiada por procedimentos de tentativa e erro ou por considerações de custo computacional (estas evidentemente levam a uma preferência pelo máximo para união e mínimo e produto para interseção). Voltaremos a este assunto na discussão dos controladores nebulosos.

O complemento de um conjunto nebuloso é usualmente definido segundo

$$\mu_{\neg A}(x) = 1 - \mu_A(x)$$

Há outras funções f(x) que satisfazem os requisitos básicos da operação complemento:

$x \geq y \rightarrow f(x) \leq f(y)$ e $f(f(x))=x$ (Gaines 1977),

mas não encontramos na literatura aplicações práticas em que definições diferentes fossem usadas.

Produto cartesiano

Dados os conjuntos nebulosos $A_1, A_2, ..., A_n$, definidos respectivamente em $U_1, U_2, ..., U_n$, o produto cartesiano $A_1 \times A_2 \times ... \times A_n$ é um conjunto nebuloso em $U_1 \times U_2 \times ... \times U_n$, caracterizado pela função de pertinência

$$\mu_{A_1 x A_2 x ... x A_n}(x) = \min(\mu_{A_1}(x), \mu_{A_2}(x), ..., \mu_{A_n}(x))$$

Lembrar que no caso clássico, $(x,y) \in UxV$ se e somente se $x \in U$ e $y \in V$, o que leva imediatamente à extensão acima para o caso nebuloso.

Relação nebulosa

Uma relação nebulosa n-ária é um conjunto nebuloso no espaço $U_1 \times U_2 \times ... \times U_n$, expressa como

$$R_{U_1xU_2x...xU_n} = \{ [(u_1, u_2, ..., u_n), \mu_R(u_1, u_2, ..., u_n)] \mid (u_1, u_2, ..., u_n) \in U_1xU_2x...xU_n \}$$

Ou seja, atribui-se a cada ponto do espaço $U_1 \times U_2 \times ... \times U_n$ um número do intervalo [0,1] indicando o quanto a relação R é válida para os valores $u_1, u_2, ..., u_n$. Como exemplo, uma relação binária pode ser usada para indicar os pares (x,y) onde vale a relação "x é muito maior do que y".

Composição de relações

Se R e S são relações nebulosas em UxV e VxW, respectivamente, a composição de R e S é outra relação nebulosa, definida pela regra "Sup-estrela" (Zadeh 1973):

$$R \circ S = \{ [(u,w), \sup(v \in V)(\mu R(u,v) * \mu S(v,w)] , u \in U, w \in W \}$$

Observar que, se as funções de pertinência tomam valores em {0,1}, a regra anterior é equivalente a definir que a relação RoS vale para o par (u,w) se e somente se existe algum $v \in V$ tal que (u,v) esteja incluído na relação R e (v,w) na relação S.

Regra composicional de inferência

No caso clássico, se elementos dos conjuntos U e V são ligados pela relação R e A é um subconjunto de U, podemos formar o conjunto $B \subset V$ tomando todos os pontos $v \in V$ tais que, para algum $u \in A$, (u,v) esteja em R. A generalização deste conceito leva à regra composicional de inferência (Zadeh 1973):

Se x é um conjunto nebuloso em U e R é uma relação nebulosa em UxV, x induz por R um conjunto nebuloso y em V dado por:

$$y = x \circ R$$

ou seja,

$$\mu_y(v) = \sup_{(u \in U)}(\mu_x(u) * \mu_R(u,v))$$

Lógica nebulosa

A definição dos conceitos da chamada lógica nebulosa a partir da extensão da teoria clássica de conjuntos costuma ser apresentada como um simples trabalho de associação entre conceitos da teoria de conjuntos e conceitos de lógica. Assim, associa-se o conceito de pertinência ao de valor-verdade de uma proposição lógica, o de união, interseção e complemento de conjuntos às operações *ou*, *e* e *não*, respectivamente; a inferência é vista como uma relação entre conjuntos.

Estas associações estão plenamente justificadas no campo da lógica clássica, porque é conhecido o isomorfismo entre álgebra de Boole e a estrutura do cálculo proposicional clássico. No campo da teoria dos conjuntos nebulosos, as inúmeras maneiras de estender as operações clássicas trazem dificuldades quanto à maneira mais adequada e rigorosa de apresentar os fundamentos da lógica nebulosa. Os trabalhos no campo de engenharia não costumam preocupar-se com esta questão. Considera-se que as associações feitas são razoáveis e, principalmente, levam a bons resultados.

Regras nebulosas, raciocínio aproximado e teoria da possibilidade

Regras (ou afirmações condicionais) são formas eficientes de formalizar conhecimento nos sistemas de inteligência artificial (Nilsson 1980). O emprego de elementos da teoria dos conjuntos nebulosos permite a manipulação mais flexível de regras, com inferências sobre condições não totalmente satisfeitas ou não claramente definidas. Um exemplo de regra que pode ser tratada desta forma é : "Se o alvo está próximo, a velocidade deve ser diminuída". Este tipo de inferência, denominada raciocínio

aproximado (Zadeh 1975) procura reproduzir a capacidade humana de lidar com imprecisão e incerteza, e estabelece pontos de contato entre teoria de conjuntos nebulosos e probabilidade.

De fato, podemos dar à função de pertinência de um conjunto nebuloso uma interpretação que torna este relacionamento mais claro (Zadeh 1978). Imaginemos que uma variável X tome valores em um universo de discurso U. Uma afirmação do tipo "X é F", onde F é representado por um conjunto nebuloso sobre U, pode ser vista como uma restrição nebulosa aos valores de X. Por exemplo, o fato "João é alto" cria restrições (não rígidas) sobre os valores aceitáveis para a sua altura. Diz-se então que a função de pertinência μ_F representa a distribuição de possibilidades associada a X.

A palavra "possível", em linguagem natural, tem basicamente dois significados (Dubois & Prade 1989): um físico ("factível") e um lógico ("compatível com a informação disponível"). Na definição apresentada acima, este conceito origina-se não de freqüências de eventos (probabilidade) mas da tradução de restrições suaves sobre variáveis em termos de conjuntos nebulosos (Dubois & Prade 1986). No entanto, probabilidade e possibilidade não são totalmente independentes. Um evento de possibilidade reduzida não pode ter elevada probabilidade e, no caso limite, um evento impossível tem necessariamente probabilidade nula. Podemos então requerer que graus de probabilidade sejam um limite inferior para o grau de possibilidade (princípio da consistência - Zadeh 1978).

Eventos mutuamente excludentes podem ter simultaneamente graus elevados de possibilidade. Assim, graus de possibilidade não são aditivos, ao contrário dos de probabilidade. Vale notar que a noção de que graus subjetivos de certeza não são aditivos é bastante antiga (Dubois & Prade 1986).

Estes conceitos intuitivos podem ser formalizados com a definição de medidas e integrais nebulosas (Sugeno 1977). Em uma medida nebulosa, o requerimento de aditividade

$$m(\cup A_j) = \sum m(A_j) \ \forall \text{ classe disjunta } \{A_j\}$$

é substituído pela condição (mais fraca) de monotonicidade :

$$A \subseteq B \rightarrow \mathrm{m}(A) \leq \mathrm{m}(B)$$

Ainda há uma enorme polêmica sobre o relacionamento exato entre probabilidade e conjuntos nebulosos, e sobre as vantagens e desvantagens destas ferramentas no tratamento da incerteza (Laviolette & Seaman Jr. 1994, Dubois & Prade 1994).

3.2. Controladores nebulosos

Com o uso de regras nebulosas, torna-se possível expressar as ações de um operador humano ao controlar sistemas de dinâmica complexa e/ou mal conhecida. Esta abordagem leva à construção dos chamados controladores nebulosos, uma das aplicações de maior sucesso da lógica nebulosa. Apresentaremos a seguir os conceitos fundamentais e resultados obtidos com controladores nebulosos (seguindo basicamente a exposição de Lee 1990)

Variáveis lingüísticas

Como dissemos acima, o controlador nebuloso emprega regras que manipulam conceitos vagos e mal definidos. As variáveis a que estas regras fazem referência tomam valores "lingüísticos", isto é, rótulos associados a conjuntos nebulosos. Um exemplo de variável lingüística poderia ser a velocidade, tomando os valores "baixa", "média" e "alta".

Esquema geral do controlador

A figura 3.1 a seguir apresenta o esquema geral do controlador nebuloso (Lee 1990). Detalharemos a seguir suas partes constituintes.

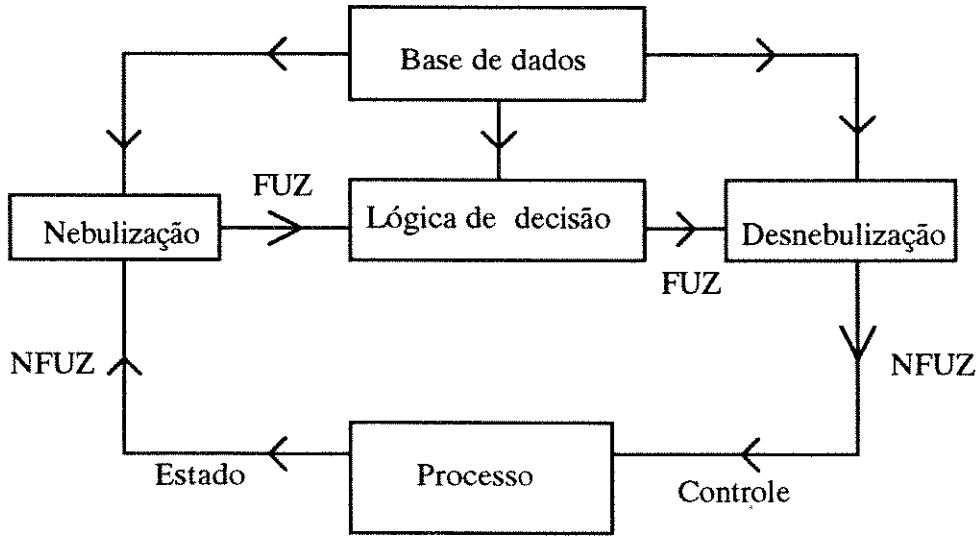


Figura 3.1 - Esquema geral do controlador nebuloso (Lee 1990)

Nebulização

As variáveis de entrada normalmente são numéricas, mas o controlador trabalha internamente com conjuntos nebulosos. O processo de conversão de um valor numérico em um conjunto nebuloso recebe o nome de *fuzzification*. O procedimento mais adotado é a criação de um *singleton*, isto é, um conjunto nebuloso A com função de pertinência:

$$\mu_A(x) = \begin{cases} 0, \forall x \neq x_0 \\ 1, x = x_0 \end{cases}$$

onde x_0 é o valor numérico da variável. Em outras palavras, o valor x_0 passa a ser conceitualmente considerado um conjunto nebuloso.

Base de dados

O conjunto de regras do controlador pode ser representado genericamente por:

If X_1 is A_1 and X_2 is B_1 and ... and X_n is C_1 then y is D_1.

also If X_1 is A_2 and X_2 is B_2 and ... and X_n is C_2 then y is D_2.

also etc.

onde X_i são variáveis (lingüísticas) de estado do processo, y_i é uma variável controlada do processo, A_i, B_i, D_i são termos lingüísticos representados por conjuntos nebulosos e i é um índice genérico. Um exemplo de regra nebulosa de controle:

"Se a pressão é muito alta, a válvula deve ficar um pouco aberta."

"pressão" e "posição da válvula" são variáveis lingüísticas, "muito alta" e "pouco aberta" são exemplos de termos das respectivas variáveis, definidos como conjuntos nebulosos nos respectivos domínios.

A maneira mais comum de construir tais regras é procurar expressar a ação de operadores humanos. No entanto, a descrição lingüística da dinâmica do sistema pode ser tomada como um modelo (nebuloso) do mesmo, e as regras podem ser obtidas utilizando-se técnicas de engenharia de controle (Czogala & Pedrycz 1982). Também foram desenvolvidos algoritmos para correção automática de parâmetros das regras (aprendizado) baseada em índices de performance (Xu & Lu 1987, Shao 1988).

A definição do número de conjuntos nebulosos a serem construídos para caracterizar cada variável (partição do espaço de entrada) é feita por tentativa e erro. Deseja-se que o controlador infira ações de controle corretas em todas as situações, o que implica na completude da base de regras, isto é, os conjuntos definidos devem cobrir todo o espaço. Outra questão complexa é a da interação entre as regras (Czogala & Pedrycz 1981): imaginemos uma regra

Se X_1 é A_1 então y é B_1,

entre outras de um sistema. Ainda que X_1 seja idêntico a A_1, y pode diferir de B_1, pela influência de outras regras cujas condições estejam parcialmente satisfeitas.

Implicação nebulosa

Dada a regra nebulosa "se A então B", que conclusão pode ser inferida de uma condição A', semelhante a A?. Este tipo de raciocínio aproximado é efetuado definindo-se $A \rightarrow B$ como uma relação nebulosa (Zadeh 1973) e aplicando-se em seguida a regra de inferência composicional apresentada anteriormente:

$$B' = A' \circ R_{A \rightarrow B}$$

Na literatura estão descritas cerca de 40 relações de implicação. Algumas delas estão indicadas no quadro 3.2 abaixo:

1. Mínimo (Mamdani)	$\mu_R(x,y) = \min(\mu_A(x),\mu_B(y))$
2. Produto (Larsen)	$\mu_R(x,y) = \mu_A(x).\mu_B(y)$
3. Regra aritmética ou de Lukaziewicz (Zadeh)	$\mu_R(x,y) = \min(1,1-\mu_A(x)+\mu_B(y))$
4. Regra Maxmin (Zadeh)	$\mu_R(x,y) = \max[\min(\mu_A(x),\mu_B(y)), 1-\mu_A(x)]$
5. Seqüência padrão	$\mu_R(x,y) = \begin{cases} 1, \mu A(x) \geq \mu B(y) \\ 0, \text{ cc} \end{cases}$
6. Booleana	$\mu_R(x,y) = \max(1-\mu_A(x),\mu_B(y))$

Quadro 3.2 - Algumas relações de implicação nebulosa

Algumas destas definições têm um embasamento lógico bastante claro. As de número 3 e 6, por exemplo, podem ser entendidas a partir da equivalência

$$A \rightarrow B \equiv \neg A \vee B$$

usando soma limitada e máximo, respectivamente, como operadores *ou*. Já a número 4 vem de

$$A \rightarrow B \equiv (A \wedge B) \vee \neg A$$

(esta expressão é equivalente à anterior no cálculo proposicional clássico, mas as duas não apresentam os mesmos resultados em lógica nebulosa para qualquer escolha de conectivos).

Outras definições, tais como as de número 1, 2 e 5 são de difícil justificativa lógica.

A escolha da relação de implicação é feita, assim como a dos operadores de união e interseção, sem o apoio de resultados teóricos que indiquem sempre o melhor caminho. A maior dificuldade destas questões é a de que o aparato da teoria dos conjuntos nebulosos é aplicado sobre conceitos intuitivos e mal definidos (afinal, era isto que se queria), tornando difícil estipular "sob quais condições as computações são eficientes ou mesmo justificadas" (Fox 1981).

Baldwin&Pilsworth(1980) propõem um conjunto de propriedades a serem satisfeitas pela relação de implicação para que requisitos intuitivos do raciocínio aproximado sejam alcançados. São elas:

- *Propriedade fundamental* : a conclusão não pode ser mais restritiva do que a premissa.
- *Suavidade*: pequenas variações na premissa acarretam pequenas variações na conclusão.
- *Inferência irrestrita*: sempre que o antecedente for falso, nenhuma restrição pode ser inferida sobre o conseqüente.
- *Simetria entre Modus Ponens e Modus Tollens*: o uso da implicação $A \rightarrow B$ e do fato A' deve produzir resultados simétricos, na inferência de B', aos do uso da implicação $\neg B \rightarrow \neg A$ e do fato $\neg B'$ na inferência de $\neg A'$. Esta característica não precisa ser avaliada no caso específico de controladores nebulosos, já que nestes o Modus Tollens não é utilizado.
- *Propagação de nebulosidade* (as conclusões devem se tornar menos restritivas à medida em que se avança em uma cadeia de implicações).

Os autores verificaram que a relação de número 3 (quadro 3.2) satisfaz todas as condições. A número 4 não satisfaz nenhuma delas. Outras, como a de número 5, apresenta alguns dos pré-requisitos mas não todos.

Lee (1990) investigou principalmente o efeito que a modificação da premissa com operadores lingüísticos ("muito", "pouco", etc.) tem sobre o conseqüente, para verificar se a implicação funcionava de acordo com nossa intuição. Chegou a resultados praticamente opostos aos relatados acima, concluindo, por exemplo, que a relação 3 é

inadequada. As relações 1 e 2, apesar de não apresentarem estrutura lógica bem definida, pareceram adequadas, segundo o autor, para o emprego em controladores.

Lógica de decisão

Uma vez compreendidos todos os conceitos envolvidos, podemos apresentar as operações efetivamente realizadas no controlador. Vamos assumir, por simplicidade, que o controlador opere com duas regras nebulosas:

If x is A_1 and y is B_1 then z is C_1

also If x is A_2 and y is B_2 then z is C_2

x e y são variáveis de entradas (tratadas como *singletons* nebulosos), z o sinal de controle e Ai, Bi, e Ci são conjuntos nebulosos adequadamente definidos.

Em geral, o antecedente das regras é interpretado como uma conjunção nebulosa no produto cartesiano dos universos de discurso sob os quais estão definidas as variáveis envolvidas. Por exemplo, o antecedente da primeira regra acima seria representado por um conjunto nebuloso com função característica

$$\mu_{R_i}(u,v) = \mu_{A_i}(u) * \mu_{B_i}(v)$$

Observar que os conjuntos A_1 e B_1 podem ser definidos sobre universos diferentes.

Definindo o antecedente desta forma, dados os valores (*singletons*) $x=x_0$ e $y=y_0$, usando-se o operador mínimo como relação de implicação e usando-se a regra sup-min para inferência, prova-se (Lee 1990) que o conjunto nebuloso C'i (conseqüente da regra i) é dado por

$$\mu_{C'_i}(w) = \min(\alpha_i, \mu_{C_i}(w)) \qquad (3\text{-}1)$$

onde $\alpha_i = \min(\mu_{A_i}(x_0) * \mu_{B_i}(y_0))$ pode ser interpretado como o grau de concordância do antecedente da i-ésima regra com os dados (denominaremos este valor "função de pertinência" da regra i).

Desta forma temos dois conjuntos nebulosos C'_1 e C'_2, referentes à mesma variável z, mas inferidos a partir de regras diferentes. A combinação das duas inferências para se construir uma conclusão única C' é função do operador *also*. Uma característica indispensável para este conectivo é a comutatividade, já que não desejamos que a ordem das regras tenha qualquer influência. As normas e conormas triangulares (quadro 3.1) têm esta propriedade, e são as funções mais utilizadas para este fim. É evidente que a escolha entre uma norma ou conorma tem enorme impacto na qualidade do sistema (Kiszka *et al.* 1985, Lee 1990) e é mais um parâmetro importante a ser determinado.

Kiszka *et al.* (1985) fazem uma avaliação das diferentes implicações nebulosas do ponto de vista prático. É verificada a eficiência de um modelo nebuloso de motor D.C. com diferentes relações de implicação (26 ao todo), utilizando-se, para cada relação, máximo ou mínimo como operador *also* (totalizando 52 tipos de inferência). Observa-se que o desempenho de uma relação de implicação é totalmente alterado pela mudança do operador *also*. Por exemplo, a relação número 3 (quadro 3,2) apresenta ótimo desempenho com o operador *also* definido como mínimo, mas o mesmo não acontece quando se usa a função máximo.

Além da relação 3 com mínimo, a número 1 com máximo também tem resultados satisfatórios e, pela sua simplicidade em termos computacionais, acaba sendo a mais utilizada para inferência nebulosa. Voltando ao nosso exemplo, se o operador max é usado como *also*, o conseqüente final é dado por:

$$\mu_{C'}(w) = \max_i(\mu_{C_i}(w))$$

Sendo C' um conjunto nebuloso, é necessário um procedimento (*defuzzification*) para chegarmos ao valor numérico do sinal de controle. Foram propostas alterações nos conseqüentes para dispensar esta etapa. Tsukamoto (1979) propõe o uso

de conjuntos C com função de pertinência monotônica. Assim, podemos inferir da i-ésima regra o valor numérico z_i tal que

$$\mu C'i(zi) = \min(\alpha i, \mu Ci(zi))$$

A saída global z_0 é a média dos z_i's ponderada pelos valores α_i. Takagi & Sugeno (1985) propõem regras com conseqüente linear :

If x is A_1 and y is B_1 then z is $k_1 x + k_2 y + k_3$

O valor z_0 final é obtido por média ponderada, como no método de Tsukamoto.

Desnebulização

Se a saída do processo de inferência é um conjunto nebuloso C', temos o problema de determinar a saída z_0 que melhor representa a distribuição de possibilidade $\mu C'(z)$. Os dois métodos mais empregados são o de média dos máximos (z_0 é a média dos valores de z que são máximos de $\mu C'(z)$) e o de centro de área. Neste último, z_0 é dado por :

$$z0 = \frac{\sum_{j=1}^{n} \mu C'(zj).zj}{\sum_{j=1}^{n} \mu C'(zj)}$$

Dificuldades e aplicações práticas

No começo desta seção foi dito que no projeto de controladores nebulosos, o conhecimento de operadores humanos experientes pode ser aproveitado. Deve-se dizer, no entanto, que este é apenas o ponto de partida. O ajuste das funções de pertinência dos conjuntos nebulosos envolvidos e a escolha dos operadores costuma levar a um longo trabalho de tentativa e erro. A transferência das estratégias de decisão de especialistas e operadores humanos para estes sistemas não é tarefa fácil, mesmo com a flexibilidade apresentada por regras nebulosas (Lee 1990). Esta dificuldade, aliada ao fato de serem recentes os resultados teóricos existentes quanto à estabilidade e robustez destes controladores, faz com que alguns especialistas tenham restrições aos sistemas nebulosos de controle, apesar dos bons resultados obtidos.

Lee (1990) cita numerosas aplicações em controle com emprego de algumas das técnicas apresentadas acima. Destacamos: controle de aquecedores, de sinais de trânsito, de tráfego aéreo, robótica, controle automático de velocidade e da transmissão em automóveis, controle de reatores nucleares. A empresa Hitachi Ltd. desenvolveu um sistema de operação automática de trens baseado em lógica nebulosa, que vem sendo usado no metrô de Sendai (Japão) desde 1987.

Uma aplicação interessante encontrada na literatura é o controle de direção de veículo proposto por Sugeno & Nishida (1985). O problema é definir a direção a ser seguida por um carro mantido a velocidade constante em um percurso sinuoso, dadas a direção atual e as distâncias para os limites da pista. São empregadas regras nebulosas com conseqüentes lineares, como visto anteriormente. A partição do espaço de entrada (número de conjuntos nebulosos) foi arbitrariamente fixada. Então, a partir de exemplos do percurso do carro quando guiado por operadores humanos, foram identificados os parâmetros dos conseqüentes de 20 regras. Os autores apresentam bons resultados, não apenas em simulações, mas no controle efetivo de um pequeno modelo a baixa velocidade.

3.3. Redes neurais e lógica nebulosa

Procuramos, na teoria dos conjuntos nebulosos, um formalismo adequado para uma descrição aproximada das operações internas de uma rede neural. É interessante observar neste ponto as semelhanças e diferenças existentes entre esta teoria e a de redes, e discutir que vantagens podemos esperar da combinação das duas.

Tanto redes neurais quanto lógica nebulosa se propõem a desenvolver computacionalmente habilidades típicas do raciocínio humano, diminuir a diferença entre inteligência humana e artificial (Zadeh 1977), entender "o que torna as pessoas mais espertas do que as máquinas" (McClelland *et al.* 1986). As abordagens são, no entanto, bem diferentes.

A teoria dos conjuntos nebulosos tornar formalismos tradicionais (conjuntos, relações, regras, etc.) tolerantes à imprecisão e adequados ao processamento de informações vagas (Zadeh 1973). Já as redes neurais surgem fundamentalmente de considerações quanto à arquitetura do processamento e quanto ao nível de análise dos processos cognitivos, como visto no Capítulo 2. As abordagens diferentes não nos impedem de identificar vários pontos de contato entre as teorias.

Fundamentalmente, em ambas está presente a idéia de substituir símbolos discretos por quantidades contínuas. Ambas enfatizam o processamento numérico de informações (Kosko 1992). Os trabalhos que procuram analisar a estrutura das operações internas das redes neurais indicam em vários momentos pontos de contato com lógica nebulosa. A começar por MacLennan (1991) (ver capítulo 2), onde se afirma que a lógica por trás dos *simulacra* é inerentemente nebulosa. Rumelhart et. al. (1986c) observam que o processsamento em certos modelos de redes neurais pode ser visto como uma busca por um estado no qual a obediência às restrições representadas pelas conexões é a maior possível. Existe, portanto, algo semelhante ao disparo simultâneo de muitas regras possivelmente conflituosas e com condições parcialmente satisfeitas.

Até mesmo a sutil relação com probabilidade e estatística é um ponto comum a redes neurais e lógica nebulosa. Vimos que a definição de conjuntos nebulosos pode ser interpretada como uma distribuição de possibilidade, um conceito subjetivo de factibilidade ou compatibilidade com a informação disponível (Zadeh 1978, Dubois & Prade 1989). Por outro lado, Smolensky(1988) afirma que um sistema sub-simbólico realiza um conjunto de inferências que, como um todo, produzem a saída que melhor se ajusta às condições de entrada "num sentido definido pelo conhecimento estatístico armazenado em suas conexões".

A expectativa de que os conceitos de lógica nebulosa constituam uma ferramenta para a análise interna de redes neurais parece confirmar-se. Além disso, as redes neurais podem contribuir muito na construção de sistemas nebulosos, principalmente no ajuste automático de parâmetros. Técnicas de aprendizado podem substituir o difícil processo de ajuste por tentativa.

Estudaremos no capítulo seguinte algumas das formas pelas quais Redes Neurais e Lógica Nebulosa podem ser combinadas nas chamadas Redes Neuronebulosas ("Neuro-Fuzzy Networks").

Capítulo 4

Redes neuronebulosas

4.1. Introdução

Estudaremos neste capítulo algumas das maneiras pelas quais lógica nebulosa e redes neurais têm sido combinadas. Podemos classificar estes sistemas híbridos em quatro grupos. No primeiro, as redes neurais são projetadas para lidar com conjuntos nebulosos ao invés de números reais (Pedrycz 1994). Em outro, a rede é usada como forma de implementar um sistema nebuloso já projetado (Keller *et al.* 1992). Estes dois primeiros grupos fogem ao escopo do trabalho, e não serão discutidos aqui.

As aplicações mais numerosas estão em um terceiro grupo, no qual as redes neurais têm estrutura semelhante à dos controladores nebulosos estudados anteriormente e treinamento supervisionado é aplicado para ajuste de parâmetros. Neste terceiro grupo utilizam-se neurônios cuja função de ativação é uma gaussiana, como nas chamadas Redes de Funções Radiais de Base, originadas de outras pesquisas. Finalmente, podemos usar redes neurais para gerar apenas as funções de pertinência de um sistema nebuloso.

4.2. Redes de funções radiais de base

As redes de funções radiais de base ("Radial Basis Function Networks" - RBFN's) são estruturas inspiradas no modelo biológico de campos sobrepostos de recepção, presentes em regiões do córtex cerebral. Este modelo de inspiração biológica acaba resultando em algo equivalente a um tipo de sistema nebuloso (Jang & Sun 1992) como veremos a seguir.

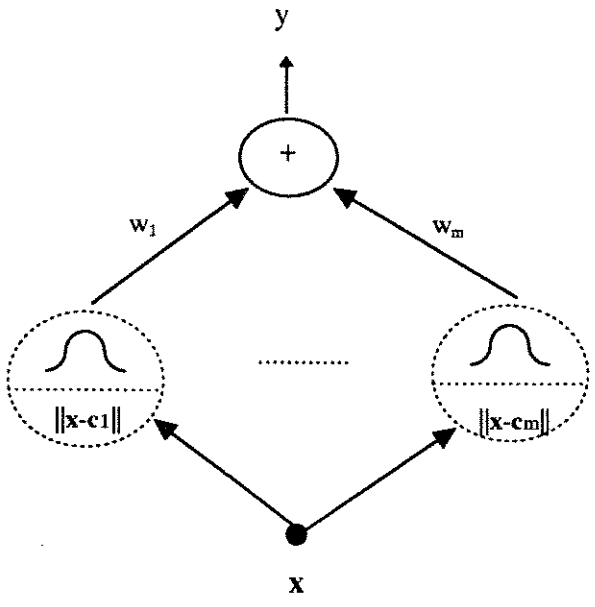


Figura 4.1: Representação de rede RBFN

A Figura 4.1 acima mostra um diagrama de RBFN. Os neurônios da 1ª camada computam funções com simetria radial, geralmente gaussianas, e a saída é uma soma ponderada dos valores da 1ª camada (eventualmente normalizada pela soma das ativações).

Agora imaginemos um sistema nebuloso com regras do tipo:

Se x_1 é A_1 e x_2 é A_2 e .... então $y=w_1$

Se x_1 é B_1 e x_2 é B_2 e .... então $y=w_2$

Neste caso, se os A_i's são gaussianas de mesmo σ (o mesmo acontecendo com os B_i's), se o produto é a operação usada como conjunção nebulosa no antecedente, e se a saída é a média dos conseqüentes, ponderada pelas funções de pertinência dos antecedentes, este sistema é equivalente a uma rede RBFN como a indicada acima (Jang & Sun 1992).

Vários trabalhos concentram-se nas RBFN's sem relação com sistemas nebulosos. Funções radiais de base formam uma possível solução para o problema de minimização do funcional (2-2) (Girosi 1992, Girosi *et al.* 1994). Estes autores propõem uma extensão desta classe, resultando em modelos do tipo:

$$f(x) = \sum_{\alpha=1}^{p} c_\alpha G(\|x - t_\alpha\|_w)$$

onde G(.) representa a função radial e $\|\mathbf{x}\|_w = \mathbf{x}^T\mathbf{W}\mathbf{x}$ é uma norma induzida pela matriz W. O caso W=I resulta nas RBF's com simetria radial na norma euclidiana. Se W é diagonal, temos o caso em que as curvas de nível destas funções são hiperelipses (e não mais hiperesferas) e para W arbitrário o eixo destas hiperelipses não coincidem necessariamente com o do sistema de coordenadas. Em termos do sistema nebuloso equivalente, estamos admitindo variâncias diferentes para cada componente do antecedente e também o uso de combinações lineares das variáveis de entrada.

Tomando-se os elementos da matriz W como parâmetros a ajustar, juntamente com os c_α, o sistema resultante aprende a métrica que melhor representa o relacionamento entre os dados. Deve-se notar, porém, que o número de parâmetros a ajustar aumentou, e por isso o número de funções (p) deve ser bem menor do que o de pontos do arquivo de treinamento (m), para que o erro de estimação (ver seção 1.4) não aumente.

Uma característica importante das RBFN's é a de que o valor de ativação da camada gaussiana pode ser tomado como indicação de novidade (Bishop 1994, Gentric & Withagen 1994). Baixos valores em todos os neurônios da camada, por exemplo, podem indicar a presença de um vetor de entrada **x** muito diferente daqueles com os quais a rede foi treinada. Este tipo de informação não pode ser obtida em uma rede neural convencional.

As RBFN's valorizam ainda o caráter local do treinamento (pontos distantes da área de atuação de uma unidade não influenciam o ajuste de seus parâmetros). A atualização de pesos em redes convencionais tem caráter global e costuma impedir a identificação de estruturas locais. Já o treinamento local, além de permitir análise da

representação resultante, faz com que novos aspectos do problema sejam aprendidos sem prejuízo do que já se conhece (Ye & Loh 1993). Esta característica incremental do aprendizado não é obtida no Perceptron multicamadas apresentado na seção 1.3.

4.3. Redes neurais estruturadas como controladores nebulosos

Nestes sistemas, uma rede neural é construída com estrutura semelhante à dos controladores nebulosos apresentados anteriormente, e técnicas de aprendizado são empregadas para o ajuste de parâmetros. A figura 4.2 abaixo apresenta a arquitetura básica de tais redes, objeto de vários trabalhos que estudaremos a seguir. Esta tecnologia já foi aplicada em produtos comerciais (Takagi 1993).

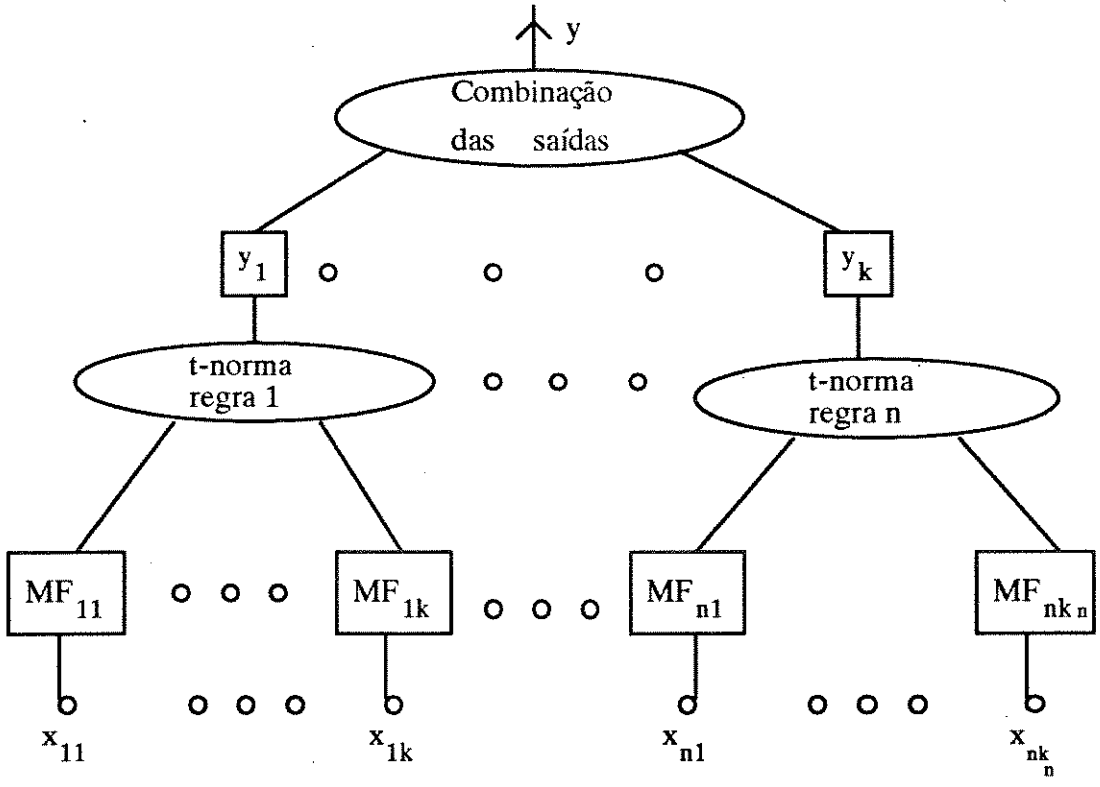


Figura 4.2: Estrutura geral de controlador neuronebuloso (Takagi 1993)

Sistema nebuloso de controle e decisão baseado em redes neurais (Lin & Lee 1991)

Neste sistema, o cálculo das funções de pertinência (MF_{ij} na figura anterior) é feito por neurônios com função de ativação gaussiana:

$$MFij(xj) = e^{\frac{-(xj-\mu ij)^2}{\sigma ij^2}}$$

Onde μ_{ij} e σ_{ij} são parâmetros a serem ajustados no treinamento. A norma triangular utilizada para a combinação das cláusulas do antecedente é o operador mínimo. A subrede que computa os valores y_k da figura 4.2 é detalhada na figura a seguir:

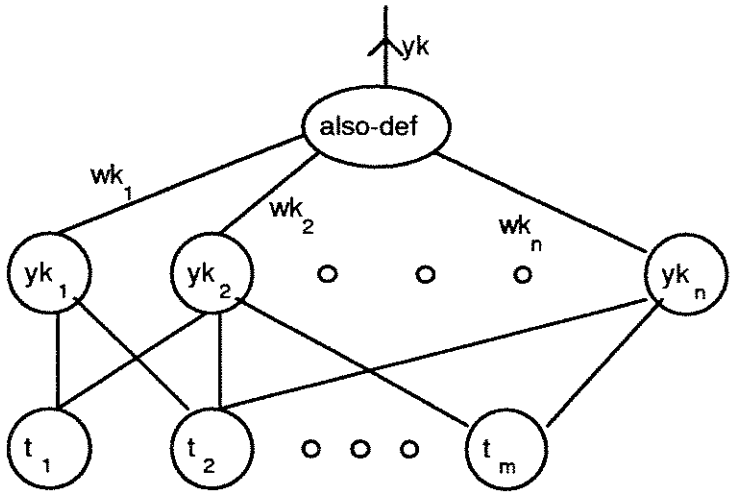


Figura 4.3: Detalhe da estrutura de conseqüente de regras do sistema de Lin & Lee (1991)

Os neurônios y_{ki} combinam, por meio de soma limitada, a saída de diferentes conseqüentes de regras que indicam ser y_{ki} o termo lingüístico adequado para a saída y_k:

$$yki = \min\left(1, \sum_j tj\right)$$

onde t_j são os neurônios de t-norma ligados a y_{ki}. Os pesos w_{ki} são dados por

$$wki = \mu ki \; . \; \sigma ki$$

onde μ_{ki} e σ_{ki} são parâmetros da sigmóide associada ao termo lingüístico y_{ki}.. O neurônio indicado por "also-def" realiza as operações *also* e *defuzzification* para se obter o valor y_k, segundo:

$$yk = \frac{\sum_{1} yki.\mu ki.\sigma ki.}{\sum_{1} yki.\sigma ki}$$

O treinamento desta rede é feito em duas etapas. Na primeira etapa, técnicas de treinamento não-supervisionado (Kohonen 1984) são usadas para definir as funções de pertinência e a estrutura da base de regras. O número de partições de cada variável de entrada e de saída é definido a priori. Esta etapa define um valor inicial para μ's e σ's, e define a existência ou não de conexão entre termos lingüísticos de entrada e regras, bem como entre regras e termos lingüísticos de saída. A segunda etapa de treinamento é supervisionada, e segue o algoritmo tradicional de retropropagação de erro (Rumelhart *et al.*1986b).

Os autores apresentam resultados positivos da aplicação desta rede no problema do carro de Sugeno & Nishida (1985) e em um problema de escalonamento em linhas de montagem. É apresentada inclusive a base de regras resultante em cada caso.

NFDS (Sistema neuronebuloso de decisão) (Bruske *et al.* 1993)

São poucas as diferenças entre este sistema, empregado no problema de reconhecimento de superfícies, e o sistema anterior. A função mínimo é substituída pelo produto na combinação dos antecedentes; é permitido o uso de sigmóides

$$S(\mu,\sigma,x) = \frac{1}{1+e^{\frac{-(x-\mu)}{\sigma}}}$$

juntamente com gaussianas na definição de funções de pertinência; não há treinamento não-supervisionado.

Os autores destacam entre seus resultados convergência mais rápida e melhor capacidade de generalização em comparação com redes neurais tradicionais. Estas vantagens são conseguidas, segundo os autores, pela inserção de conhecimento parcial sobre o problema na estrutura da rede, antes do treinamento.

ANFIS (Sistema de inferência nebulosa baseado em redes adaptativas) (Jang 1992)

Este trabalho se concentra no raciocínio nebuloso com conseqüentes lineares (Takagi & Sugeno 1985) apresentado no capítulo anterior. A parte do conseqüente é, portanto, diferente, como mostra a figura seguinte:

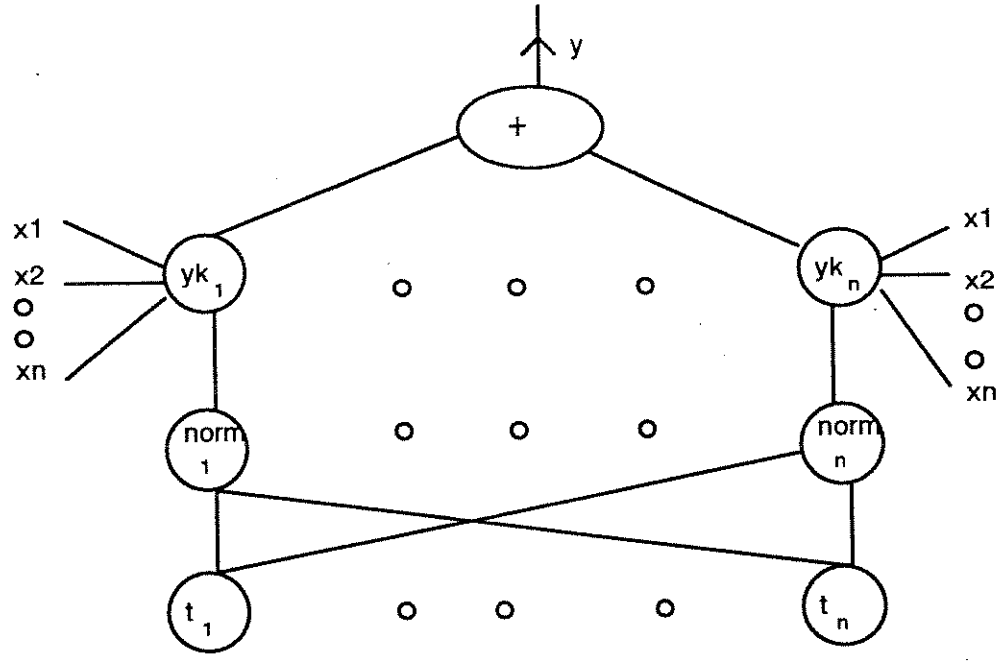


Figura 4.4: Estrutura de conseqüentes do sistema ANFIS (Jang 1992)

Também aqui usa-se o produto como *and* nos antecedentes. A força de disparo de cada regra (t_i) é normalizada em relação à soma das mesmas, e em seguida uma camada de neurônios computa o valor

$$y_i = w_i.f_i = w_i.\sum_{j=1}^{n}(k_j x_j) + k_{n+1}$$

de acordo com os conseqüentes de cada regra (w_i é a força normalizada de disparao da regra i). A saída global é a soma dos valores y_i (observar a semelhança com a arquitetura modular de Jacobs *et al.* (1990) estudada no capítulo 2).

O autor apresenta uma série de testes em problemas tais como modelamento de funções não-lineares a várias variáveis, modelamento e controle de sistemas não-lineares e previsão de séries temporais caóticas, problemas estes já tratados com redes ou sistemas nebulosos separadamente. Os resultados apresentados são superiores aos das outras abordagens.

Tresp *et al.* (1993) utilizam um sistema semelhante a este, mas desenvolveram uma técnica para eliminar automaticamente regras de menor importância ao final do treinamento. Esta idéia foi aproveitada em nosso sistema, como veremos no capítulo seguinte.

4.4. Raciocínio nebuloso orientado a redes neurais

Takagi (1993) propõe um sistema nebuloso no qual as funções de pertinência (valores αi da equação 3-1) são geradas por rede neural. O esquema é novamente muito semelhante ao da rede de Jacobs *et al.* (1990) (figura 2.4), com as redes especialistas representando conseqüentes de regras nebulosas e a rede de conexão indicando a força de disparo de cada regra para a entrada apresentada.

A característica especial deste sistema é a de que toda a operação de combinação de cláusulas dos antecedentes é absorvida pela rede neural, ou seja, ela implementa diretamente funções multidimensionais de pertinência. Esta é uma abordagem mais ampla do que a de definir funções de pertinência unidimensionais e combiná-las com

conectivos de lógica nebulosa. A partição desenvolvida por este sistema tende a ser mais adequada do que a convencional, especialmente quando há dependência entre as variáveis de entrada.

4.5. Partição do espaço de entrada e representação de conhecimento

A estruturação de Redes Neurais e o uso de conceitos de lógica nebulosa contribuem para a interpretação de suas operações. Mas dar à rede uma estrutura semelhante à dos controladores nebulosos não nos garante um sistema de comportamento compreensível. Tudo depende da forma pela qual as regras são definidas e ajustadas, em particular da chamada partição do espaço de entrada.

Em tais sistemas, uma partição inicial regular do espaço é fornecida. Mesmo quando técnicas de treinamento não supervisionado são usadas (Lin & Lee 1991), o número de partições para cada variável é arbitrariamente determinado. Em seguida, aplica-se treinamento supervisionado para ajustar todos os parâmetros (inclusive os que definem o formato das funções de pertinência). A partição resultante pode apresentar tanta sobreposição entre as regras que o resultado do processamento na maioria dos casos envolve uma interação complexa entre vários conseqüentes. Nestes casos, talvez não tenhamos um sistema menos opaco do que uma rede neural convencional. Funções de erro especiais, que penalizam partições indesejadas podem ser adotadas no treinamento (Jang 1992), mas os trabalhos que encontramos na literatura apenas citam esta possibilidade, sem apresentar resultados obtidos. A princípio, espera-se neste caso maiores problemas no treinamento.

No sistema de Takagi (1993), a situação é mais grave ainda, já que as funções de pertinência, construídas diretamente no espaço multidimensional de entrada, não devem ser facilmente decompostas em funções de uma variável para facilitar a interpretação (este, aliás, não era um dos objetivos do autor).

O aproveitamento de conhecimento prévio também não se torna uma questão trivial nestes sistemas. Mais uma vez, o algoritmo de treinamento, ao procurar a minimização do erro global, pode deturpar regras corretas introduzidas, em virtude da interação com regras ainda não ajustadas. Talvez por isto, o aproveitamento de conhecimento prévio seja mais citado do que implementado nestes trabalhos.

Construir um sistema neuronebuloso eficiente sem que a partição resultante nos remeta às representações distribuídas encontradas em Redes Neurais convencionais não é uma questão simples (Babuska *et al.* 1994). Este é um dos objetivos que tínhamos em mente na definição do nosso sistema, apresentado à seguir.

Capítulo 5

Sistema Conexionista para Aquisição de Regras (SiCAR)

5.1. Objetivos

Esta seção apresenta as características desejadas para o sistema, e a partir delas as idéias fundamentais que guiaram as decisões envolvidas na definição da arquitetura e do algoritmo de treinamento.

1) *Aprendizado a partir de exemplos aliado à estruturação do conhecimento.* Queremos um sistema que suporte aprendizado supervisionado como em uma rede neural convencional, mas no qual possamos introduzir conhecimento parcial sobre o problema. Queremos também obter descrições, ainda que aproximadas, das operações da rede na elaboração de uma resposta. A partir disso, torna-se claro que a rede deve estar estruturada em regras (Capítulo 2).

2) *Uso de regras nebulosas.* Conforme estudamos anteriormente, regras nebulosas constituem um caminho promissor na tentativa de interpretação das operações de redes neurais. A arquitetura do sistema proposto deve portanto se assemelhar às dos sistemas neuronebulosos de inferência.

3) *Aquisição incremental de conhecimento.* Se nosso objetivo fundamental é a troca de conhecimento com a rede, ela deve ser capaz de adicionar regras sem prejuízo das já existentes (que podem representar conhecimento previamente introduzido). A definição a priori do número de regras e de uma partição inicial do espaço de entrada para otimização global não são compatíveis com este princípio (ver seção 4.6). Já a técnica

Jogo Conexionista da Ciência ("Connectionist Scientist Game") (McMillan *et al.* 1992) vem de encontro a nossa expectativa (ver Capítulo 2).

4) *Busca da simplificação da base de regras.* Especificar que a base de regras deve explicar todo o arquivo de treinamento não é suficiente para os nossos propósitos. Afinal, colocado desta forma, o problema admite uma solução trivial, na qual cada exemplar do arquivo de treinamento é visto como uma regra. Devemos adicionar a restrição de que a base de regras seja tão compacta quanto possível (McMillan *et al.* 1992). Para tanto, os seguintes princípios são essenciais:

4a) *Avaliação da regra elaborada.* Cada regra introduzida deve ser avaliada para que se aplique na maior região possível do espaço de entrada. Procuramos fazer com que o erro entre a saída desejada e a obtida (a informação empregada no treinamento convencional de redes neurais) seja o único guia deste processo. Esta avaliação é essencial também na introdução prévia de regras, para que o sistema infira em quais regiões do espaço pode aplicar as regras já fornecidas.

4b) *Eliminação de regras redundantes.*

4c) *Eliminação de variáveis desnecessárias à descrição da regra.*

Estes dois últimos princípios traduzem-se em uma série de procedimentos que serão detalhados na descrição do algoritmo.

5.2. Arquitetura

Como definido anteriormente, o sistema tem arquitetura similar às das redes neurais estruturadas em regras nebulosas. A Figura 5.1 a seguir traz uma nova representação geral para estes sistemas. Redes de conseqüente computam funções do vetor de entrada. Outra rede (de antecedentes) indica a concordância entre a entrada e as condições de cada regra. A saída global do sistema combina todas estas informações.

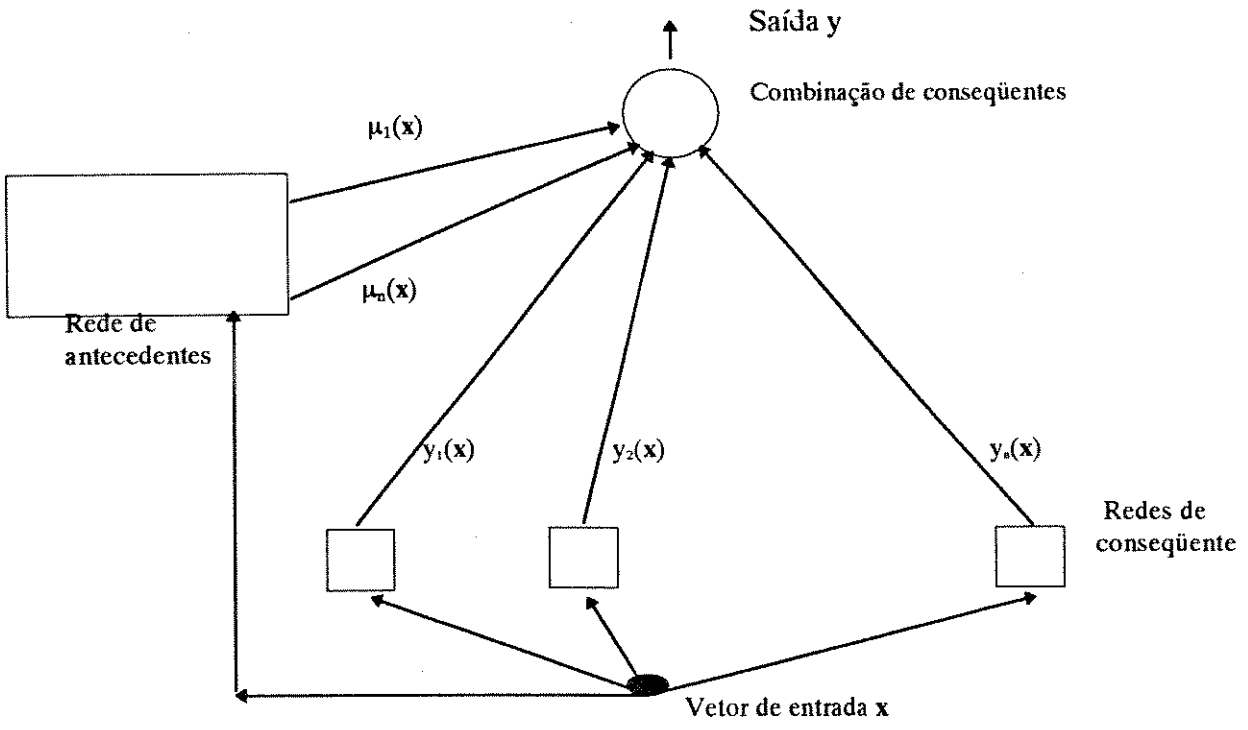


Figura 5.1: Esquema geral de sistema neuronebuloso

Precisamos agora detalhar o formato dos antecedentes e conseqüentes da base de regras para especificarmos a arquitetura das redes indicadas na figura.

Todo o algoritmo de aquisição de regras baseia-se na seleção de uma região do espaço de entrada, no desenvolvimento de uma regra induzida pelos exemplos desta região e na avaliação da regra para estender ao máximo o número de pontos cobertos. Assim, a partição é feita diretamente sobre o espaço multidimensional de entrada. As funções de pertinência (antecedentes) devem descrever convenientemente esta partição e ainda permitir a decomposição em termos de funções monodimensionais para facilitar a interpretação. Escolhemos funções gaussianas do tipo

$$\mu(\mathbf{x}) = e^{\frac{-\|\mathbf{x}_p - \mathbf{c}_p\|^2}{\sigma^2}} \qquad (5\text{-}1)$$

onde $\|.\|$ denota norma euclidiana, $\mathbf{c}$ é o ponto central da região de aplicação , σ controla o tamanho da região e $\mathbf{x_p}$ é a projeção do vetor de entrada $\mathbf{x}$ sobre as variáveis não eliminadas do antecedente.

Esta função permite a decomposição

$$\mu(\mathbf{x}) = \exp\left(\frac{-\left((x_1 - c_1)^2 + (x_2 - c_2)^2 + \ldots + (x_n - c_n)^2\right)}{\sigma^2}\right) =$$

$$= \exp\left(\frac{-(x_1 - c_1)^2}{\sigma^2}\right) . \exp\left(\frac{-(x_2 - c_2)^2}{\sigma^2}\right) . \ldots \exp\left(\frac{-(x_n - c_n)^2}{\sigma^2}\right)$$

que por sua vez pode ser interpretada como a conjunção (*and* nebuloso) de condições monodimensionais.

Tendo em mente a simplificação da base de regras, o algoritmo permite a combinação de regras que tenham conseqüentes semelhantes. Assim, a função de pertinência em sua forma mais geral é uma disjunção (OR nebuloso) de funções do tipo descrito acima:

$$\mu(\mathbf{x}) = \max_j \left(e^{\frac{-\|\mathbf{x} - \mathbf{c}_j\|^2}{\sigma_j^2}} \right) \qquad (5\text{-}2)$$

Agora estamos em condições de descrever em detalhes a rede de antecedentes. A Figura 5.2 abaixo mostra a representação do antecedente de uma regra composta por m gaussianas:

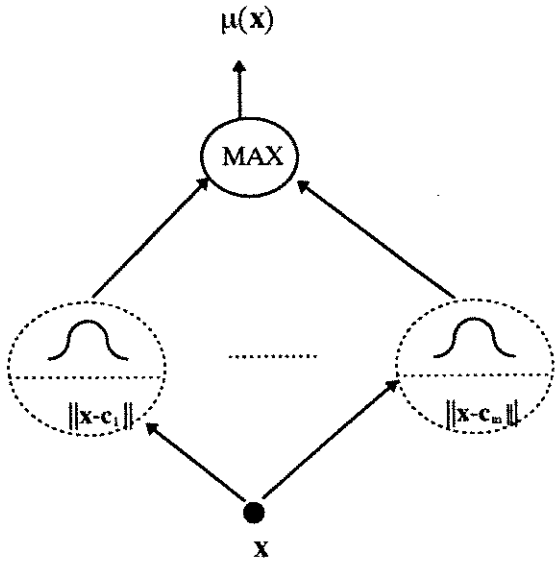


Figura 5.2: Representação de antecedente de regra

O conseqüente da regra nebulosa utilizada é uma função linear dos vetores de entrada (Takagi & Sugeno 1985, ver ítem 3.2). Para o caso de n variáveis de entrada, temos:

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{n} wi.xi \qquad (5\text{-}3)$$

motivo pelo qual a rede de conseqüente resume-se a um neurônio do tipo mais simples, com função de ativação identidade (Figura 5.3 a seguir).

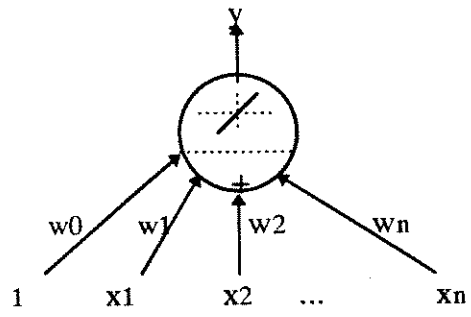


Figura 5.3: Representação de conseqüente de regra

Finalmente, a saída global do sistema é obtida, como usual, pela média das saídas de cada regra, ponderada pelas funções de pertinência das mesmas. Seja p o número de regras. Temos então:

$$y(x) = \frac{\sum_{i=1}^{p} y_i(x).\mu_i(x)}{\sum_{i=1}^{p} \mu_i(x)}$$

A estrutura da rede de saída é detalhada na figura 5.4 abaixo:

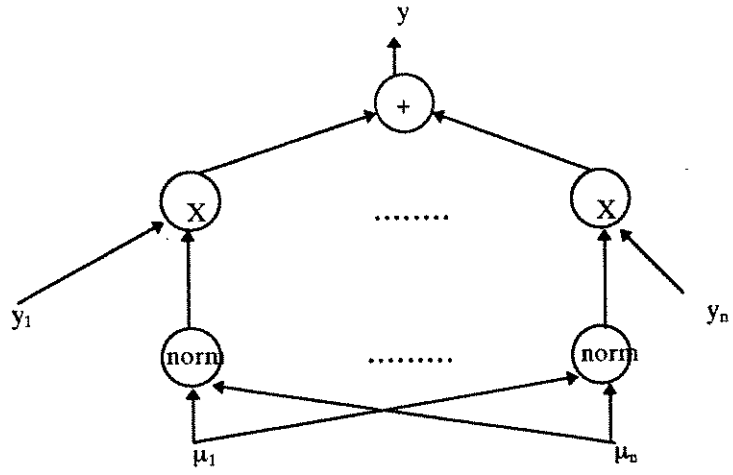


Figura 5.4: Rede de combinação de conseqüentes

Os valores μ_i representam as saídas das redes de antecedentes, e os valores y_i representam as saídas das redes de conseqüente.

Para o caso particular de classificação de padrões, a arquitetura proposta é ligeiramente modificada. Em primeiro lugar, o conseqüente das regras é um indicador de classe ($w_i=0\ \forall i \neq 0$ na equação 5-3). Além disso, neste caso não faz sentido tomar a média das saídas. As funções de pertinência das regras (agora representando pertinência às classes) já nos dão diretamente a classificação. O neurônio de saída simplesmente indica a regra (classe) para a qual a pertinência é maior.

5.3. Algoritmo

As informações necessárias ao início do processo de aquisição de regras resumem-se a um arquivo com exemplos de pares entrada-saída (padrões de treinamento) e o conjunto de parâmetros de treinamento. Para a eliminação de regras (um procedimento efetuado ao final do processo), um outro arquivo de exemplos é empregado (padrões de validação). Eventualmente, existe um conjunto de regras já definidas, decorrente de treinamentos anteriores ou de conhecimento prévio do problema.

O ponto central do algoritmo é um método para avaliação da regra e definição da função de pertinência. Para explicá-lo, vamos supor que temos o conseqüente de uma regra, induzido pelos exemplos de uma certa região do espaço de entrada. Como dito anteriormente, gostaríamos que o treinamento fosse semelhante ao de uma rede convencional, e que portanto todos os ajustes (inclusive da função de pertinência) fossem baseados nas medidas de erro entre saída desejada e obtida. Surge então uma idéia elementar: a regra deve ser aplicada nos pontos em que seu conseqüente aproxima-se da saída desejada. Procuramos traduzir esta idéia numa expressão matemática da seguinte forma:

$$\mu'_i(\mathbf{x}) = \max\left(0, \frac{\Delta - \delta_i(\mathbf{x})}{\mathrm{abs}\left(\Delta - \min_{j(x_j \in T_i)}\left(\delta_i(\mathbf{x}_j)\right)\right)}\right) \qquad (5\text{-}4)$$

onde Δ é o máximo erro quadrático admissível, δi é o erro quadrático entre a saída desejada e a da regra avaliada, e T_i representa todos os pontos usados no treinamento da regra.

$\mu'_i(\mathbf{x})$ servirá como uma restrição flexível para a região de aplicação da regra. Seu valor é 1 no ponto onde o erro de conseqüente da regra é mínimo, decresce com o aumento do erro, e vale 0 para todos os pontos no qual a regra produz erro maior ou igual ao máximo admissível. Este parâmetro Δ tem efeito direto sobre o grau de refinamento da partição obtida. Quanto menor Δ, um menor erro é tolerado e uma partição mais refinada (com maior número de regras) é obtida.

Resta o problema de definir a função de pertinência a partir dos pontos obtidos pela aplicação da expressão anterior sobre os pontos do arquivo de treinamento. Precisamos de uma função que possa ser calulada para outros pontos do espaço de entrada, caso contrário não temos capacidade de generalização.

A solução encontrada foi o ajuste de uma gaussiana diretamente sobre os pontos de $\mu'_i(\mathbf{x})$. Este procedimento foi evitado a princípio porque traz consigo perda de informação a respeito da região de aplicação. Antecedentes como os definidos no ítem anterior (equação 5-1) produzem fronteiras (nebulosas) esféricas para a partição e portanto não podem representar regiões irregulares. Para obter-se uma base de regras compacta, as funções de pertinência devem poder representar partições complexas. Por outro lado, o uso de gaussianas permite uma descrição simples do antecedente de cada regra. A solução adotada foi usar gaussianas e procurar outros meios de simplificar a base de regras (em termos de número de regras e de variáveis envolvidas). Estes procedimentos de simplificação serão posteriormente apresentados.

A figura 5.5 a seguir apresenta um gráfico ilustrativo do procedimento de avaliação de regra para o caso monodimensional. Nela, estão representados uma função desejada, o conseqüente de uma regra, o resultado da aplicação da expressão (5-4) para esta regra e a função de pertinência definitiva.

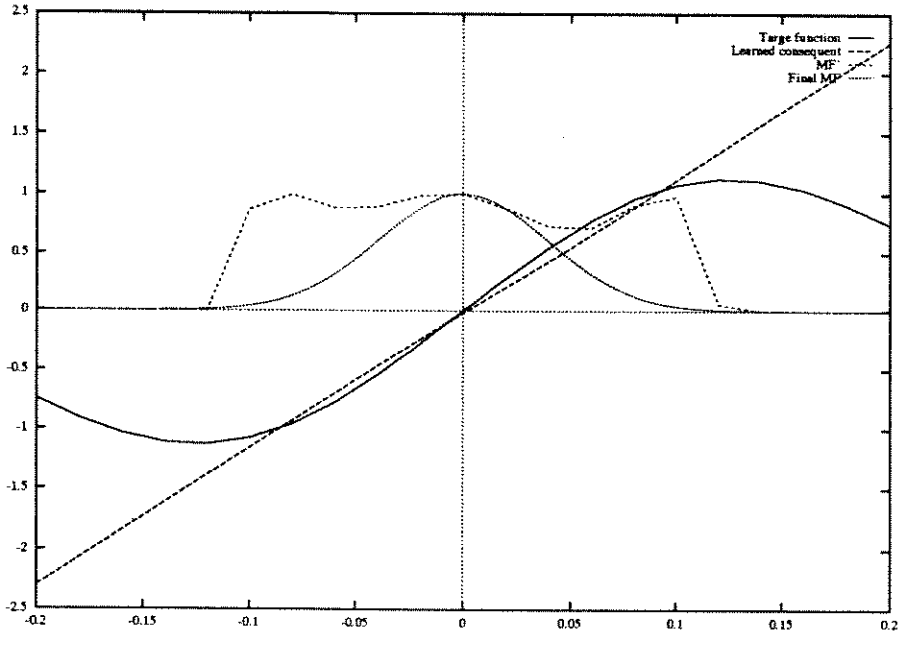


Figura 5.5 - Ilustração da avaliação de desempenho de regra. A curva contínua representa a função desejada, a reta o conseqüente de uma regra após treinamento, a curva tracejada o resultado da aplicação da expressão (5-4) e a gaussiana a função de pertinência definitiva para a regra.

Uma vez apresentado o procedimento básico de definição de nova regra, o algoritmo pode ser melhor compreendido. A Figura 5.6 mostra os pontos principais do algoritmo em um fluxograma.

O primeiro passo é a procura por uma região não coberta pelas regras existentes para a definição de uma nova regra. Para tanto, a união (OR nebuloso) das regras já existentes é calculada:

$$\rho(x) = \max_j(\mu_j(x))$$

O problema então resume-se a achar uma região do espaço na qual $\rho(\mathbf{x})$ tenha baixo valor. Chamemos r_t o raio desejado para a região e θ um limiar de cobertura (um ponto do arquivo de treinamento é considerado coberto se a função de pertinência de alguma regra tem valor acima de θ).

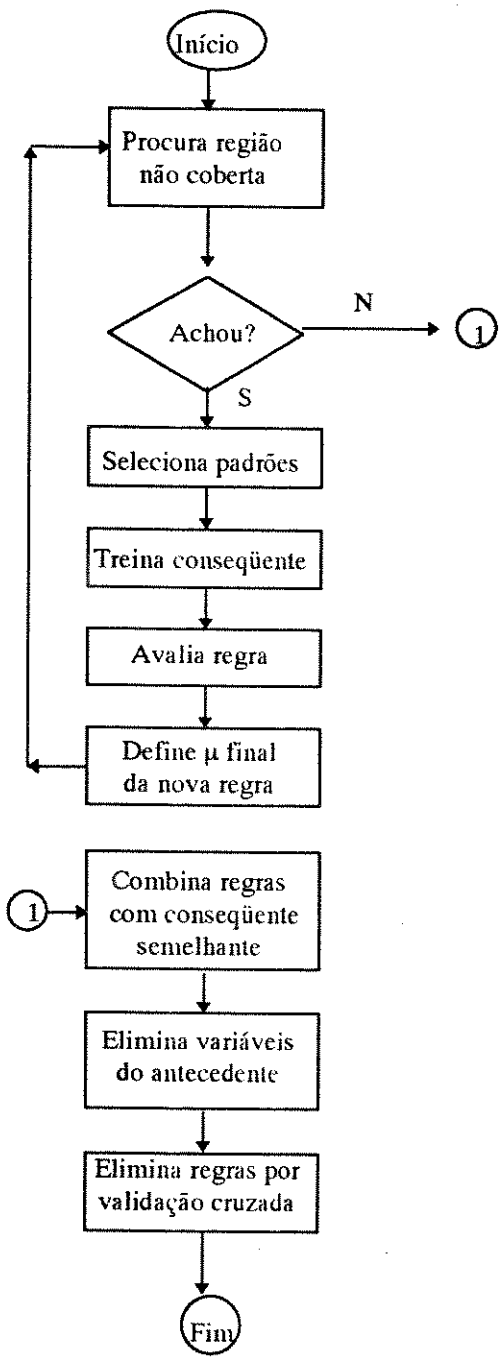


Figura 5.6: Fluxograma do algoritmo

Todos os pontos **c** que satisfazem

$$\rho(\mathbf{x}) < \theta, \forall \mathbf{x} \mid \|\mathbf{x}-\mathbf{c}\| \leq r_t$$

são candidatos. A escolha entre eles é tomada aleatoriamente. Se nenhum ponto é encontrado, o raio é diminuído de acordo com:

$$r_{t+1} = \alpha \, . \, r_t \, , \alpha < 1 \qquad (5\text{-}5)$$

e a busca é reiniciada. Evidentemente, se r_t é menor do que a menor distância entre pontos do arquivo de treinamento e ainda assim nenhum ponto é encontrado, todos os pontos estão cobertos e a partição está terminada. Encontrado o ponto, os exemplos oriundos desta região do espaço de entrada são usados para a determinação do conseqüente da regra.

Observamos que mais dois parâmetros foram incluídos para esta etapa do algoritmo: um raio inicial (podemos chamá-lo r_0), e um fator de diminuição de raio (α). A definição a priori de um raio inicial para aplicação da regra não estava nos planos iniciais mas não pôde ser evitada sem prejuízo do treinamento do conseqüente.

De fato, a idéia de dividir o espaço de entrada em regras baseia-se na expectativa de que a saída desejada em toda uma região contígua possa ser descrita por uma função simples (no nosso caso, linear). Esta função será ajustada a partir de exemplos do arquivo de treinamento, e só depois poderemos avaliar a regra para determinarmos a região final de aplicação. Assim, para o treinamento de conseqüente, temos que selecionar exemplos de uma certa região antes de determinarmos sua fronteira, razão pela qual arbitramos um valor inicial para seu raio.

Deve ficar claro que não estamos determinando a partição do espaço em regiões iguais de raio r_0. Este valor apenas guia a seleção de pontos do arquivo de treinamento para desenvolvimento do conseqüente da regra. O número, posição e tamanho das partições será encontrado na avaliação de cada regra introduzida.

Selecionados os pontos do arquivo de treinamento, o conseqüente é ajustado pelo método de gradiente que, como já apresentado (equação 1-3) resulta na seguinte equação de treinamento:

$$\Delta w_i = \varepsilon.(y_d - y).x_i$$

lembrando que o *bias* w_0 pode ser considerado um peso ligado a uma entrada $x_0 = 1$.

Para determinarmos o fim do treinamento, observamos a evolução do erro quadrático total:

$$E = \sum_{i=1}^{t}(y_d(x_i) - y(x_i))^2$$

onde t é o número de padrões no arquivo de treinamento. Costuma-se chamar *época* a um ciclo de treinamento que envolve todo o arquivo de exemplos. Se em um certo número de épocas a queda em E é menor do que um determinado valor, o treinamento é terminado e procede-se a avaliação da regra como descrita anteriormente. Outra alternativa seria fixar o número de épocas para o treinamento, mas acreditamos que o procedimento adotado dá maior flexibilidade ao sistema, que poderá interromper o treinamento quando este se tornar pouco produtivo na diminuição do erro.

Para o problema de classificação de padrões, todas as questões envolvendo raio de aplicação e treinamento deixam de existir. O conseqüente de uma nova regra é simplesmente a classe correspondente ao ponto não coberto que foi selecionado. Neste caso, o algoritmo se aproxima da técnica de classificação por vizinhos mais próximos ("k-nearest neighbours") (Cover & Hart 1967), e do aprendizado baseado em instâncias ("instance-based learning") no sentido de que instâncias são selecionadas para representar casos típicos do problema, cada instância selecionada exercendo certa influência sobre as regiões vizinhas do espaço. No nosso caso, a região de influência é determinada no processo de avaliação apresentado anteriormente.

Ao final da avaliação da regra recém-introduzida, podemos descobrir uma relação de dominação entre ela e uma das regras já existentes, isto é, uma regra lida com todos os padrões cobertos pela outra e mais alguns. Neste caso, a regra dominada é eliminada.

Terminada a partição, alguns passos são tomados no sentido de simplificar a base de regras final. Primeiramente, queremos que regras com conseqüentes semelhantes sejam combinadas em uma única regra com antecedente do tipo descrito na equação 5-2 (várias gaussianas e um único conseqüente).

Definimos a idéia de conseqüentes semelhantes como se segue: Sejam $\mu_1(\mathbf{x})$ e $\mu_2(\mathbf{x})$ funções de pertinência de duas regras e seja $y_1(\mathbf{x})$ o conseqüente da regra 1. O conseqëunte da regra 2 pode ser englobado pelo da regra 1 se

$$\left(y_d(\mathbf{x}) - y_1(\mathbf{x})\right)^2 \leq \Delta, \forall \mathbf{x} \mid \mu_2(\mathbf{x}) > \theta$$

isto é, se em todos os pontos cobertos por $\mu_2(\mathbf{x})$ o conseqüente da regra 1 produz um erro aceitável. Neste caso, a gaussiana $\mu_2(\mathbf{x})$ é incorporada ao antecedente da regra 1 (equação 5-2).

O processo de combinação começa tentando eliminar o conseqüente da regra que cobre menos pontos e segue por ordem crescente de número de pontos cobertos.

Para o caso de classificação de padrões, este processo terminará por fazer com que o número de regras seja igual ao de classes (cada regra eventualmente representada pela união de várias gaussianas).

Um outro passo para a simplificação da base de regras é a eliminação de variáveis. Se uma variável é retirada do cálculo da distância em (5-1), ou seja, se a distância euclidiana é substituída por uma distância de Mahalanobis que desconsidera um dos eixos, a região de aplicação da regra evidentemente aumenta. Se a regra é corretamente aplicada (erro menor que Δ) nesta nova região, a variável pode ser eliminada. Neste caso, a função de pertinência como apresentada em (5-1) terá alguns componentes retirados.

A ordem da tentativa de retirada de variáveis é importante, mas a princípio não temos nenhuma informação como guia deste processo e não podemos portanto garantir que a base de regras resultante está otimizada quanto ao número de variáveis envolvidas, ainda que os resultados obtidos mostrem redução significativa de dimensionalidade.

O último passo de simplificação consiste na eliminação de gaussianas por validação cruzada ("cross-validation") (Turney 1990, Tresp *et al.* 1993). A razão para este procedimento é a seguinte: na presença de ruído acentuado no arquivo de treinamento, regras serão também desenvolvidas para "explicar" os padrões ruidosos. A retirada destas regras possivelmente diminui ou deixa inalterado o erro sobre o arquivo de validação (um arquivo com exemplos não apresentados no treinamento). Assim, eliminamos as gaussianas cuja retirada não aumenta o erro sobre um arquivo de validação e também as regras cujas gaussiansas foram todas eliminadas.

Outros motivos para que regras possam ser eliminadas são: valor muito alto de θ (o que cria uma condição muito forte para o fim da partição, com número excessivo de regras) e efeito da eliminação de variáveis (que faz com que uma regra "invada" o espaço de outras). Assim, a eliminação também pode ser conseguida sobre o próprio arquivo de treinamento, mas o uso de validação cruzada deve produzir melhores resultados de generalização.

5.4. Introdução de conhecimento prévio

Antes do início do processo de aquisição de regras, uma avaliação de todas as regras já existentes é efetuada, na forma descrita anteriormente. Estas regras podem ser resultado de um treinamento anterior interrompido ou da introdução de conhecimento parcial sobre o problema. Chamaremos estas últimas regras de *pré-definidas*, enquanto as regras desenvolvidas pelo sistema serão chamadas induzidas.

A princípio, poderíamos tratar regras pré-definidas de maneira idêntica à que usamos para regras induzidas: uma função provisória de pertinência e o vetor de conseqüentes seriam introduzidos *a priori*, e então o processo de avaliação de regra e definição da função de pertinência definitiva seria iniciado. Mas este processo leva, como vimos, a funções de pertinência com simetria esférica. Perderíamos a oportunidade de representar partições mais complexas que fossem de conhecimento do usuário. Para que isto não ocorra, determinamos que *as regras pré-definidas não sejam alteradas pelo processo de avaliação.* A avaliação ocorre apenas para que o sistema identifique quais regiões não estão cobertas pelas regras introduzidas, e desenvolva novas regras para estas regiões. Mas as funções de pertinência das regras pré-definidas não se modificam.

Estamos particularmente interessados no caso em que as regras pré-definidas representam conhecimento parcial sobre o problema, uma informação segura na qual o sistema pode confiar, na ausência de informações advindas dos exemplos. Assim, não podemos permitir que o arquivo de treinamento induza (pela presença de ruído, por exemplo), regras que comprometam o desempenho das regras pré-definidas.

A solução deste novo problema foi conseguida definindo-se que *as regras pré-definidas têm preferência sobre as induzidas.* Explicitamente,

$$\mu(\mathbf{x}) < 1 - \max_{k} \mu_{p_k}(\mathbf{x})$$

onde μ_p representa uma regra pré-definida e μ uma regra induzida. Em termos de arquitetura de rede, isto significa que a ativação de neurônios correspondentes a regras induzidas inibe os neurônios das regras pré-definidas. Isto é conseguido com o arranjo mostrado na Figura 5.7 a seguir

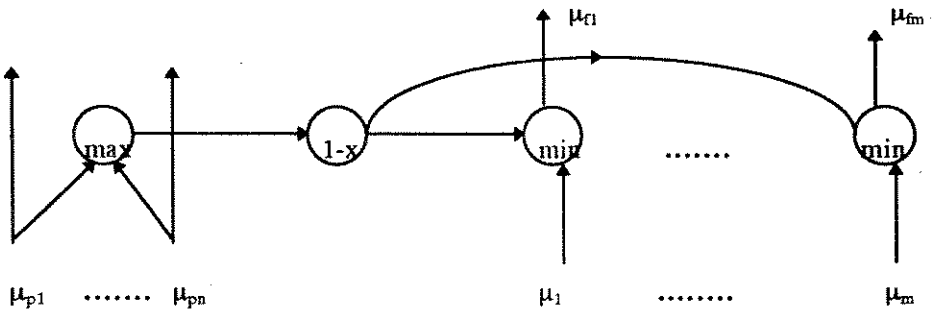


Figura 5.7: Esquema de inibição de regras induzidas

Pode parecer que ao sistema não está sendo dada nenhuma possibilidade de criticar as regras introduzidas, mas é preciso lembrar que a avaliação da regra pré-definida continua sendo feita normalmente para efeito de verificação de cobertura. Regras novas serão induzidas para a região onde a regra pré-definida não for considerada satisfatória. A inspeção da base de regras final pode revelar desta forma regiões onde regras pré-definidas foram "reprovadas" pelas informações disponíveis. Os experimentos descritos no capítulo seguinte deixaram claro que o sistema não fica sensível em demasia à qualidade das regras pré-definidas.

É preciso dizer também que esta relação de inibição talvez precise ser modificada ou mesmo invertida em outras situações. As regras pré-definidas podem, por exemplo, representar sugestões com baixo grau de certeza. Neste caso, elas é que devem ser inibidas pelas regras induzidas pelos exemplos.

Pelos mesmos motivos expostos acima, as regras pré-definidas não podem ter seus conseqüentes absorvidos por alguma outra, nem participam da tentativa de simplificação da base de regras.

A Figura 5.8 a seguir apresenta um esquema detalhado da arquitetura final de um sistema hipotético com 4 regras. Duas regras são pré-definidas, e duas induzidas (uma composta por duas gaussianas e outra por três).

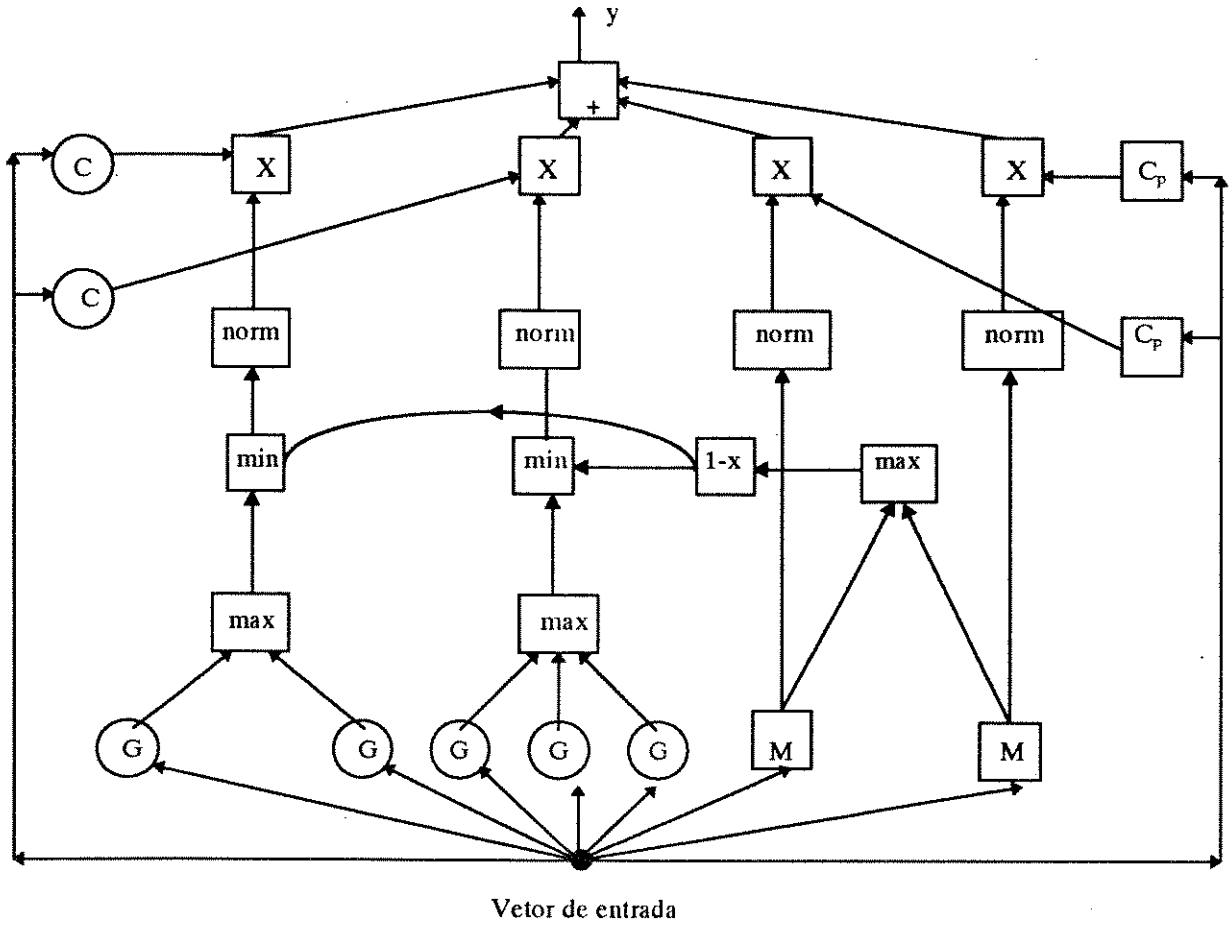


Figura 5.8: Exemplo completo da forma final do sistema

Na figura acima:

- Círculos representam neurônios com parâmetros variáveis.
- Quadrados representam operações fixas na rede.
- *G* representa neurônios com função de ativação gaussiana (lembrar que nem todos os componentes do vetor de entrada são necessáriamente utilizados).
- *M* representa as funções de pertinência previamente introduzidas.
- *C* representa os neruônios lineares de conseqüente.
- C_p representa o conseqüente de regras pré-definidas.

A interconexão entre os neurônios de saída da rede que computa funções de pertinência e a camada de normalização é total, mas nem todas as conexões são representadas acima por questão de clareza.

Definido o sistema, passamos, no capítulo seguinte, à descrição da implementação, experimentos e resultados obtidos.

Capítulo 6

Experimentos e resultados

6.1. Implementação e metodologia

Descreveremos a seguir os experimentos realizados para validação das idéias desenvolvidas no capítulo anterior, identificação de pontos fortes e limitações do sistema, e definição de possíveis extensões e melhoramentos.

Um grupo de programas em linguagem C foi desenvolvido para implementar o algoritmo descrito no ítem anterior. As simulações foram efetuadas em estações de trabalho SUN Sparc 1 e Sparc 1+ do Laboratório de Engenharia de Computação e Automação Industrial (LCA). Aproveitamos algumas idéias do software NeuralShell da Ohio State University (Little 1992), especialmente para apresentação de relatórios.

A metodologia empregada procura evitar que os resultados obtidos sejam influenciados em demasia por alguma combinação eventual de parâmetros, escolhas aleatórias durante o treinamento e partição dos dados disponíveis entre treinamento e teste. Para isto, em cada caso, várias simulações com diferentes partições de dados foram feitas. Para os casos de aproximação de funções, a cada simulação um arquivo de treinamento diferente era construído por amostragem uniforme do espaço de entrada.

O arquivo de teste é usado para estimar a performance do sistema em casos não cobertos pelos exemplos do treinamento, e portanto não pode ser usado para nenhum ajuste de parâmetros (Prechelt 1994). O procedimento de eliminação de regras, por exemplo, foi na maioria das vezes efetuado sobre o próprio arquivo de treinamento. Para os testes de tolerância a ruído, o arquivo para tal procedimento (que será chamado arquivo de validação seguindo Prechelt 1994) incluía pontos adicionais (não vistos no

treinamento) com a mesma amplitude de ruído. Apenas o arquivo de teste estava livre de ruído neste caso.

Ao lado dos resultados obtidos indicaremos o valor de todos os parâmetros uitlizados e a forma de construção dos diferentes arquivos, para permitir a reprodução dos resultados aqui obtidos.

6.2. Objetivos dos experimentos

A cada grupo de simulações, tínhamos em mente o teste de uma característica específica do sistema:

- *Sensibilidade à variação de parâmetros.* No desenvolvimento do algoritmo de aquisição de regras, foram introduzidos vários novos parâmetros. Era necessário verificar a sensibilidade do sistema à variação dos parâmetros introduzidos. Uma sensibilidade dramática a algum parâmetro ou uma dificuldade maior de ajuste seriam sérios obstáculos ao uso do sistema.

- *Capacidade de generalização e imunidade a ruído.* Foi nosso objetivo verificar como SiCAR comparaa-se a um Perceptron multicamada com relação a estas duas importantes características.

- *Problemas típicos de aprendizado em máquina.* Um grupo de "benchmarks" obtidos via Internet da base de dados de aprendizado em máquina da Universidade da Califórnia em Irvine, EUA (Murphy & Aha 1994) foi usado para comparação do sistema com outras técnicas de aprendizado. Utilizamos também um problema proposto por Sugeno & Kang (1988), de aproximação de uma função não linear de três variáveis.

- *Interpretação do conhecimeneto adquirido.* Analisamos a base de regras resultante das simulações anteriores para verificar se o principal objetivo do trabalho (obter descrições aproximadas do conhecimento adquirido) foi alcançado.

- *Desempenho com introdução prévia de regras.* Era outro objetivo central do trabalho poder introduzir no sistema conhecimento parcial sobre o problema estudado. Algumas das simulações anteriores foram repetidas, agora com introdução de regras que

cobriam parcialmente a solução dos problemas, comparando-se o desempenho do sistema com o obtido anteriormente.

6.3. Sensibilidade à variação de parâmetros

Para este grupo de simulações, escolhemos o problema de aproximação de uma função monodimensional. A função objetivo é dada por :

$$y = 0.3 \text{ sen}(\pi x) + 0.6 \text{ sen } (3\pi x) + 0.5 \text{ sen } (5\pi x)$$

Os arquivos de treinamento foram preparados amostrando-se 100 pontos $(x,f(x))$ com $x \in [0,1]$ segundo uma distribuição uniforme de probabilidades. Procedimento idêntico foi adotado para os arquivos de validação.

Partindo-se de um conjunto inicial de parâmetros, especificado na Tabela 6.1, 5 grupos de simulações foram realizados, cada um com variações em um parâmetro. As tabelas 6.2 a 6.6 mostram os resultados. Os valores indicam média e desvio-padrão obtidos em certo número de simulações (entre 3 e 6) com diferentes condições iniciais e arquivos de treinamento e teste.

Parâmetro	Significado	Valor Inicial
ε	Constante de aprendizado (Eq. 1-3)	0.15
Δ	Erro qudrático máximo admissível (Eq. 5-4)	0.025
R_0	Raio inicial de atribuição de regra	0.08
α	Constante para diminuição de raio (Eq. 5-5)	0.7
θ	Limiar de cobertura	0.1
I_{pt}	Iterações entre testes de erro	1000
D_{em}	Queda mínima de erro para continuação do treinamento	1e-6

Tabela 6.1: Conjunto inicial de parâmetros para teste de sensibilidade

Δ	Num. Regras	Num. Gaus	Erro RMS	Tempo (s)
0.01	16±1	27	7±3	150±9
0.025	13	23	7±1	150±10
0.05	11±2	19±3	8±2	110±4
0.1	11±2	19±1	11±1	90±20

Tabela 6.2: Resultados da variação de Δ

θ	Num. Regras	Num. Gaus	Erro RMS (x 10^{-2})	Tempo (s)
0.1	13	23	7±1	150±10
0.5	14	25±2	9	160±50
0.7	14	28	10±2	190±50

Tabela 6.3: Resultados da variação de θ

R_0	Num. Regras	Num. Gaus	Erro RMS (x 10^{-2})	Tempo (s)
0.03	15	24	7±1	210±10
0.08	13	23	7±1	150±10
0.2	17±1	28±2	8±1	160±10

Tabela 6.4: Resultados da variação de R_0

I_{pt}	Num. Regras	Num. Gaus	Erro RMS (x 10^{-2})	Tempo (s)
100	15±1	24±1	9±3	240±40
500	15±1	22±4	7±1	155±10
1000	13	23	7±1	150±10
2000	15±2	22±3	8±1	115±8

Tabela 6.5: Resultados da variação de I_{pt}

α	Num. Regras	Num. Gaus	Erro RMS (x 10^{-2})	Tempo (s)
0.3	14±1	25±1	8±1	180±20
0.5	14±1	25±4	7±1	190±30
0.7	13	23	7±1	150±10

Tabela 6.6: Resultados da variação de α

Explicamos abaixo o significado das diferentes colunas das tabelas anteriores:

- **Num. Regras** : É o número de conseqüentes diferentes na base de regras.
- **Num. Gaus**: É o total de gaussianas que descrevem as regras induzidas.
- **Erro RMS** : É o resultado da expressão

$$E_{RMS} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - yd_i)^2}{n}} \qquad (6\text{-}1)$$

onde y_i é o valor obtido pelo sistema para a entrada x_i do arquivo de validação, yd_i o valor desejado para a mesma entrada, e *n* o número de pontos (no nosso caso 100).

- **Tempo** : Tempo real de execução, incluindo treinamento, testes, cálculos de erro e geração dos arquivos de saída (base de regras e relatório).

O aspecto mais significativo das variações estudadas foi o efeito dos parâmetros Δ e θ no refinamento da partição. Conforme esperado, uma maior tolerância no erro admissível (Δ) resulta em uma partição mais grosseira, com menor número de regras. Resulta ainda em maior erro médio na saída, mas a variação obtida no erro foi bem menor do que a excursão do parâmetro Δ. O parâmetro θ tem efeito semelhante. Um valor mais alto de θ significa um citério mais exigente de cobertura, e conseqüentemente um número maior de regras. Todas as simulações foram bastante rápidas, variando entre 1 e 3 minutos.

Os demais parâmetros estudados (R_0, I_{pt} e α) só tiveram efeito significativo sobre a duração do treinamento. Por exemplo, especificar um maior número de iterações entre os testes para interrupção do treinamnto (parâmetro I_{pt}) resulta evidentemente em um maior número total de iterações. Mas os resultados finais (com relação à partição obtida e ao erro médio) não foram significativamente alterados.

Não foram exploradas as inúmeras possibilidades de variação conjunta de parâmetros. Mas os resultados obtidos com variações isoladas e o fato de não termos encontrado dificuldades para o ajuste de parâmetros nas várias outras simulações relatadas a seguir indicam que não há sensibilidade acentuada à maioria dos parâmetros introduzidos, excetuando-se o efeito (previsível) de Δ e θ no refinamento da partição.

Temos na Figura 6.1 gráficos que mostram a saída obtida sobreposta à desejada, a evolução do erro médio de saída durante o treinamento e a partição do espaço (valores normalizados de todas as funções de pertinência). Estes resultados foram obtidos com o conjunto inicial de parâmetros. Os pontos obtidos estão ligados por segmentos de reta para facilitar a visualização. Observa-se um caráter não monotônico na curva de erro durante o treinamento. A introdução de uma nova regra (com conseqüente aleatório) produz por vezes saltos no valor de erro. Efeito semelhante é relatado em McMillan *et al.* 1992, onde também é usada a introdução periódica de novas regras.

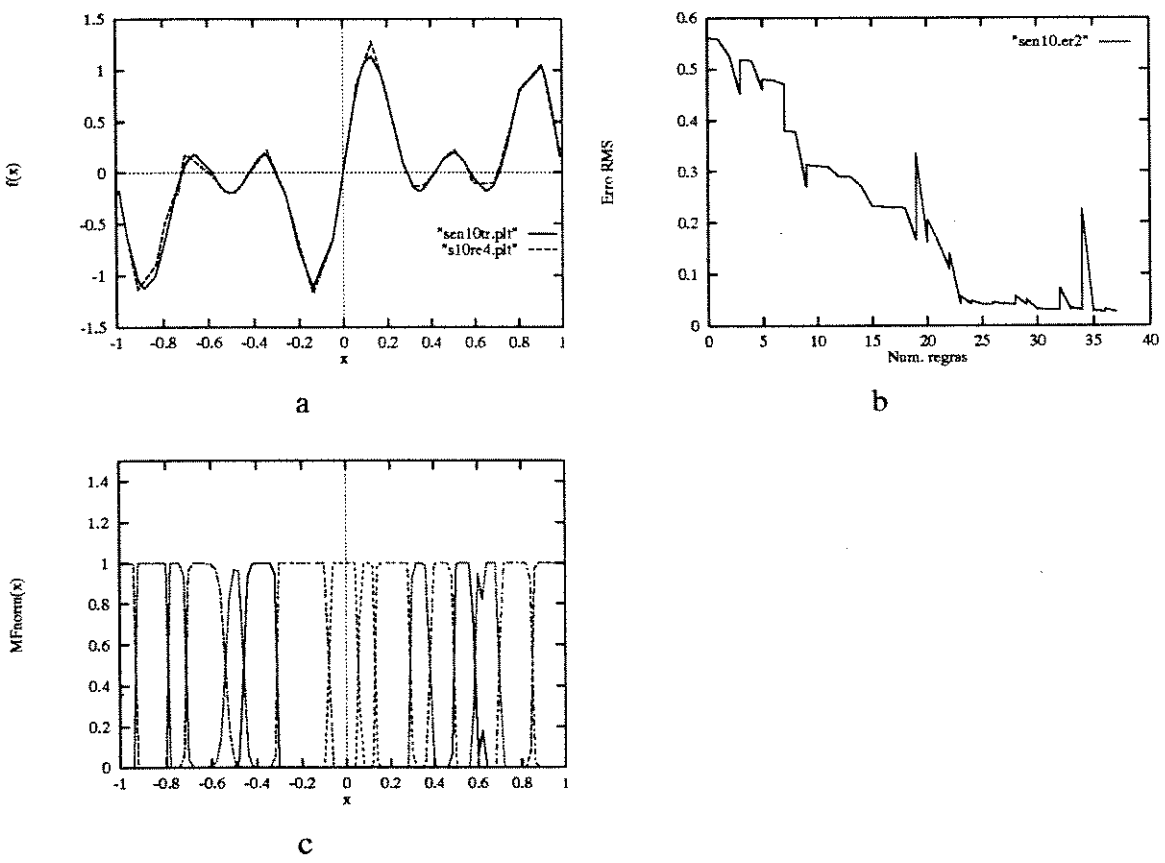
A base de regras é bastante clara, no sentido de indicar regras predominantes em cada região do espaço. Os conseqüentes destas regras representam efetivamente aproximações locais da função objetivo. É interessante observar que os trechos onde há maior sobreposição de regras (é menos nítida a dominação de uma regra sobre as demais) corresponde aos trechos de maior curvatura da função-objetivo.

6.4. Capacidade de generalização e imunidade a ruído

Para este grupo de simulações, outra função-objetivo foi escolhida:

$$y = \begin{cases} -0.2679x + 0.732, & -1 \le x < -0.5 \\ \sqrt{1 - x^2}, & -0.5 \le x \le 0 \\ e^{-4.6x}, & 0 < x \le 1 \end{cases}$$

Esta função foi extensamente utilizada durante o desenvolvimento inicial do sistema. Seu trecho linear permite avaliar se as regras induzidas são aproveitadas ao máximo (uma única regra deve lidar com este trecho).



a b c

Figura 6.1: Saída da rede sobreposta à desejada (a), evolução do erro (b) e partição final do espaço de entrada (c) para função senoidal.

Mais uma vez, o arquivo de teste foi obtido amostrando-se 100 pontos da função acima com $x \in [0,1]$. Diferentes arquivos de treinamento foram também preparados, variando-se, para estudo de generalização, o número de pontos amostrados. A Tabela 6.7 e 6.8 a seguir apresentam os parâmetros utilizados nestas simulações e os resultados obtidos.

Parâmetro	Valor
ε	0.05
Δ	0.005
R_0	0.08
D_{em}	$1e^{-6}$
θ	0.1
α	1.4
I_{pt}	1000

Tabela 6.7 - Parâmetros utilizados para aproximação de função monodimensional

Taxa Comp.	Num. Regras	Num. Gauss	Erro Tr	Erro Teste	Correções	Tempo (s)
1.0	6±1	14±3	0.018±4	0.021±3	1.7±0.2	70±30
	6±1	10±2	0.017±2	0.017±2		
1.25	6±1	15±5	0.017±1	0.021±6	1.3±0.3	50±20
	6±1	10±3	0.015±2	0.019±6		
1.67	6±1	15	0.017±3	0.021±6	1.7±0.2	46
	6±1	11±2	0.016±3	0.022±3		
2.5	6±1	12±1	0.016±9	0.034±4	1.5±0.3	60±30
	6±1	8±1	0.015±8	0.035±4		
5.0	5±1	8±2	0.024±2	0.070±1	0.6±0.3	30±20
	5±1	6±2	0.021±5	0.070±2		

Tabela 6.8 - Resultados de generalização (função monodimensional)

Significado das colunas (valem também as explicações para as tabelas anteriores):

- **Taxa de compressão**: Relação entre o número de pontos no arquivo de teste e no arquivo de treinamento.

- **Erro Treinamento:** Resultado da aplicação de (6-1) sobre o arquivo de treinamento.
- **Erro Teste** : Resultado da aplicação de (6-1) sobre o arquivo de teste.
- **Correções**: Número de vezes que algum parâmetro de conseqüente é modificado durante o treinamento.

Dois dos arquivos anteriores (com taxas de compressão 1.0 e 5.0) foram usados para uma comparação com um Perceptron multicamadas. A rede utilizada continha 1 camada escondida com 10 neurônios. Todos os neurônios tinham funções de ativação sigmoidais. Transformações lineares foram usadas na entrada e na saída, para ajuste das faixas de valores. Na Tabela 6.9 temos os resultados da comparação (os resultados de SiCAR são valores médios). Os valores de erro referem-se sempre ao arquivo de teste.

Taxa Comp.	Erro PMC	Erro Sicar	Correções PMC x 10^6	Correões Sicar x 10^6	Tempo PMC	Tempo Sicar
1.0	0.017	0.021	396.8	1.7	162 min	70 s
5.0	0.018	0.070	297.6	0.6	118 min	30 s

Tabela 6.9 - Comparação entre Sicar e Perceptron multicamada

O desempenho do sistema com grande quantidade de dados foi equivalente ao de uma rede neural convenional, conseguindo-se enorme redução de custo computacional. Além disso, o uso do PMC exigiu várias tentativas para determinação da arquitetura e de parâmetros de treinamento, e vários testes foram realizados para definir quando interromper o treinamento. No caso de SiCAR, apenas o parâmetro Δ exigiu algumas tentativas para ajuste, e o processo de treinamento, uma vez iniciado, não exigiu mais nenhuma intervenção até seu término.

No caso de um arquivo de treinamento pobre em informações, o desempenho do sistema declinou dramaticamente, enquanto o do perceptron manteve-se no mesmo nível. Neste caso, a diferença de custo computacional foi ainda maior. Isto parece indicar que, no caso de treinamento com poucos dados, o critério de parada usado em SiCAR encerrou prematuramente o aprendizado. Outra possível razão para a melhor capacidade de generalização do PMC está relacionada com a suavidade da função de saída deste último sistema. Como vimos (seção 1.4), a restrição de suavidade é um importante fator de regularização, e auxilia na solução do problema (mal-posto) de minimização do erro empírico. A arquitetura do Perceptron multicamada (cascata de processadores com função de transferência contínua) favorece a suavidade da função resultante. O mesmo não se pode dizer de SiCAR, onde nenhuma restrição global de suavidade é imposta para o conjunto de regras lineares.

Foram preparados também arquivos com taxa de compressão 1.67 corrompidos por ruído branco com diferentes amplitudes, para estudo da imunidade ao ruído. Em cada caso, o arquivo de validação foi composto adicionando-se ao arquivo de treinamento um outro com mesma quantidade de pontos e amplitude de ruído. Os parâmetros utilizados foram os mesmos da Tabela 6.7, exceto quanto a Δ, que foi aumentado acompanhando a amplitude de ruído conforme indicado na Tabela 6.10 a seguir. A Tabela 6.11 mostra os resultados para este grupo de simulações. A coluna **Erro Val.** refere-se a aplicação da expressão (6-1) sobre o arquivo de validação.

Amplitude do Ruído	Valor de Δ
0.05	0.005
0.2	0.04
0.4	0.16

Tabela 6.10: Valores de Δ para simulações com ruído

Amplitude Ruído	Num. Regras	Num. Gauss	Erro Val	Erro Teste	Correções	Tempo (s)
0.05	6±2	20±1	0.037±8	0.033±5	7±2	100±100
	6±2	11±1	0.037±1	0.04±1		
0.20	7±1	18±5	0.14±2	0.11±6	11±10	200±200
	5±1	11±5	0.13±1	0.11±6		
0.40	5±1	21±3	0.24±1	0.19±2	20±10	1000±300
	5±1	14±2	0.229±4	0.17±2		

Tabela 6.11: Resultados de testes de imunidade a ruído

O sistema foi capaz de desprezar, em certo grau, o ruído introduzido (para pontos corrompidos com ruído de amplitude 0.4, o erro médio de saída foi 0.2). Este grupo de simulações mostra também a eficiência do algoritmo de retirada de regras. Em alguns casos, metade das gaussianas foram eliminadas, com pequena queda de desempenho, ou , como nos casos de ruído mais elevado, com melhoria de desempenho.

Nota-se também que o critério automático de interrupção do treinamento foi capaz de promover ciclos de aprendizado maiores quando necessário (o casos de maior ruído, e conseqüentemente maior conflito de informações, foram resolvidos com treinamentos mais longos, sem que houvesse intervenção externa neste sentido.

As Figuras 6.2a a 6.2d são gráficos exibindo a evolução do sistema em diversos pontos do treinamento (função de saída sobreposta à objetivo) para o caso em que nenhum ruído foi acrescentado aos dados de entrada. É interessante acompanhar como o sistema vai desenvolvendo regras que pouco a pouco vão "explicando" o arquivo de treinamento. A Figura 6.2e mostra o resultado quando o parâmetro Δ é diminuído de 0.04 para 0.005 e o treinamento reiniciado, obrigando o sistema a desenvolver novas regras para esta condição mais exigente. Isto pode ser acompanhado também no gráfico de evolução de erro (Figura 6.2f). Com 10 regras, foi feita a diminuição de Δ, e verifica-se que as novas regras inicialmente perturbam o sistema, mas acabam contribuindo para a queda do erro.

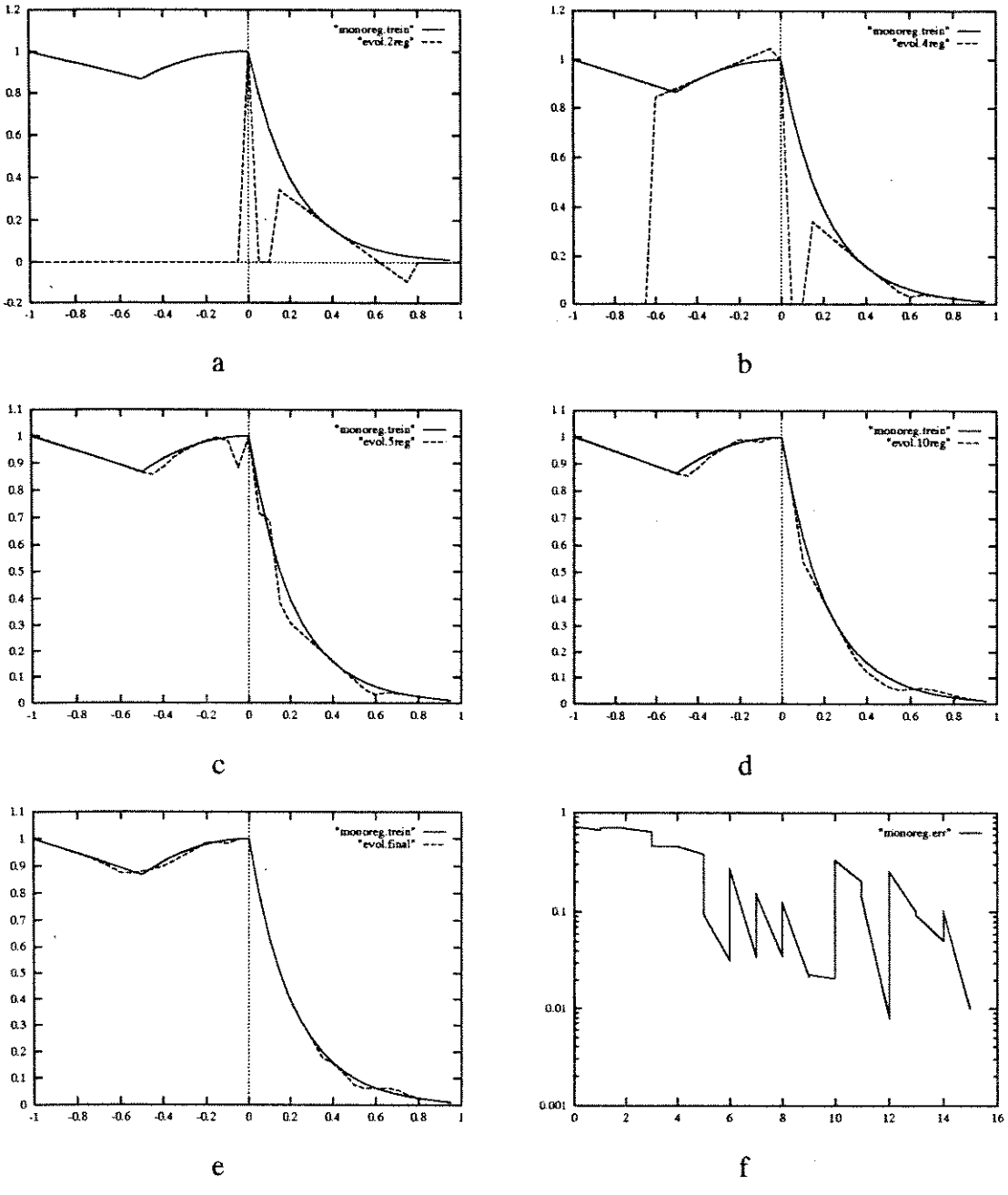
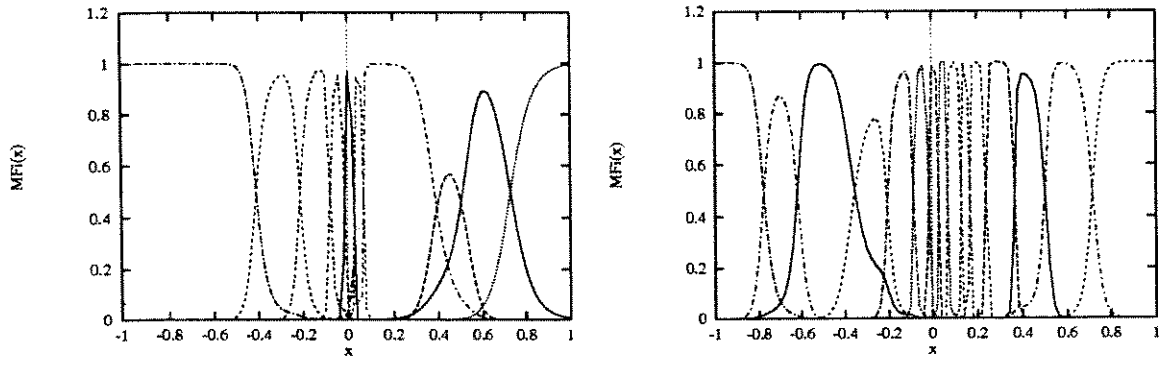


Figura 6.2: Evolução da aproximação da função objetivo ao longo do treinamento (a-e); Evolução do erro quadrático médio em função do número de regras (f).



a b

Figura 6.3: Partição do espaço de entrada antes da redução no valor de Δ (a) e resultado final após redução de Δ (b).

Na Figura 6.3 temos a comparação entre as partições obtidas para os dois valores de delta. Pode-se notar como um valor mais baixo de Δ efetivamente produz uma partição mais refinada do espaço de entrada.

6.5. Aproximação de função de três variáveis

Em termos de aproximação de funções multivariáveis, escolhemos como objetivo à função

$w = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2$ proposta por Sugeno & Kang (1988) para teste de sistemas nebulosos. O procedimento adotado foi semelhante ao empregado com a função monodimensional. A Tabela 6.12 mostra os parâmetros utilizados e na Tabela 6.13 temos os resultado obtidos. Os testes foram feitos sempre sobre um mesmo arquivo com 216 pontos amostrados no espaço [1,6] x [1,6] x [1,6].

Parâmetro	Valor
ε	0.02
Δ	1.0
R_0	2.0
D_{em}	$1e^{-6}$
θ	0.1
α	1.01
I_{pt}	1000

Tabela 6.12 - Parâmetros utilizados para aproximação de função de três variáveis

Arquivo de trein.	Num. Regras	Num. Gauss	APE Trein.	APE Teste	Correções	Tempo (min)
216 pts..	8±1	120±10	2.9±0.5	3.7±0.6	40±10	50±20
	8±1	55±9	2.6±0.4	3.7±0.6		
100 pts..	8±3	71±9	2.7±0.2	4.4±0.6	30±30	10±10
	8±2	34±2	2.4±0.4	4±1		
50 pts.	8±2	39±4	2.5±0.2	7±2	14±9	4±2
	8±2	20±3	2.1±0.1	7±3		

Tabela 6.13 - Resultados de aproximação de função de três variáveis

APE significa erro percentual médio. Aqui os tempos de simulação foram bem maiores, especialmente para arquivos de treinamento grandes. A grande qunatidade de gaussianas desenvolvidas para cobrir o espaço resultou em um esforço computacional bem maior do que no caso monodimensional. Mesmo assim, para os piores casos tivemos tempos reais de execução em torno de uma hora. Foi conseguida uma enorma redução no número final de guassianas, com nenhuma queda notável de erro, o que indica que a condição de cobertura era mais exigente do que o necessário para este caso. Para treinamento com poucos pontos, observa-se algum "overfiting", com um bom ajuste sobre o arquivo de treinamento e pior desempenho sobre o arquivo de teste.

Alguns resultados divulgados na literatura para o mesmo problema são mostrados abaixo (erro percentual médio sobre arquivo de teste), juntamente com o erro verificado para um modelo linear ajustado por mínimos quadrados sobre o mesmo arquivo de teste:

- GMDH (Kang - Polinômios de grau não-inteiro) 5.7%
- Sistema Fuzzy 1 (Sugeno & Kang 1988) 2.1%
- Sistema Fuzzy 2 (Sugeno & Kang 1988) 3.4%
- ANFIS (Jang 1992) 1.07%
- Modelo linear (ajustado por quadrados mínimos) 11.5%

Com exceção do último valor, os resultados referem-se a diferentes arquivos, e diferentes metodologias, dificultando uma comparação. Deve-se observar apenas que todos eles estão em uma mesma faixa de valores, com exceção do modelo linear, que evidentemente não representa bem o relacionamento entre os dados, e ANFIS, que apresenta resultados bem superiores. Pode-se dizer que o sistema proposto neste trabalho, construído sobre idéias intuitivas de aquisição de conhecimento, apresenta resultados comparáveis aos dos sistemas cuja construção visa preferencialmente a redução do erro.

6.6. O Problema *Iris*

Os próximos problemas estudados foram todos obtidos da base de dados de aprendizado em máquina da Universidade de Califórnia em Irvine, EUA (Murphy & Aha 1994) pela Internet. Com isso, podemos comparar resultados com aqueles divulgados na literatura, ainda que raramente sejam relatados os detalhes das metodologias empregadas, o que dificulta a comparação (Prechelt 1994b).

O problema *Iris* é um clássico da área de reconhecimento de padrões, empregado em testes de inúmeros algoritmos desde a década de 40. O objetivo é a classificação de exemplares da espécie *Iris* em uma de três subespécies, baseada em quatro medidas: largura e comprimento de sépala e pétala. São dados 150 pontos. O problema não é difícil, e taxas de erro abaixo de 10% são normalmente conseguidas.

A Tabela 6.14 traz os resultados obtidos. Para o caso de reconhecimento de padrões, os únicos parâmetros de interesse são Δ e θ (ver seção 5.3). Nestes casos, os valores de Δ sempre foram fixados em

$$\Delta = \left(\frac{1}{2}d\right)^2$$

onde d é a menor diferença entre os números associados às classes. Para o caso *Iris*, o valor para θ foi 0.1. A partição de dados adotada reserva os últimos 75 pontos para teste, o que parece ser a metodologia mais adotada.

Número de guasianas.	Erro Teste %	Tempo (s)	Num.Gaus. com redução	Erro com redução %	Erro NN %	Erro Abe %
37±1	2.7	49±5	6±1	6.1±0.7	5.3	2.7

Tabela 6.14 - Resultados para o problema *Iris*

A coluna Erro NN refere-se ao erro de classificação da técnica vizinhos mais próximos. Erro Abe refere-se ao resultado divulgado em Abe *et al.* (1994), um sistema neuronebuloso.

Os resultados sem eliminação de gaussianas são iguais aos melhores divulgados. A eliminação de 37 para 6 gaussianas em média eleva o erro, mas altas taxas de acerto ainda são conseguidas. Na seção 6.7 faremos uma análise da base de regras resultante para este problema.

6.7. O Problema *Glass*

O objetivo aqui é classificar vidros dada sua composição química, um problema de interesse criminalístico (Murphy & Aha 1994). Especificamente, existem 6 classes de vidros:

- Janelas de prédios
  - *Float Processed*
  - *Non-float processed*
- Janela de carros (*non-float processed*)
- Containers
- Utensílios de mesa
- Farol de automóveis

São os seguintes os atributos fornecidos:

- Índice de refração
- Composição química em termos dos seguintes componentes: Na,K,Mg,Al,Si,K,Ca,Ba,Fe

O arquivo de dados contém 506 pontos. A divisão entre arquivo de treinamento e de teste foi tomada aleatoriamente. Os valor de θ para este caso foi 0.1.

A Tabela 6.15 a seguir mostra os resultados obtidos para vários tamanhos do arquivo de treinamento (média de três simulações para cada caso, com diferentes arquivos). As simulações duraram entre 1 e 2 minutos.

% Trein.	Num. Gauss.	% Erro Teste	Num Gauss Red	% Erro Teste	% Erro NN
50	97±1	17±1	50±2	20±3	16.5±0.5
30	60±3	31±4	31±3	33±2	26±2
20	37±3	40±6	20±4	40±4	32±6

Tabela 6.15 - Resultados com o problema *Glass*

A coluna Max Var/Gauss refere-se ao maior número de variáveis presentes em uma gaussiana da base de regras

Uma versão simplificada do problema seria a classificação dos vidros em dois tipos (*float processed* e *non-float processed*). Neste caso apenas os exemplos correspondentes às três primeiras classes da lista anterior são considerados. A Tabela 6.16

a seguir exibe os resultados de procedimento semelhante ao anterior para este problema simplificado, com o mesmo valor de θ indicado anteriormente.

% Trein.	Num. Gauss.	% Erro Teste	N. Gaus Red	% Erro Teste	% Erro NN	Max Var/Gaus
50	77±3	11±1	25±3	11±1	12±1	4±1
30	46±2	16±1	14±2	16±1	17±4	3±1
20	31±1	24±4	12±1	24±2	21±4	3
10	16±1	29±3	6±1	32±7	26±3	3±1

Tabela 6.16 - Resultados com o problema GLASS simplificado

Tanto na versão completa do problema quanto na simplificada, obtivemos desempenho semelhante à técnica NN com grande redução de dimensionalidade. Na maioria das simulações, no máximo 4 ou 5 das 9 variáveis de entrada estavam presentes em cada gaussiana da base de regras final. A classificação simplificada em apenas 2 classes permitiu um uso de uma base de regras bem menor. Murphy & Aha (1994) apresentam um resultado de 17.8% de erro com o sistema baseado em regras Beagle, mas sem detalhar a metodologia empregada (arquivos de treinamento e teste), o que torna difícil alguma comparação.

6.8. O problema *Landing*

Este problema baseia-se em informações da equipe de projeto do controlador de pouso do ônibus espacial da NASA (Murphy & Aha 1994). Dadas as condições de vôo, deve ser feita a escolha entre o pouso manual e automático. As variáveis de entrada assumem os seguintes valores:

- **Estabilidade:** 1 - Boa. 2 - Má.
- **Magnitude de erro:** 1 - XL, 2 - LX, 3 - MM, 4 - SS.
- **Sinal de erro:** 1 - Positivo, 2 - Negativo.
- **Direção do vento:** 1 - Cabeça, 2 - Cauda.
- **Magnitude do vento:** 1 - Baixa, 2 - Média, 3 - Forte, 4 - Fora de faixa.
- **Visibilidade** 1 - Boa, 2 - Má.

A Tabela 6.17 mostra a base de regras que cobre todos os casos de interesse, e cuja aplicação gerou os arquivos de dados. Asteriscos representam condições irrelevantes. A regra 7, por exemplo, indica que sob má visibilidade o pouso deve ser automático, não importando as outras condições.

Regra	Estab.	Magn. erro	Sinal erro	Direção vento	Magn. vento	Visib.	Decisão
1	2	*	*	*	*	1	Manual
2	*	*	*	*	4	1	Manual
3	1	2	*	*	*	1	Manual
4	1	1	*	*	*	1	Manual
5	1	3	1	1	3	1	Manual
6	*	*	*	*	*	2	Auto
7	1	4	*	*	1	2	Auto
8	1	3	1	2	3	1	Auto
9	1	4	*	*	3	1	Auto
10	1	4	*	*	2	1	Auto
11	1	3	1	2	2	1	Auto
12a	1	3	1	2	1	1	Auto
12b	1	3	1	1	1	1	Auto

Tabela 6.17: Base de regras original do problema Landing - os números referem-se à lista apresentada no texto

Trata-se de um problema com variáveis discretas, coberto por um conjunto de regras não-nebulosas. Não seria adequado, à primeira vista, como teste para um sistema como SiCAR, totalmente voltado para a manipulação de variáveisa contínuas e extração de regras nebulosas. Mas a existência de uma pequena base de regras que gera todos os padrões de interesse constitui excelente oportunidade de verificação das regras extraídas pelo sistema, pelo que decidimos usar também este problema. A Tabela 6.18 mostra os resultados obtidos. As simulações duraram em média 20 minutos e o valor de θ foi arbitrado em $1x10^{-6}$.

% Treinamento	Num. Gauss.	Erro Teste (%)	Erro NN (%)	Reg. incorretas.
100	13	0.0	0.0	0
80	13±2	1.2±0.7	1.0±0.8	1±1
40	15±7	8±4	5±1	2±1
20	9±1	12±7	13±2	0

Tabela 6.18 - Resultados do problema Landing

Evidentemente, o erro para o caso em que todos os pontos são apresentados para treinamento não poderia ser diferente de zero. O interesse aqui era pela análise da base de regras resultantes (ver seção seguinte). Deve-se notar que, mais uma vez, as taxas de erro obtidas estiveram próximas às resultantes da aplicação da técnica de vizinhos mais próximos.

A última coluna da Tabela 6.18 refere-se ao número de regras incorretas na base de regras final. Definimos regra incorreta como aquela cuja aplicação leva à classificação errônea de algum padrão. Observa-se que raramente regras incorretas foram induzidas. Isto indica que os erros cometidos na classificação foram produzidos principalmente pela falta de informações no arquivo de treinamento. O processo de extração de regras revelou-se robusto, mesmo usando-se apenas 20% dos pontos para treinamento.

6.9. Verificação do conhecimento adquirido

Nesta seção apresentaremos as bases de regras resultantes da aplicação do sistema em alguns dos problemas estudados anteriormente. Procuramos assim verificar se um dos principais objetivos do trabalho foi alcançado: a possibilidade de explicitar conhecimento estruturado a partir da inspeção da estrutura final da rede.

Escolhemos para esta análise os problemas de classificação de padrões, onde as regras estão mais claramente relacionadas a conceitos pertinentes ao problema. Os resultados obtidos com aproximação de funções (seções 6.3, 6.4 e 6.5) já evidenciam que

as regras representam aproximações lineares locais para a função-objetivo, como desejávamos.

6.9.1. *Problema* Iris

Apresentamos a seguir a base de regras resultante para uma das simulações sobre o problema *Iris*. Estas regras produzem classificações corretas sobre todo o arquivo de treinamento e erro de 5.3% sobre o arquivo de teste:

- Se o comprimento da pétala está em torno de 1.7 cm (σ^2=0.32) então a classe é *Setosa*.
- Se a largura da pétala está em torno de 1.0 cm (σ^2=0.11) então a classe é *Versicolor*.
- Se o compriemtno da pétala está em torno de 6.0 cm (σ^2=0.12) ou a largura da pétala está em torno de 2.4 (σ^2=0.14) então a classe é *Virginica*.

É de se notar a extrema simplicidade da base de regras resultante, não apenas quanto ao número de regras, mas também quanto à forma de cada uma delas. Duas variáveis de entrada não aparecem em nenhuma regra. As duas variáveis usadas não aparecem simultaneamente em nenhuma das gaussianas. Isto significa que, apesar de se fazer inicialmente a partição sobre o espaço multidimensional de entrada, a eliminação de variáveis resulta em uma partição simples em um espaço de dimensão reduzida.

Para compreender como esta partição foi obtida, plotamos todos os pontos disponíveis em função das duas variáveis de interesse, o que é mostrado na Figura 6.4. De fato, a projeção determinada pelo sistema revela uma separação simples entre as classes, e se pode verificar facilmente a correção das regras extraídas. O problema de classificação, como previámos, não é difícil. Mas foi o algoritmo particular de treinamento que estudamos que permitiu que esta simplicidade pudesse ser demonstrada.

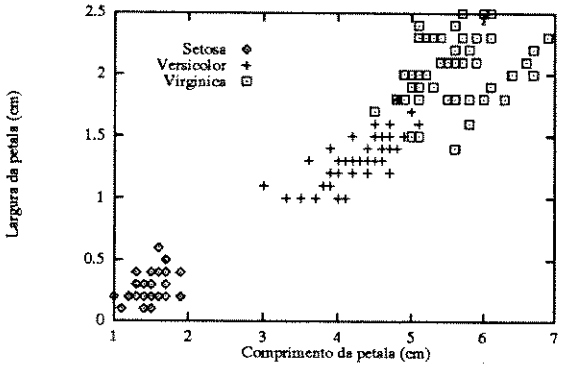


Figura 6.4: Arquivo de treinamento do problema *Iris*

6.9.2. Problema Landing

Faremos a seguir uma comparação entre a base de regras original do problema *Landing* e a extraída pelo sistema com todos os dados disponíveis. A Tabela 6.17 é repetida abaixo para comparação com a Tabela 6.19.No caso da base de regras gerada por SiCAR, há uma coluna adicional (Dist.). Um valor 2 nesta coluna indica que a regra ainda se aplica se uma das condições especificadas for alterada em 1 unidade. Esta não é uma forma usual de representar regras clássicas, e decorre da orientação, presente no algoritmo, no sentido de definir regras nebulosas. Isto não impede, no entanto, a comparação entre as bases de regra.

As duas bases de regras são perfeitamente equivalentes, e as extraídas de SiCAR apresentam algumas simplificações interessantes. Algumas regras extraídas são idênticas às presentes na base de regras original (números 1,2 e 7). Outras apresentam simplificação nas condições (maior número de variáveis irrelevantes, como as número 3, 4 e 11. Na número 3, por exemplo, a estabilidade foi corretamente considerada irrelevante, porque o valor 1 é coberto pela regra 3 original, e um valor 2 seria coberto pela regra 1 original, com mesmo conseqüente.

As regras indicadas por 12a e 12b na base original foram combindadas em uma única regra (12), motivo pelo qual a base de regras resultante é menor do que a original. É fácil notar que a simplificação efetuada foi correta.

Regra	Estab.	Magn. erro	Sinal erro	Direção vento	Magn. vento	Visib.	Decisão
1	2	*	*	*	*	1	Manual
2	*	*	*	*	4	1	Manual
3	1	2	*	*	*	1	Manual
4	1	1	*	*	*	1	Manual
5	1	3	1	1	3	1	Manual
6	*	*	*	*	*	2	Auto
7	1	4	*	*	1	2	Auto
8	1	3	1	2	3	1	Auto
9	1	4	*	*	3	1	Auto
10	1	4	*	*	2	1	Auto
11	1	3	1	2	2	1	Auto
12a	1	3	1	2	1	1	Auto
12b	1	3	1	1	1	1	Auto

Tabela 6.17 - Base de regras original Landing

Regra	Estab.	Magn. erro	Sinal erro	Direção vento	Magn. vento	Visib.	Decisão	Dist.
1	2	*	*	*	*	1	Manual	1
2	*	*	*	*	4	1	Manual	1
3	*	2	*	*	*	1	Manual	1
4	*	1	*	*	*	1	Manual	1
5	*	3	*	1	3	1	Manual	1
6	*	*	*	*	*	2	Auto	1
7	1	4	*	*	1	2	Auto	2
8	1	3	*	2	3	2	Auto	2
9	1	4	*	*	3	2	Auto	2
10	1	4	*	*	2	2	Auto	2
11	1	3	*	*	2	2	Auto	2
12	1	3	*	*	1	2	Auto	2

Tabela 6.19 - Base de regras geradas para o problema Landing

A indicação de uma região de maior aplicação (dist=2) para algumas regras ocorreu nos casos em que o conseqüente indica pouso automático em condições de boa visibilidade, como nas regras 10 e 11. Nestes casos, a regra correspondente em SiCAR indica o pouso para a condição de má visibilidade (mantidas as outras condições) e permite a alteração de uma unidade em qualquer variável. Como existe uma outra regra que indica pouso automático sempre que há má visibilidade (regra 7), não é difícil mostrar que estas regras são de fato equivalentes. Como dissemos antes, esta questão da região de aplicação é decorrência do fato do sistema ser projetado para lidar com variáveis contínuas, e perde um pouco seu sentido neste problema. De qualquer maneira, a base de regras do sistema pôde ser extraída e validada, e foi possível recuperar as regras que originaram os dados fornecidos.

6.10. Introdução de conhecimento parcial

Os últimos experimentos realizados procuraram avaliar até que ponto o sistema desenvolvido permite a introdução de conhecimento sobre o problema estudado, e também o quanto este conhecimento efetivamente funciona como complemento à informação implícita dos exemplos do arquivo de treinamento.

Escolhemos o problema de aproximação de função monodimensional e o problema *Glass* (com 6 classes) para verificar se o desempenho do sistema pode ser melhorado pela introdução de conhecimento.

A Tabela 6.19 mostra as regras introduzidas para o caso da função monodimensional. O sistema passa a dispor da expressão exata do trecho linear da função-objetivo, e de aproximações para dois outros trechos. A qualidade da aproximação promovida pela terceira regra deixa a desejar, e sua introdução visou exatamente verificar a reação do sistema a regras pré-definidas de baixa qualidade. Os valores de σ foram definidos por uma estimativa visual da região em que a aplicação da regra parecia razoável. Esta definição pouco rigorosa também visa testar a robustez do sistema com relação às regras pré-definidas.

Regra	μ	σ^2	w_0	w_1
1	-0.7	0.03	-0.2679	0.732
2	0.9	0.01	0.0	0.0
3	-0.2	0.015	0.25	1.0

Tabela 6.20 - Regras introduzidas para função monodimensional

A Tabela 6.21 compara resultados obtidos com e sem introdução de regras (os parâmetros utilizados foram os mesmos da Tabela 6.7). No caso de trienamento com grande quantidade de pontos, as regras pré-definidas não melhoraram o desempenho do sistema (na verdade, o valor médio de erro foi ligeiramente superior). Isto indica que, neste caso, as regras pré-definidas não trouxeram para o sistema nenhum conhecimento adicional ao que o arquivo de treinamento implicitamente informava.

Caso	Num. Regras	Num. Gaus.	Erro Tr/Val.	Erro Teste	Corr.	Tempo(s)
Comp1	6±1	10±2	0.017±3	0.017±2	1.7±0.2	70±30
Comp1 + Rg	7	11±2	0.016±2	0.020±2	1.1±0.5	40±10
Comp 5	5±1	6±2	0.021±5	0.07±2	0.6±0.3	30±20
Comp 5 +Rg	6±1	6±1	0.015±1	0.043±7	0.3±0.3	16±5
Ruído 0.4	5±1	14±2	0.229±4	0.17±2	20±10	1000±300
Ruído 0.4 +Regras	4±1	10±3	0.225±4	0.07±2	10±10	200±200

Tabela 6.21 - Comparação de desempenho com e sem conhecimento prévio (função monodimensional).

A situação é muito diferente no caso de poucos pontos no treinamento ou de presença de muito ruído. Nestes casos, houve uma grande diminuição do erro médio sobre o arquivo de teste, e também do número de correções e tempo de treinamento, indicando que o conhecimento intrduzido efetivamente auxiliou o sistema.

A Figura 6.5 traz uma comparação do resultado final da aproximação com e sem regras previamente introduzidas, para o caso de presença de ruído elevado. Estão representados os pontos ruidosos do arquivo de treinamento, a função desejada e a obtida. Observar que a amplitude do ruído na Figura 6.5b é o dobro da existente na Fig. 6.5a, e ainda assim, apoiado pelas regras introduzidas, o sistema obteve melhor aproximação.

As Figuras 6.5c e 6.5d mostram a partição inicial resultante da definição prévia das regras, e a partição final após treinamento. Observa-se que o grau de aproveitamento das três regras foi diferente. A regra 1 manteve praticamente a mesma função de pertinência. A regra 2 foi parcialmente aproveitada, e a atuação efetiva da regra 3 ocorre num trecho bem menor do que o indicado anteriormente. O sistema soube definir aproximações melhores do que as iniciais para alguns trechos, e como previámos não confiou inteiramente nas regras fornecidas.

Um efeito curioso ocorreu para a região $0<x<0.2$. A grande quantidade de ruído fez com que o sistema tenha julgado válido combinar o conseqüente da regra desta região com o da regra 1 (observar que a função de saída neste trecho é a continuação da reta correspondente ao trecho linear). Isto indica que a combinação de conseqüente pode produzir erro grande, especialmente em condições de ruído elevado. Neste caso específico, seria melhor impedir esta combinação. É interessante notar também que a representação interna em termos de regras permite que problemas sejam identificados com maior facilidade.

Finalmente, o gráfico 6.6e mostra a evolução do erro para ambos os casos, em termos do número de regras. Aqui também fica claro o efeito das regras introduzidas.

Para o problema *Glass*, observamos o desempenho médio do sistema para três diferentes conjuntos de regras pré-definidas:

- RegI : 2 regras induzidas pelo sistema sobre pontos onde houve erro
- RegII : 11 regras induzidas pelo sistema sobre o arquivo de teste
- RegIII: 23 regras definidas por inspeção dos dados

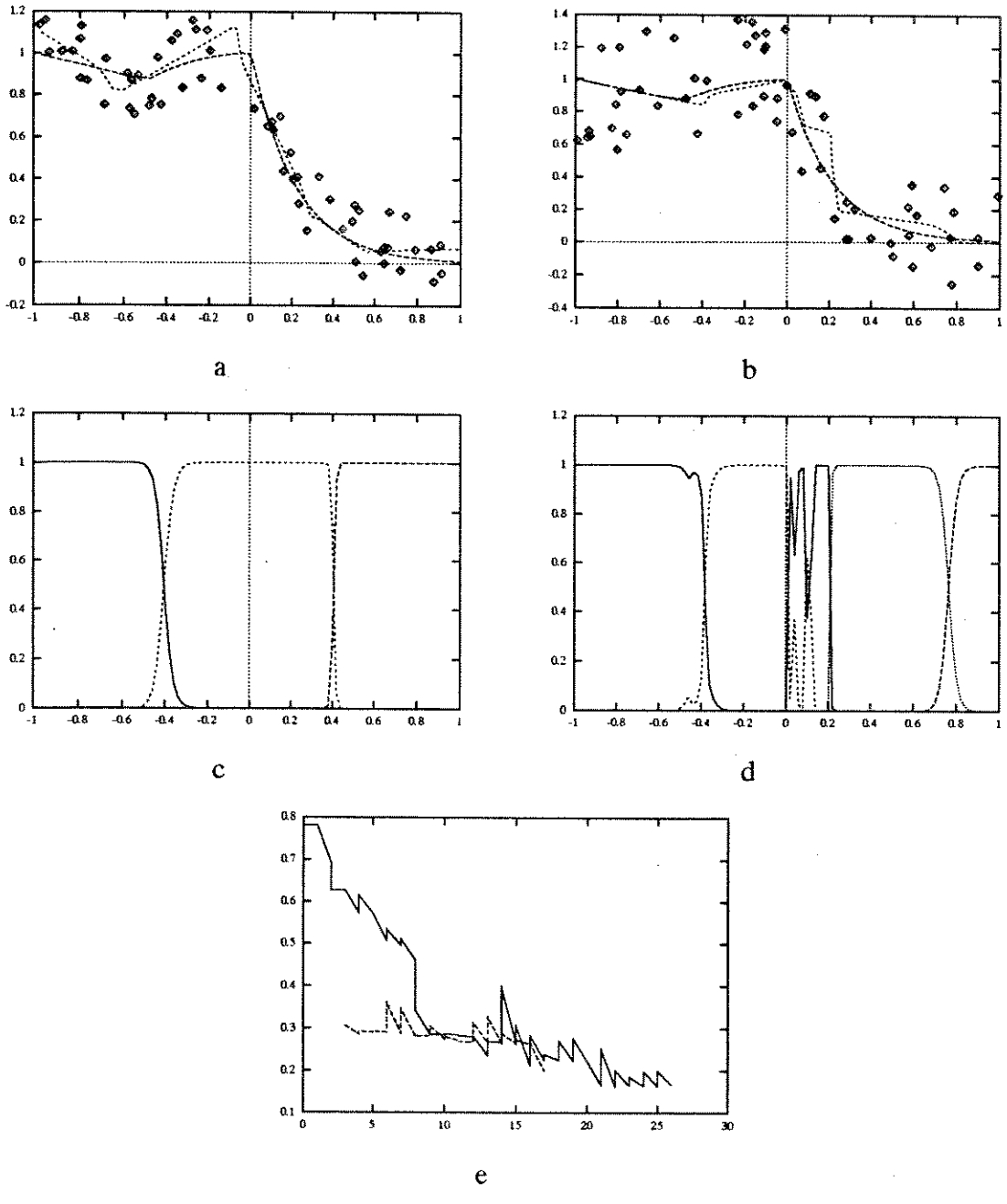


Figura 6.5: Comparação do desempenho sem regras pré-definidas (a) e com regras pré-definidas (b) para o caso de função monodimensional com ruído; partição inicial (c) e final (d) do espaço de entrada; evolução do erro para ambos os casos(e)

A introdução do grupo I e II tem duplo propósito: com ela verificamos a capacidade de estruturação do conhecimento adquirido no treinamento e a capacidade de utilização do conhecimento prévio (o conhecimento é extraído dos exemplos e passado a outro sistema em forma de regras). A Tabela 6.22 mostra o desempenho médio do sistema em cada caso. Verifica-se de imediato que a introdução de regras proporcionou ao sistema conhecimento não contido no arquivo de treinamento. Isto é mostrado com clareza maior ainda nas curvas de evolução de erro (Figura 6.6). O aproveitamento do conhecimento prévio é muito citado na literatura de sistemas neuronebulosos, mas em nossa pesquisa não encontramos nenhuma demonstração (de clareza comparável à da Figura 6.6) de uso efetivo deste conhecimento.

Caso	Núm. Gauss.	Erro Teste	Tempo (s)
Sem regras	50±2	20±3	239±8
Reg I	49±2	18±1	250±20
Reg II	53±1	17.4±0.3	388±8
Reg III	67±3	15.3±0.7	240±10

Tabela 6.22: Desempenho no problema *Glass* com vários graus de conhecimento prévio

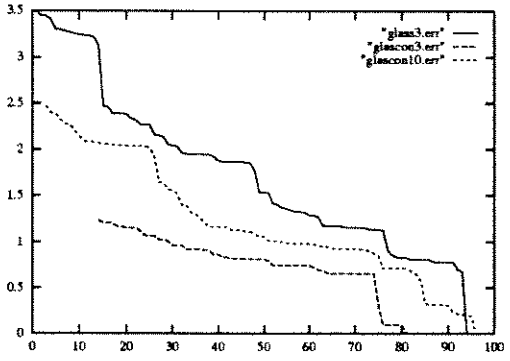


Figura 6.6: Evolução de erro (problema *Glass)* para diferentes condições de conhecimento inicial.

Apresentados todos os experimentos realizados e os resultados obtidos, devemos agora fazer uma síntese de todo o trabalho, avaliando até que ponto nossos objetivos foram alcançados, discutindo as dificuldades não previstas na definição inicial do sistema, as virtudes e limitações da forma final do algoritmo. Este é o propósito do capítulo seguinte, onde são apontadas ainda várias linhas de pesquisa inspiradas pela experência adquirida durante o trabalho.

Capítulo 7

Conclusões

Nosso trabalho foi guiado pela intenção de estruturar o conhecimento adquirido por uma rede neural. As linhas de pesquisa divulgadas na literatura indicavam os sistemas neuronebulosos como um caminho promissor nesta direção, e o estudo destas arquiteturas serviu como ponto de partida.

Foi verificado que a aquisição de conhecimento não é obtida de imediato pela simples associação de elementos da rede com conceitos de lógica nebulosa. Uma partição específica do espaço de entrada é necessária para que desta associação resulte uma base de regras compreensível.

Desenvolveu-se então um algoritmo que enfatiza o desenvolvimento incremental de regras, o aperfeiçoamento local das regras desenvolvidas (ao invés da otimização global da estrutura resultante) e a cooperação entre regras pré-definidas e induzidas. Promove-se a adaptação estrutural do sistema em paralelo ao ajuste de parâmetros. Vários experimentos foram efetuados com o algoritmo, com ênfase no estudo de três aspectos: desempenho do sistema em comparação a outros algoritmos, qualidade das bases de regras resultantes e possibilidade de introdução de conhecimento prévio no sistema.

7.1. Comparação com outros algoritmos

De imediato, verificou-se que os vários parâmetros definidos no algoritmo não introduzem problemas de sensibilidade. O desempenho do sistema (em termos de erro resultante sobre o arquivo de teste) esteve próximo, na grande maioria dos casos, aos obtidos com outras técnicas (vizinhos mais próximos, perceptron multicamadas) e resultados divulgados na literatura. Uma exceção importante é o caso do treinamento com pontos muito esparsos, que revelou uma

limitação do sistema, possivelmente relacionada à falta de uma condição global de suavidade da função resultante da combinação dos vários conseqüentes.

Pode-se dizer que o preço para se obter operações inteligíveis em um sistema conexionista não foi alto, especialmente nos problemas de reconhecimento de padrões. O algoritmo proposto trouxe ainda duas vantagens com as quais não contávamos a princípio.

Em primeiro lugar, uma grande redução de dimensão do espaço do problema foi conseguida nos casos multidimensionais. Mais importante, a eliminação de variáveis foi obtida com testes muito simples sobre a base de regras, que não trouxeram nenhuma sobrecarga computacional significativa.

Em segundo lugar, o caráter local e incremental do desenvolvimento de regras proposto trouxe uma notável redução no custo computacional do treinamento. Para a grande maioria dos problemas tratados, poucos minutos foram suficientes para todo o processo de desenvolvimento de regras e avaliação de desempenho.

7.2. Qualidade das bases de regras resultantes

A partição final obtida para os problemas de aproximação de funções são condizentes com a idéia de descrição das operações da rede em termos de regras. As regras desenvolvidas são nebulosas, compartilham seu espaço de atuação com regras próximas, mas a interação entre elas não chega a impedir a identificação de regras dominantes (isto nos levaria a algo próximo à representação distribuída das redes convencionais). Obteve-se assim um bom equilíbrio entre o caráter nebuloso da base de regras e a possibilidade de interpretação das mesmas. Esta é uma questão que julgamos importante, e que não é contemplada nos sistemas neuronebulosos que estudamos.

Tudo isto pôde ser comprovado nos problemas de classificação de padrões, nos quais as operações da rede puderam ser relacionadas facilmente a regras envolvendo variáveis relevantes do problema. Deve-se destacar o ocorrido no problema *Landing*, onde a base de regras (não nebulosa) foi perfeitamente recuperada a partir dos exemplos, e algumas simplificações interessantes foram obtidas.

7.3. Aproveitamento de conhecimento introduzido *a priori*

Os cuidados especiais presentes no algoritmo com relação às regras pré-definidas revelaram-se acertados. A partição incremental e a otimização local impediram que a base de regras fornecida *a priori* fosse desfigurada durante o treinamento. Com isso, observou-se uma notável cooperação entre regras pré-definidas e arquivo de treinamento. O sistema efetivamente utilizou regras dadas como um complemento aos exemplos do arquivo, mas também soube desenvolver suas próprias regras onde as pré-definidas não se mostraram úteis. Não conhecemos na literatura resultados que revelem tão claramente a combinação de conhecimento explícito (regras) e implícito (exemplos).

7.4 Pesquisa futura

A investigação de mecanismos de representação de conhecimento em sistemas conexionistas apresenta inúmeras possibilidades. Este trabalho representa apenas a investigação de uma idéia intuitiva sobre desenvolvimento de regras. Apontamos a seguir alguns aspectos que merecem estudos mais aprofundados no sistema desenvolvido, e caminhos alternativos de pesquisa que não foram explorados.

Em primeiro lugar, é necessário investigar formas de melhorar a suavidade da função resultante na saída. Como afirmamos anteriormente, esta parece ser a origem da principal limitação do sistema (capacidade de generalização mais fraca do que a do perceptron multicamadas). Acreditamos que esta condição de suavidade possa se traduzir em termos de restrições sobre os conseqüentes de regras espacialmente próximas.

O procedimento de eliminação de variáveis no antecedente, que revelou bons resultados, não leva em conta a importância relativa das variáveis (a ordem em que se tenta retirar variáveis é fixa). Seria interessante investigar o uso de técnicas como análise de sensibilidade para procurar projeções ótimas sobre as variáveis relevantes.

A partição do espaço de entrada desenvolvida pelo sistema é toda baseada em uma heurística sobre região de aplicação de regras. Seria desejável uma cuidadosa análise do que se pode dizer sobre esta heurística à luz dos diversos resultados teóricos sobre capacidade de representação de redes neurais, e de redes de funções radiais de base em particular.

Nossa pesquisa concentrou-se sobre o chamado treinamento supervisionado, no qual dispomos de um "professor" que fornece informações suficientes para o cálculo exato do erro cometido. No entanto, uma análise mais detalhada do procedimento adotado para avaliação de regras revela que a única informação efetivamente utilizada é o tamanho da região na qual o erro cometido está dentro da faixa especificada. Isto parece indicar um ponto de contato do algoritmo com *reinforcement learning*, onde o sinal utilizado para ajuste de parâmetros é pouco informativo (do tipo sucesso/fracasso).

Nós também não exploramos a aplicação das idéias aqui desenvolvidas no problema de identificação e controle de sistemas dinâmicos. Pesquisas em controle inteligente têm dado grande importância a redes de caráter local (como as RBFN's) e a aprendizado estrutural. Uma vez que as mesmas questões foram tratadas aqui, este nos parece um caminho natural de extensão do trabalho. Em particular, cremos que a atuação de um controlador já existente sobre o processo pode fornecer um conjunto inicial de regras para posterior otimização. A boa cooperação obtida entre regras pré-definidas e induzidas indica ainda a possibilidade de definir regras de segurança *a priori*, evitando atuação inaceitável do sistema durante o treinamento.

As idéias desenvolvidas neste trabalho só estarão plenamente validadas quando forem aplicadas em problemas de grande complexidade, nos quais exista conhecimento especialista (na forma de regras nebulosas) que cubra parte do domínio. Então será possível verificar as vantagens de um "diálogo" com a rede em termos de conhecimento estruturado.

O trabalho que agora concluímos revelou que o estudo da representação do conhecimento nos sistemas conexionistas é um campo fértil de pesquisa. Os bons resultados obtidos com uma simples heurística de avaliação de regras a partir do erro no conseqüente indicam que há caminhos para obter-se de uma rede de processadores subsimbólicos simples comportamento global estruturado e que lida com símbolos, ao menos no conceito estendido de símbolo no escopo da teoria de conjuntos nebulosos. Acreditamos que esta linha de pesquisa

possa vir a sugerir caminhos viáveis de aplicação das redes neurais em tarefas inteligentes de alto nível.

Bibliografia

S.Abe, M.S.Lan & R.Thawonmas (1994) Tuning of a fuzzy classifier derived from data. *Proceedings of FUZZ-IEEE 93.*

R.Babuska, R.Jager & H.B.Verbruggen (1994) Interpolation issues in Sugeno-Takagi reasoning. *Proceedings of FUZZ-IEEE 93.*

J.F.Baldwin & B.W.Pilsworth (1980) Axiomatic approach to implication for approximate reasoning with fuzzy logic. *Fuzzy Sets and Systems* 3 pp 193-219.

A.G.Barto (1989) Connectionist learning for control: an overview. Relatório técnico COINS 89-89. Amherst, EUA: University of Massachusetts. Department of Computer and Information Sciences.

A.G.Barto, R.S.Sutton & C.W.Anderson (1983) Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics* v. 13 pp. 835-846.

C.M.Bishop (1994) Novelty detection and neural newtork validation. Preprint (Internet). Reino Unido: Aston University. Department of Computer Science.

J.Bruske, E. von Puttkamer & U.R.Zimmer (1993) SPIN-NFDS : Learning and preset knowledge for surface fusion - a neural fuzzy decision system. *Proc. ANZIIS,* Perth Western, Australia.

T.W.Cacciatore & S.J.Nowlan (1994) Mixtures of controllers for jump linear and non-linear plants. Preprint (Internet).

D.A.Cohn (1994) Neural network exploring using optimal experimental design. Preprint (Internet). Cambridge, EUA: Massachusetts Institute of Technology. Department of Brain and Cognitive Science.

T.M.Cover & P.E.Hart (1967) Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* v. 13 pp. 21-27.

E.Czogala & W.Pedrycz (1981) Some problems concerning the construction of algorithms of decision-making in fuzzy systems. *International Journal on Man-Machine Studies* v. 15 pp 201-221.

(1982) Fuzzy rules generation for fuzzy control. *Cybernetics and Systems* v. 13 pp. 275-293.

D.Dubois & H.Prade (1986) Fuzzy sets and statistical data. *European Journal of Operational Research* v. 25 pp. 345-356.

(1989) Fuzzy sets, probability and measurement. *European Journal of Operational Research* v. 40 pp 135-154.

(1994) Fuzy sets - a convenient fiction for modeling vagueness and possibility. *IEEE Transactions on Fuzzy Systems* v. 2 n. 1 pp. 16-21.

M.G.Dyer (1988) The promise and problems of connectionism. Comentário ao artigo de Smolensky (1988).

J.A.Fodor & Z.W.Pylyshyn (1988) Connectionism and cognitive architecture : a critical analysis. *Cognition* n.28 pp. 3-71.

J.Fox (1981) Towards a reconcilliation of fuzzy logic and standard logic. *International Journal on Man-Machine Studies* v.15 pp. 213-220.

B.R.Gaines (1977) Foundations of fuzzy reasoning. In: M.M.Gupta, G.N.Saridis & B.R.Gaines (Eds.): *Fuzzy automata and decision processes.* Holanda: North-Holland.

P.Gentric & H.C.A.M.Withagen (1994) Constructive methods for a new classifier based on a radial-basis-function neural network accelerated by a tree. Preprint (Internet). França: Laboratoires d"Electronique Philips.

F.Girosi (1992) Some extensions of radial basis functions and their application in artificial intelligence. *Computers Math. Applic.* v. 24 n. 12 pp. 61-80.

F.Girosi, M.Jones & T.Poggio (1994) Priors, stabilizers and basis functions: From regularization to radial, tensor and additive splines. Memo No. 1430. Cambridge, EUA: Massachusetts Institute of Technology. Artificial Intelligence Laboratory.

M.M.Gupta (1977) "Fuzzy-ism", the first decade. In M.M.Gupta, G.N.Saridis & B.R.Gaines (Eds.): *Fuzzy automata and decision processes.* Holanda: North-Holland.

J.P.Hayes (1988) *Computer Architecture and Organization.* Nova Iorque, EUA: McGraw-Hill.

R.Hecht-Nielsen (1991) *Neurocomputing*. Reading, EUA: Addison-Wesley Publishing Company.

G.E. Hinton, J.L. McClelland & D.E. Rumelhart (1986) Distributed representations. In Rumelhart *et al.* 1986.

J.J.Hopfield (1982) Neural networks and physical systems with emergent collective computacional abilities. Reimpresso em J.A.Anderson & E.Rosenfeld (Eds.): *Neurocomputing: Foundations of Research*. Cambridge, EUA: The MIT Press 1988.

E.B.Hunt (1975) *Artificial Intelligence*. Nova Iorque, EUA: Academic Press.

K.J.Hunt, D.Sbarbaro, R.Zbikowski & P.J. Gawthrop (1992) Neural networks for control systems - a survey. *Automatica* v.28 n. 6 pp. 1083-1112

R.A.Jacobs, M.Jordan & A.Barto (1990) Task decomposition through competition in a modular connectionist architecture : The what and where vision tasks. Relatório Técnico 90-27. Amherst, EUA: University of Massachusetts. Department of Computer Science.

W. James (1890) Psychology (Briefer course). Reimpresso em J.A.Anderson & E.Rosenfeld (Eds.): *Neurocomputing: Foundations of Research*. Cambridge, EUA: The MIT Press 1988.

J.S.R.Jang (1992) ANFIS: Adaptive-network-based fuzzy inference system. Preprint (Internet). Berkeley, EUA: University of California. Department of Electrical Engineering and Computer Science.

J.S.Jang & C.T.Sun (1992) Functional equivalence between radial basis function networks and fuzzy inference systems. Preprint (Internet). Berkeley, EUA: University of California. Department of Electrical Engineering and Computer Science.

M.I.Jordan & R.A. Jacobs (1993) Hierarchical mixtures of experts and the EM algorithm. Memo No. 1440. Cambridge, EUA: Massachusetts Institute of Technology. Artificial Intelligence Laboratory.

J.M.Keller, R.R.Yager & H.Tahani (1992) Neural network implementation of fuzzy logic. *Fuzzy Sets and Systems* v. 45 pp. 1-12.

J.B.Kiska, M.E.Kochanska & D.S.Sliwinska (1985) The influence of some fuzzy implication operators on the accuracy of a fuzzy model. *Fuzzy Sets and Systems* v. 15 pp. 111-128 (Part I) e pp. 223-240 (Part II).

T.Kohonen (1984) *Self-Organization and Associative Memory*. EUA: Springer-Verlag..

B.Kosko (1992) *Neural Networks and Fuzzy Systems; A Dynamical System Approach to Machine Intelligence.* Englewood Cliffs, EUA: Prentice-Hall.

M.Laviolette & J.W.Seeman Jr.(1994) The efficiency of fuzzy representations of uncertainty. *IEEE Transactions on Fuzzy Systems* v. 2 n. 1 pp. 4-15.

C.C.Lee (1990) Fuzzy logic in control systems: fuzzy logic controller. *IEEE Transactions on Systems, Man and Cybernetics* v.20 n. 2 pp. 404-433.

C.T.Lin & C.S.G.Lee (1991) Neural-network based fuzzy logic control and decision system. *IEEE Transactions on Computers* v. 40 n. 12 pp. 1320-1336.

T.E.Little (1992) *The Neural Shell Version 3.5: A Customizable X WIndow System Interface for Neural Network Simulation.* Columbus, EUA: The Ohio State University. Department of Electrical Engineering. SPANN Laboratory.

B.MacLennan (1991) Characteristics of connectionist knowledge representation. Relatório técnico CS-91-147. Knoxville, EUA: University of Tenessee. Computer Science Department.

J.L.McClelland, D.E.Rumelhart & G.E.Hinton (1986) The appeal of parallel distributed processing. In Rumelhart *et al.* 1986

W.S.McCulloch & W.Pitts (1943) A logical calculus of the ideas immanent in nervous activity. Reimpresso em J.A.Anderson & E.Rosenfeld (Eds.): *Neurocomputing: Foundations of Research.* Cambridge, EUA: The MIT Press 1988.

C.McMillan, M.C.Mozer & P.Smolensky (1992) The Connectionist Scientist Game: Rule extraction and refinement in a neural network. Preprint (Internet). Boulder, EUA: University of Colorado. Department of Computer Science and Institute of Cognitive Science.

R.Meir (1994) Bias, variance and the combination of estimators; The case of linear least squares. Preprint (Internet). Haifa, Israel: Technion. Department of Electrical Engineering.

M.Minsky (1988) Connectionist models and their prospects. Prefácio de D.Waltz & J.A.Feldman (Eds.) *Connectionist Models and Their Implications: Readings from Cognitive Science.* Norwood, EUA: Ablex Publishing Corporation.

M.Minsky & S.Papert (1969) *Perceptrons.* Introduçao reimpressa em J.A.Anderson & E.Rosenfeld (Eds.): *Neurocomputing: Foundations of Research.* Cambridge, MA, EUA: The MIT Press 1988.

P.M.Murphy & D.W.Aha (1994): Repositório de bases de dados sobre aprendizado em máquina da UCI. Irvine, EUA: University of California. Department of Information and Computer Sciences. `ftp ics.uci.edu` Diretório: `pub/machine-learning databases`.

J. von Neumann (1945): First draft of a report on the EDVAC. Reimpresso em B.Randell (ed.) *The Origins of Digital Computers: Selected Papers.* Berlim, Alemanha: Springer-Verlag 1973.

D.Nguyen & B.Widrow (1989): The truck backer-upper: an example of self-learning in neural networks. *Proceedings of the International Joint Conference on Neural Networks* pp. 357-363. Washington, EUA.

N.J.Nilsson (1980) *Principles of Artificial Intelligence.* Palo Alto, EUA: Tioga Publishing Company.

P.Niyogi & F.Girosi (1994) On the relationship between generalization error, hypothesis complexity and sample complexity for radial basis functions. Memo No. 1467. Cambridge, EUA: Massachusetts Institute of Technology. Artificial Intelligence Laboratory.

C.W.Omlin & C.L.Giles (1992) Training second-order recurrent neural networks using hints. In P.Edward (ed.): *Proc. of the 9th International Confeence on Machine-learning.*

W.Pedrycz (1994) Fuzzy neural networks. Concepts and architectures. *Brazil-Japan Joint Symposium on Fuzzy Systems.* Campinas, Brasil.

L.Prechelt (1994) Proben 1 - A set of neural networks benchmark problems and benchmarking rules. Relatório Técnico 21/94. Karlsruhe, Alemanha: Universität Karlsruhe. Fakultät für Informatik.

L.Prechelt (1994b) A study of experimental evaluation of neural network learning algorithms: current research practice. Relatório Técnico 19/94. Karlsruhe, Alemanha: Universität Karlsruhe. Fakultät für Informatik.

N.Rescher (1969) *Many-Valued Logic.* Nova Iorque, EUA : McGraw-Hill.

M.Riedmiller (1994) Advenced supervised learning in multi-layer perceptrons - From backpropagation to adaptive learning algorithms. Preprint (Internet).

A.Röbel (1994) The dynamical pattern selection algorithm: effective training size and controlled generalization performance of back-propagation neural networks. Preprint (Internet). Berlim, Alemanha: Tecnhische Universität.

F.Rosenblatt (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Reimpresso em J.A.Anderson & E.Rosenfeld (Eds.): *Neurocomputing: Foundations of Research.* Cambridge, MA, EUA: The MIT Press 1988.

D.E.Rumelhart & J.L.McClelland (1986) PDP models and general issues in cognitive science. In Rumelhart *et al.* 1986.

D.E.Rumelhart, J.L.McClelland & the PDP Research Group (1986) (Eds.) *Parallel Distributed Processing: Explorations in the microstructure of cognition.* Cambridge, EUA: The MIT Press.

D.E.Rumelhart, G.E.Hinton & R.J.Williams (1986b) Learning internal representations by error propagation. In Rumelhart *et al.* 1986

D.E.Rumelhart, P.Smolensky, J.L.McClelland & G.E.Hinton (1986c) Schemata and sequential thought processes in PDP models. In: Rumelhart *et al.* 1986

S.Shao (1988) Fuzzy self-organizing controlller and its application for dynamic processes. *Fuzzy sets and systems* v. 26 pp 151-164.

P.Smolensky (1988) On the proper treatment of connectionism. *Behavioral and Brain Sciences* v. 11 n.1 pp 1-74.

M.Sugeno (1977) Fuzzy measures and fuzzy integral: a survey. In M.M.Gupta, G.N.Saridis & B.R.Gaines (Eds.) *Fuzzy automata and decision processes.* Holanda: North-Holland.

M.Sugeno & M.Nishida (1985) Fuzzy control of model car. *Fuzzy sets and systems* v. 16 pp. 103-113.

M.Sugeno & G.T.Kang (1988) Structutre identification of fuzzy model. *Fuzzy sets and systems* v. 28 pp. 15-33.

H.Takagi (1993) Fusion techniques of fuzzy systems and neural networks, and fuzzy systems and genetic algorithms. *SPIE Proceedings of Technical Conference on Applications of Fuzzy Logic Technology.*

T.Takagi & M.Sugeno (1985) Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics* v.15 n. 1, pp. 116-132.

V.Tresp, J.Hollatz & S.Ahmad (1993) Network structuring and training using rule-based knowledge. In: C.L.Giles, S.J.Hanson & J.D.Cowan (Eds): *Advances in neural information processing systems 5* . EUA: Morgan Kaufman.

V.Tresp, S.Ahmad & R.Neunier (1993b) Training NN with deficient data. Preprint (Internet).

Y.Tsukamoto (1979) An approach to fuzzy reasoning method. In: M.M.Gupta, R.K. Ragade & R.R. Yager (eds): *Advances in fuzzy set theory and applications.* Holanda: North-Holland.

P.Turney (1990) The gap between abstract and concrete results in machine learning. Preprint (Internet). Canadá: National Research Council. Institute for Information Technology.

P.Turney (1993) A theory of cross-validation error. Preprint (Internet). Canadá: National Research Council. Institute for Information Technology.

G.Wahba (1994) Generalization and regularization in nonlinear learning systems. Relatório Técnico No. 921. EUA: University of Wisconsin. Department of Statistics.

D.Waltz & J.A.Feldman (1988) Connectionist models and their implications. In D.Waltz & J.A.Feldman (Eds.) *Connectionist Models and Their Implications: Readings from Cognitive Science.* Norwood, EUA: Ablex Publishing Corporation.

C.Wang, S.S.Venkatesh & J.S.Judd (1993) Optimal stopping and effective machine complexity in learning. Preprint (Internet). EUA: University of Pennsylvania. Department of Systems Sciences and Engineering.

B.Widrow & M.A.Lehr (1990) 30 years of adaptive neural networks: perceptron, madaline and backpropagation. *Proceedings of the IEEE* v. 78 n. 9 pp. 1415-1442.

E.Wong(1971) *Stochastic processes in information and dynamical systems.* EUA McGraw-Hill.

C.W.Xu & Y.Z.Lu (1987) Fuzzy model identification and self-learning for dynamic systems. *IEEE Transactions on Systems, Man and Cybernetics* v. 17 n. 4 pp. 683-696.

X.Ye & N.K.Loh (1993): Dynamical system identification using recurrent radial basis function networks. *Proceedings of the American Control Conference.* São Francisco, EUA.

L.A.Zadeh (1965) Fuzzy sets. *Information and Control* n. 8 pp. 338-353.

(1973) Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics* v.3 n.1 pp. 28-44.

(1975) Fuzzy logic and approximate reasoning. *Synthese* 30 pp. 407-428.

(1977) Fuzzy set theory - a perspective. In: M.M.Gupta, G.N.Saridis & B.R.Gaines (eds): *Fuzzy automata and decision processes.* Holanda: North-Holland.

(1978) Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems* 1 pp. 3-28.

R.Zbikowski, K.J.Hunt, A.Dzielínski, R.Murray-Smith & P.J.Gawthrop (1994): A review of advances in neural adaptive control systems. Artigo Técnico TP-1. NACT Project. European Comission.