UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação
Departamento de Engenharia de Computação e Automação Industrial

Redes Neurais Fuzzy Aplicadas em Identificação e Controle de Sistemas

Autora: **Ivette R. Luna Huamaní**

Orientador: **Prof. Dr. Fernando Gomide**
Co-orientadora: **Prof. Dra. Rosangela Ballini**

Banca Examinadora:
Prof. Dr. Fernando Gomide (DCA/FEEC/UNICAMP)
Prof. Dr. Walmir Matos Caminhas (DEE/UFMG)
Prof. Dr. Peter Sussner (IMECC/UNICAMP)
Prof. Dr. Wagner Caradori do Amaral (DCA/FEEC/UNICAMP)

Dissertação submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, para o preenchimento dos pré-requisitos parciais para obtenção do Título de Mestre em Engenharia Elétrica.

Agosto - 2003

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

Luna Huamaní, Ivette

L971s Redes neurais fuzzy aplicadas em identificação e controle de sistemas / Ivette Luna Huamaní. −− Campinas, SP: [s.n.], 2003.

Orientadores: Fernando Gomide e Rosangela Ballini
Dissertação (mestrado) − Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes neurais (Computação). 2. Lógica difusa. 3. Identificação de sistemas. I. Gomide, Fernando. II. Ballini, Rosangela. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Resumo

Este trabalho apresenta um estudo comparativo entre redes neurofuzzy híbridas, redes neurais e sistemas fuzzy, aplicados a problemas de identificação e controle de sistemas dinâmicos não lineares. Devido à necessidade de representação temporal e de elementos de memória, para a resolução dos problemas tratados, duas estruturas de redes neurofuzzy recorrentes são propostas, a partir de uma rede neurofuzzy estática. As relações temporais são induzidas por realimentação local e global internas nas redes neurofuzzy recorrentes. Para a aprendizagem das redes neurofuzzy propõe-se um algoritmo baseado no método do gradiente e no método de treinamento por reforço associativo. Resultados de simulação mostram que as redes neurofuzzy propostas proporcionam uma alternativa efetiva para modelar e controlar sistemas dinâmicos não lineares.

Abstract

This work compares the performance of neural fuzzy, neural network and fuzzy systems, to model and control non-linear dynamical systems. Due to the need of temporal representations, two recurrent neural fuzzy networks are proposed based on an hybrid static neural fuzzy architecture. Temporal processing is induced by local and global recurrence in the hidden layer neurons. A learning method based on gradient search and associative reinforcement learning is proposed. Computational experiments suggest that recurrent neural fuzzy networks provide an effective alternative to model and control non-linear dynamical systems.

A mis padres Héctor y Rosenda.

Agradecimentos

Agradeço aos professores Fernando Gomide e Rosângela Ballini pela orientação e paciência ao longo deste tempo.

Ao CNPQ pelo suporte financeiro.

Aos meus pais Héctor e Rosenda, por serem o maior presente divino de amor que Deus me deu na vida.

Aos meus grandes amigos da graduação José Luis e Ricardo, pelo apoio constante a distância.

Aos meus amigos da FEEC/UNICAMP, Edgar, Leila, Gisselle, Virginia, Marina, Ivana, Raquel, Marcia, Edilson, Luiza, Pepe e tantos amigos pelas discussões, sugestões e calorosa acolhida durante estes anos.

Finalmente, agradeço a Deus por tudo o que ele me oferece.

Sumário

Lista de Figuras

Lista de Tabelas

Notação e Simbologia

n Número de componentes da entrada do sistema ou rede neural/neurofuzzy.

M Número de neurônios na camada intermediária da rede neural/neurofuzzy.

p Número de componentes da saída do sistema ou rede neural/neurofuzzy.

$\mathbf{x}$ Vetor de entrada x.

$\mathbf{y}$ Vetor de saída y.

x_i $i-$ésima componente do vetor x.

$\hat{y}_i$ Valor estimado de y_i.

z_j Saída do $j-$ésimo neurônio da camada intermediária.

$f(\cdot)$, $\psi(\cdot)$ Função de ativação.

$x\ s\ y$ Norma triangular do tipo OR (x $s-$norma y).

$x\ t\ y$ Norma triangular do tipo AND (x $t-$norma y).

q^{-1} Operador atraso.

w_{ji} Peso entre o neurônio i da camada de entrada e o neurônio j da camada intermediária.

r_{jl} Peso de recorrência do neurônio l ao neurônio j na camada intermediária.

v_{kj} Peso entre o neurônio j da camada intermediária e o neurônio k da camada de saída.

c_{ij} $j-$ésimo centro da $i-$ésima componente do vetor de entrada $\mathbf{x}$.

N_i Número de conjuntos nebulosos na partição do espaço de x_i.

$\alpha_1,\ \alpha_2,\ \alpha_3,\ \alpha_4,\ \eta$ Taxas de aprendizado.

t Tempo.

EQM Erro quadrático médio.

LVQ *Learning Vector Quantization*.

Capítulo 1

Introdução

Redes neurais e sistemas fuzzy são dois importantes componentes da área de inteligência computacional. Estes componentes procuram desenvolver sistemas que apresentem alguma forma de inteligência similar à exibida por determinados sistemas biológicos, embora não sejam inspirados propriamente por algum sistema natural, mas sim, por algum tipo de comportamento observado em sistemas naturais (Iyoda 2000).

As redes neurais são uma alternativa para a solução de problemas de identificação e controle envolvendo não linearidades nas dinâmicas inerentes aos sistemas. Contudo, elas apresentam algumas questões quanto a determinação da estrutura necessária para representar um determinado sistema, técnicas de treinamento e as dinâmicas temporais envolvidas no processamento (de Moraes Lima 2000).

Uma importante característica dos sistemas fuzzy é sua capacidade de fornecer descrições da relação entre os sinais de entrada e saída de um determinado sistema (Yager & Filev 1994). No entanto, estes sistemas não possuem uma capacidade inerente de aprendizagem.

O interesse no estudo da combinação de redes neurais com a teoria de conjuntos fuzzy e a lógica associada têm tido um grande crescimento, com aplicação nas mais diversas áreas. Em particular tem proporcionado soluções para o tratamento de sistemas dinâmicos não lineares, com resultados promissores e modelos computacionais eficientes (Buckley & Hayashi 1995). Estes modelos, conhecidos como *sistemas neurofuzzy*, são exemplos de sistemas híbridos, dentre outros existentes na área da inteligência computacional (Giles et al. 1999).

Diversas aplicações de redes neurais, sistemas fuzzy e sistemas neurofuzzy podem ser encontrados na literatura, sendo algumas destas:

- Detecção e diagnóstico de falhas em sistemas dinâmicos (Caminhas 1997);
- Classificação de padrões (Brouwer 2000), (Féraud & Clérot 2002);
- Diagnóstico médico (Mitra & Hayashi 2000);
- Aproximação de funções (Figueiredo & Gomide 1999);
- Identificação e controle de sistemas (Nguyen & Widrow 1990), (Savkovic-Stevanovic 1996), (Hussain 1999);
- Predição de séries temporais (Lin & Cunningham III 1995);
- Filtragem adaptativa não linear (Nerrand et al. 1993), etc.

Em particular, o projeto de controladores, utilizando redes neurofuzzy foi sugerido em (Lee et al. 1995). Nesta proposta, um identificador neural aproxima um sistema fuzzy e o modelo obtido é utilizado para representar o sistema a ser controlado. O controlador foi projetado de forma a fornecer um sinal de controle ótimo quando trabalhando em malha fechada.

Lin e Cunningham III (1995), propuseram um sistema neurofuzzy para identificação de sistemas. Este sistema foi utilizado para identificar as variáveis de entrada de maior relevância, determinar a estrutura do modelo e inicializar os pesos de uma arquitetura neurofuzzy (Lin & Cunningham III 1995).

Uma rede neurofuzzy estática de duas camadas intermediárias, formada por neurônios lógicos do tipo AND e OR foi proposta em (Caminhas et al. 1999) incluindo um método alternativo para o treinamento da estrutura neurofuzzy. A rede foi testada e aplicada a problemas de classificação de padrões.

Em (Iyoda 2000), foi proposto uma rede neurofuzzy estática que permite uma composição aditiva ou multiplicativa em cascata de diferentes funções de ativação, selecionadas a partir de um conjunto finito de candidatas, sendo a rede treinada via algoritmos genéticos.

Contudo, uma limitação das redes neurofuzzy estáticas, como aquelas mencionadas acima, é, ainda, a sua restrita aplicação devido à estrutura não recorrente ou à falta de mecanismos eficientes de aprendizado para as conexões de realimentação.

Um dos primeiros modelos de sistemas fuzzy adaptativos recorrentes foi proposto em (Bersini & Gorrini 1994). Neste modelo, o sistema de inferência é composto por dois subsistemas, sendo o primeiro para determinar o valor das variáveis internas envolvidas na recorrência e o segundo responsável pelo cálculo da saída do sistema. O modelo é treinado via retropropagação do erro.

Em (Lee & Teng 2000), foi proposta uma rede neurofuzzy recorrente aplicada a controle de processos, onde as funções de pertinência associadas a base de regras que forma a rede neural também são ajustadas via retropropagação do erro.

(Lin & Wai 2001) desenvolveram uma arquitetura neurofuzzy recorrente para controlar um motor de indução. O controlador neurofuzzy é treinado *on-line*, utilizando o método do gradiente. Os autores também mostram que o sistema de controle é estável utilizando o método de Lyapunov.

Assim como as redes neurais recorrentes, as redes neurofuzzy recorrentes podem ser classificadas em duas grandes classes: redes parcial ou totalmente recorrentes. Recorrência parcial ou total são conseqüência do tipo de conexão de realimentação entre as unidades que compõem a rede. Em ambos os casos, a realimentação é o mecanismo que permite a criação de representações internas e de memória capazes de processar e armazenar informações temporais e seqüenciais (Von Zuben 1996).

Em geral, redes recorrentes são mais difíceis de serem analisadas, o seu processo de treinamento é mais trabalhoso, e os algoritmos de aprendizagem mais complexos e lentos (Lee & Teng 2000). Estas características também ocorrem com as redes neurofuzzy recorrentes. Assim, além de modelos e arquiteturas neurofuzzy recorrentes, se faz necessário desenvolver mecanismos de treinamento rápidos e eficazes.

1.1 Motivação

Redes neurais e sistemas fuzzy são abordagens complementares entre si. Sistemas neurofuzzy resultam da combinação destas duas abordagens, para superar os pontos fracos e desvantagens individuais e oferecer um desempenho superior.

O objetivo principal de combinar sistemas fuzzy e redes neurais é desenvolver arquiteturas que utilizem sistemas fuzzy para representar e processar conhecimento de forma clara e de fácil interpretação, e que aproveitem a capacidade de aprendizado das redes neurais. Obtém-se assim, modelos mais transparentes, com capacidade de adaptação e inclusão de conhecimento, caso este esteja disponível.

Embora os sistemas neurofuzzy sejam estudados com grande interesse, a ênfase na consideração de recorrência em arquiteturas neurofuzzy é menor. A principal diferença entre os modelos neurofuzzy estáticos e os modelos neurofuzzy recorrentes decorre no fato de que estes últimos são dotados de memória e, portanto, capazes de tratar informação temporal. Portanto, os modelos recorrentes são tão ou mais eficientes na resolução e análise de sistemas dinâmicos do que os modelos estáticos.

Assim, o desenvolvimento de arquiteturas de sistemas neurofuzzy recorrentes e de métodos de treinamento rápidos e eficientes, mostra-se como uma opção necessária para um tratamento mais efetivo de sistemas dinâmicos.

1.2 Objetivos

O objetivo deste trabalho é o de desenvolver sistemas neurofuzzy estáticos e recorrentes e aplicá-los na resolução de problemas de identificação e controle de sistemas dinâmicos não lineares.

Duas arquiteturas de redes neurofuzzy recorrentes e uma rede neurofuzzy estática são propostas neste trabalho. A incorporação de recorrência nos sistemas neurofuzzy fornecem elementos de memória que expandem a capacidade dos sistemas neurofuzzy estáticos de tratar relações temporais.

Para o treinamento das redes neurofuzzy propostas, desenvolve-se um método de treinamento para estas redes; este método é baseado no método do gradiente e no método de treinamento por reforço associativo.

As redes neurofuzzy propostas, junto com outros modelos neurais dinâmicos, são implementadas e aplicadas a problemas de identificação e controle de sistemas dinâmicos não lineares. Um estudo comparativo também é feito, considerando quatro exemplos de processos dinâmicos não lineares.

1.3 Organização da Tese

Este trabalho é dividido em seis capítulos. Este capítulo apresenta uma visão geral do escopo da tese, assim como a motivação para o desenvolvimento do trabalho e seus objetivos.

O Capítulo 2 apresenta uma revisão dos principais conceitos da teoria de redes neurais clássica. Isto inclui um breve histórico da área, definições básicas, classificação de arquiteturas, métodos de treinamento e algumas considerações adicionais, sendo apresentado em detalhe o método do gradiente para o treinamento de algumas das estruturas implementadas.

O Capítulo 3 apresenta os principais conceitos relacionados a teoria de conjuntos fuzzy: definições, sistemas de inferência estáticos e recorrentes, além de um breve comentário referente aos métodos de treinamento de sistemas fuzzy, capacidade de aprendizado herdada das redes neurais. Desta forma, se abre um espaço para o capítulo seguinte, que trata particularmente dos sistemas neurofuzzy.

O Capítulo 4 apresenta, inicialmente, uma breve comparação entre os sistemas neurais e os sistemas neurofuzzy para melhor compreender a origem dos sistemas neurofuzzy. Também se faz uma descrição dos neurônios lógicos fuzzy, estáticos e recorrentes a serem utilizados neste trabalho. Além disso, descreve-se, de forma detalhada, duas redes neurofuzzy recorrentes e uma rede neurofuzzy estática com os métodos de treinamento correspondentes. Estas se baseiam no método do gradiente e no método de treinamento por reforço associativo.

O Capítulo 5 discute as abordagens utilizadas, para identificação e o controle de sistemas

dinâmicos não lineares. Resultados de simulação são apresentados, incluindo comparações entre os sistemas neurofuzzy propostos e outros sistemas neurais e neurofuzzy propostos na literatura.

Finalmente, o Capítulo 6, apresenta as conclusões do trabalho e sugestões para pesquisas futuras.

Capítulo 2

Redes Neurais Artificiais

2.1 Introdução

As redes neurais artificiais, também denominadas como sistemas conexionistas ou de processamento distribuído paralelo, são paradigmas computacionais de processamento de informação inspirados no sistema nervoso biológico.

Durante os últimos anos, o campo de aplicação de modelos neurais tem-se expandido notavelmente, tanto em engenharia (Rafiq et al. 2001), como em ciências biológicas (Cai et al. 2000), (Cai & Zhou 2000), economia (Timmerman 1997), (Peat 1996), computação (Féraud & Clérot 2002), medicina (Baxt 1991), (Baxt 1990), sobressaindo em aplicações como classificação de padrões, aproximação de funções e previsão de séries temporais (Ballini 2000), entre outras.

Este capítulo apresenta uma revisão dos principais conceitos sobre modelos de redes neurais que serão utilizados no decorrer deste trabalho.

A seção 2.2 apresenta um histórico resumido da área e a seção 2.3 as principais definições da teoria de redes neurais.

Embora exista uma grande quantidade de arquiteturas de redes neurais, sem dúvida a estrutura multicamada é a mais conhecida e utilizada (Hush & Horne 1993), devido à propriedade de aproximação universal e de generalização para uma ampla classe de problemas, utilizando um mesmo algoritmo de aprendizado (Iyoda 2000).

Esta estrutura e outras importantes são explicadas na seção 2.4; na seção 2.5 explica-se brevemente os principais métodos de treinamento: métodos supervisionados, não-supervisionados e por reforço associativo.

Na última seção é feita uma breve revisão dos detalhes a considerar durante o treinamento de redes neurais, tais como inicialização dos pesos, modos de treinamento e critérios de parada.

2.2 Um Breve Histórico

O interesse nos estudos de mecanismos e estruturas baseados no cérebro humano tem permitido nestes últimos anos, um grande desenvolvimento de modelos computacionais no plano biológico. A evolução das redes neurais artificiais passou por um processo iniciado por um período de grande atividade, seguido por anos de estagnação nas pesquisas para, logo a seguir, permitir um processo de ressurgimento do interesse científico como conseqüência do desenvolvimento de novas tecnologias e fundamentos teóricos (Von Zuben 1993). Em (Tatibana & Kaetsu 2002) e (Von Zuben 1993) pode-se encontrar um resumo do processo de evolução da área de redes neurais.

A primeira tentativa de construir um modelo conexionista foi na década de 40 por W. S. McCulloch e W. Pitts (McCulloch & Pitts 1943), cujo trabalho fazia uma analogia entre células vivas e o processamento eletrônico, simulando o comportamento do neurônio natural, onde o neurônio possuía apenas uma saída, dada por uma função (*threshold*) da soma ponderada das suas diversas entradas.

Poucos avanços foram feitos até 1949, quando Donald Hebb (Hebb 1949) publicou o livro intitulado "The Organization of Behavior", o qual propõe, pela primeira vez, uma lei de aprendizagem específica para as sinapses dos neurônios.

Anos depois, Frank Rosenblatt (Rosenblatt 1958), tendo como base os estudos de McCulloch e Pitts (McCulloch & Pitts 1943), mostrou em seu livro "Principles of Neurodynamics" o modelo dos *perceptrons*. Neste modelo, os neurônios eram organizados em camadas de entrada e saída, onde os pesos das conexões eram adaptados a fim de se atingir a eficiência sináptica. Em 1960, Widrow e Hoff (Widrow & Hoff 1960), propuseram o modelo *Adaline* (ADAptative

LINear combiner Element), baseando-se também na proposta de McCulloch.

Em 1969, Minsky e Papert (Minsky 1969), realizaram um estudo mais rigoroso do modelo do perceptron proposto por Rosenblatt, enfatizando as suas limitações e provocando uma paralisação de atividades na área.

Na década de 70 e inícios de 80, surgiram estudos voltados para modelos com memória associativa. (Von der Malsburg 1973) e (Grossberg 1976) desenvolveram idéias de aprendizado competitivo, enquanto (Kohonen 1982) propunha os mapas auto-organizáveis.

O interesse em redes neurais foi renovado quando John Hopfield (Hopfield 1982) publica um importante estudo baseado no princípio físico sobre o armazenamento de informação em configurações dinamicamente estáveis. Hopfield considerou para estas configurações, um conjunto de neurônios binários, dispostos de forma que suas saídas fossem realimentadas para as entradas, sendo este um dos primeiros modelos a introduzir dinâmica em redes neurais (Von Zuben 1993).

Em 1986, Rumelhart e McClelland (Rumelhart & McClelland 1986) publicam o livro "Parallel Distributed Processing", divulgando o método para ajuste de parâmetros para redes estáticas multicamadas denominado *algoritmo de retropropagação*, sendo este inicialmente proposto por Werbos (Werbos 1974). Este fato originou uma grande "explosão" na área, fazendo com que pesquisadores de diversos campos passassem a visualizar interessantes aplicações para as redes neurais artificiais.

Em 1987, ocorreu em São Francisco a primeira conferência de redes neurais em tempos modernos, "IEEE International Conference on Neural Networks", sendo formada a "International Neural Networks Society" (INNS). A partir destes acontecimentos decorreram a publicação do periódico "Neural Networks" da INNS em 1989, seguido do "Neural Computation" e do "IEEE Transactions on Neural Networks", isto em 1990.

2.3 Definições Básicas

2.3.1 O Neurônio Artificial Clássico

O neurônio artificial clássico é baseado no neurônio biológico, o qual é formado por um corpo celular que contém o núcleo da célula; diversos dendritos, através dos quais impulsos elétricos são recebidos (receptor), e um axônio, pelo qual impulsos elétricos são enviados (transmissor). O neurônio recebe os sinais de entrada através dos dendritos, processa-os no corpo celular, e transmite o resultado do processamento através do axônio e ramificações. As conexões entre neurônios são efetuadas pelas sinapses, as quais são pontos de contato entre dendritos e axônios controlados por impulsos elétricos e reações químicas.

O neurônio artificial proposto por Mcculloch e Pitts tinha grandes limitações. Entre elas destaca-se a forma binária. O funcionamento deste modelo pode ser descrito intuitivamente da seguinte maneira: se a soma ponderada dos sinais de entrada de um neurônio ultrapassar um determinado limiar a, então a saída $y = f(u)$ recebe valor um; se não, recebe valor zero. As entradas x_j do neurônio também são binárias. A Figura 2.1 ilustra o modelo deste neurônio.

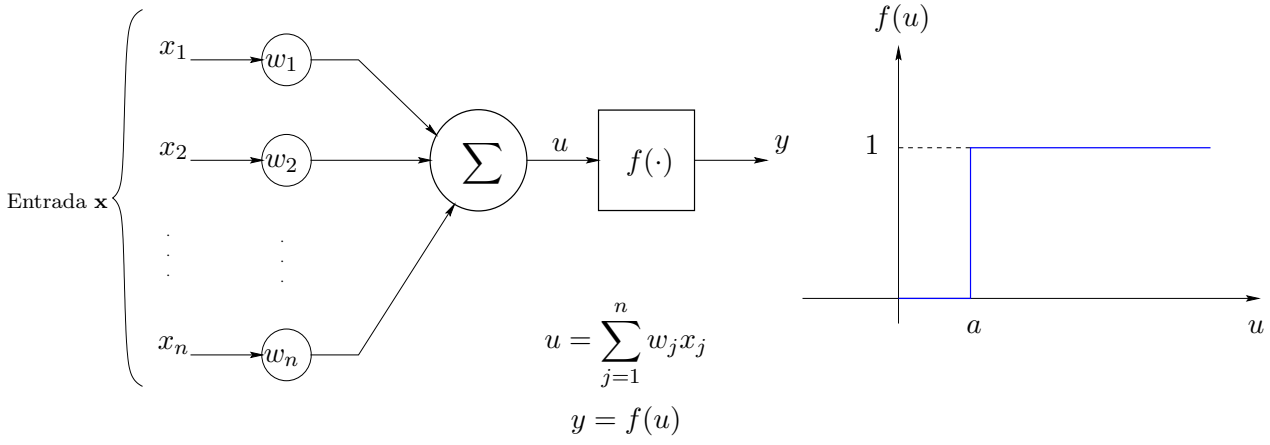


Figura 2.1: Neurônio binário proposto por McCulloch e Pitts.

O modelo atual do neurônio artificial considera um vetor de entrada $\mathbf{x}$ $n-$dimensional não necessariamente binário, um operador de agregação genérico, uma saída y, pesos sinápticos w_i e uma função de ativação $f(\cdot)$, podendo esta última adotar diversas formas.

Cada componente x_i $(i = 1, \ldots, n)$ do vetor de entrada está ligado ao neurônio através de

conexões que fazem o papel dos dendritos, cujas intensidades são representadas pelos pesos sinápticos w_i. Usualmente o neurônio efetua uma soma ponderada entre os componentes do vetor de entrada e o vetor peso, obtendo assim a agregação das entradas e a correspondente saída y.

Uma entrada fixa denominada de polarização, pode ser introduzida de modo que a ativação e a correspondente saída do neurônio é representado por:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + x_0 w_0\right) \tag{2.1}$$

onde $f(\cdot)$ é a função de ativação, w_0 é o peso sináptico correspondente à entrada de polarização x_0. Tipicamente faze-se $x_0 = +1$ ou $x_0 = -1$.

2.3.2 Funções de Ativação

A função de ativação $f(\cdot)$ determina a saída do neurônio em termos do valor u da agregação das entradas. Segundo (Haykin 1994), podem ser definidos quatro tipos principais de funções de ativação:

1. ***Função Sinal***: utilizada no modelo original de Rosenblatt (Rosenblatt 1958) e ilustrada na Figura 2.2.(a), é definida da seguinte maneira:

$$f(u) = \begin{cases} 1 & \text{se } u > 0, \\ 0 & \text{se } u \leq 0. \end{cases} \tag{2.2}$$

2. ***Função Rampa***: define-se como:

$$f(u) = \begin{cases} 0 & \text{se } u \leq \alpha, \\ \dfrac{u-\alpha}{\beta-\alpha} & \text{se } \alpha < u \leq \beta, \\ 1 & \text{se } u > \beta. \end{cases} \tag{2.3}$$

   ilustrada na Figura 2.2.(b) para $\alpha = -0.5$ e $\beta = 0.5$.

3. ***Função Sigmoidal***: esta função é também denominada função logística:

$$f(u) = \frac{1}{1 + \exp^{-\xi u}} \tag{2.4}$$

onde, ξ é o parâmetro que determina o ponto de inflexão da função sigmoidal. A Figura 2.2.(c) ilustra um caso particular para $\xi = 2$.

4. ***Função Tangente Hiperbólica***: é utilizada quando se requer de valores tanto positivos como negativos da saída do neurônio:

$$f(u) = \tanh(\xi u) = \frac{\exp^{\xi u} - \exp^{-\xi u}}{\exp^{\xi u} + \exp^{-\xi u}} \tag{2.5}$$

A Figura 2.2.(d) ilustra função para $\xi = 1$, onde ξ é o parâmetro que determina o ponto de inflexão da função.

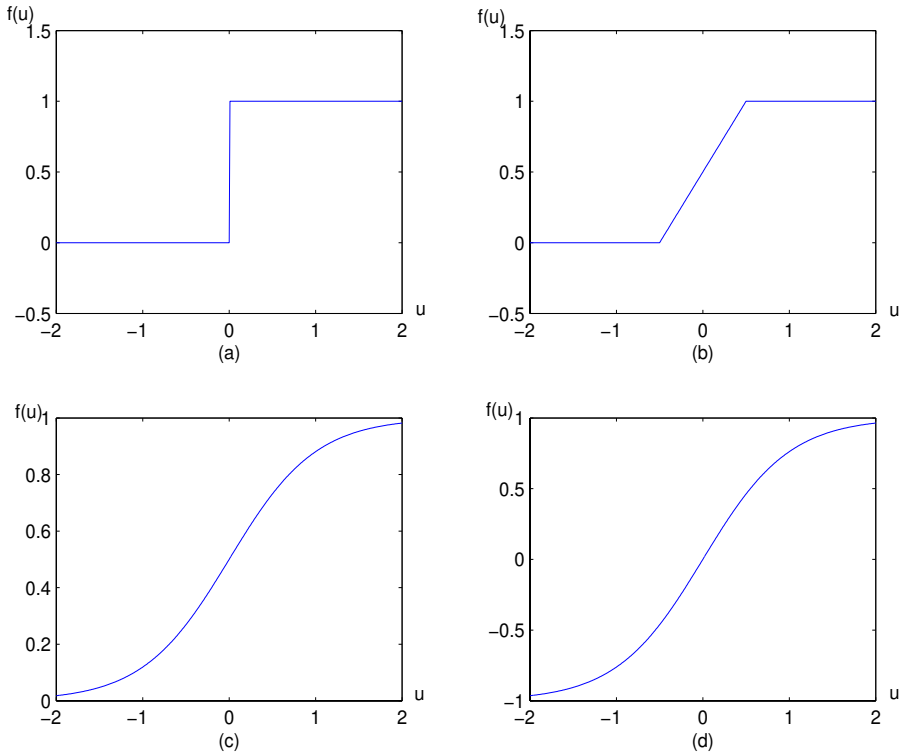


Figura 2.2: Funções de ativação. (a): função sinal, (b): função rampa, (c): função sigmoidal, (d): função tangente hiperbólica.

2.4 Classificação de Redes Neurais

Do ponto de vista funcional, uma rede neural é homogênea se todos os modelos dos neurônios que a compõe são idênticos, ou, caso contrário, heterogêneas. As redes neurais artificiais são normalmente homogêneas, diferentemente das redes neurais biológicas que são bastante heterogêneas.

Por outro lado, do ponto de vista estrutural, as redes neurais artificiais podem ser classificadas dentro de dois grandes grupos: redes neurais estáticas ou não recorrentes, e redes neurais dinâmicas ou recorrentes.

2.4.1 Redes Neurais Estáticas

As redes neurais estáticas são aquelas cujas saídas dependem somente do valor atual das entradas e não de valores passados nem futuros. A estrutura estática mais utilizada na literatura é sem dúvida a Rede Multicamadas com função de ativação sigmoidal (MLP), isto devido a sua capacidade de aproximação universal e processamento paralelo (Scarselli & Tsoi 1998). Outro modelo estático importante é denominado Rede Neural com Funções de Ativação de Base Radial (RBF). Estas duas arquiteturas serão detalhadas a seguir.

Redes Multicamadas (MLP)

Esta arquitetura consta de uma camada de entrada, uma ou várias camadas intermediárias e uma camada de saída. Os neurônios da camada de entrada, são os responsáveis pela transmissão do sinal de entrada para a camada intermediária e, geralmente, possuem uma função de ativação linear. As camadas intermediárias transmitem informações entre a camada de entrada e a camada de saída, sendo que as funções de ativação dos neurônios pertencentes a esta camada são tipicamente funções de ativação não decrescentes e diferenciáveis, em geral sigmóides. As saídas dos neurônios intermediários são processadas pelos neurônios da camada de saída, fornecendo sinais de saída correspondentes.

A Figura 2.3 mostra a arquitetura de uma rede MLP com uma camada intermediária e entradas de polarização, onde $\mathbf{x} = [x_0,\ x_1,\ x_2,\ \ldots,\ x_n]'$; $\mathbf{y} = [y_1,\ y_2,\ \ldots,\ y_p]'$; w_{ji} e v_{kj} são os

pesos da camada intermediária e de saída, respectivamente, para $i = 0, \ldots, n$; $j = 0, \ldots, M$; $k = 1, \ldots, p$; sendo M o número de neurônios da camada intermediária.

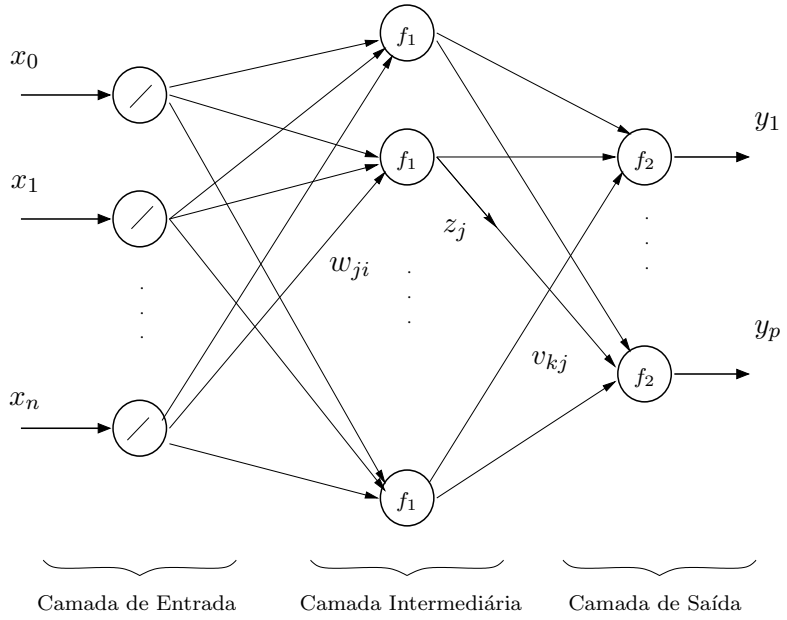


Figura 2.3: Rede MLP com uma camada intermediária.

De forma análoga aos neurônios biológicos, as conexões entre os neurônios das distintas camadas da rede, são denominadas de sinápses. Cada sinápse está associada a um peso que representa a intensidade da conexão. O fluxo de informação está sempre direcionado em um sentido entrada-saída (*feedforward*), ou seja, não existe realimentação.

Considerando entrada de polarização na rede MLP e funções de ativação lineares na camada de entrada, define-se a saída z_j do $j-$ésimo neurônio da camada intermediária como:

$$z_j = f_1 \left(\sum_{i=0}^{n} w_{ji} x_i \right) \tag{2.6}$$

onde $x_0 = +1$ e f_1 é a função de ativação na camada intermediária. A $k-$ésima saída da rede y_k é calculada como:

$$y_k = f_2 \left(\sum_{j=0}^{M} v_{kj} z_j \right) \tag{2.7}$$

sendo $z_0 = +1$ e f_2 a função de ativação dos neurônios da camada de saída da rede.

Assim, existem três elementos a considerar para definir uma rede MLP (Ballini 2000):

1. O número de camadas intermediárias e o número de neurônios em cada camada;
2. As funções de ativação;
3. Os tipos de conexões determinadas pelas sinápses.

Redes Neurais com Funções de Ativação de Base Radial (RBF)

As redes neurais com funções de ativação de base radial (RBF) são modelos com uma camada intermediária, cujos neurônios na camada de saída formam uma combinação linear das funções de base computadas pelos neurônios da camada intermediária. As funções de ativação dos neurônios da camada intermediária são as funções de base (Scarselli & Tsoi 1998), sendo a mais difundida e utilizada a função Gaussiana.

Deste modo, utilizando funções de base Gaussianas, cada neurônio da camada intermediária produz uma saída idêntica para entradas que se encontram a uma mesma distância do centro da função base correspondente ao neurônio (Hush & Horne 1993). É por este motivo, que estas funções são denominadas de base radial. A Figura 2.4 mostra uma rede RBF.

Assim, a saída z_j do neurônio j da camada intermediária com funções de base Gaussianas, é definida como:

$$z_j = \exp\left(-\frac{\sum_{i=1}^{n}(x_i - w_{ji})^2}{2\sigma_j^2} \right) \tag{2.8}$$

para $j = 1, \ldots, M$, sendo M o número de neurônios da camada intermediária, n o número de entradas da rede, w_{ji} os pesos da camada de entrada que, ao mesmo tempo, fazem o papel de centros das funções de ativação Gaussianas na camada intermediária, e σ_j são os parâmetros de dispersão.

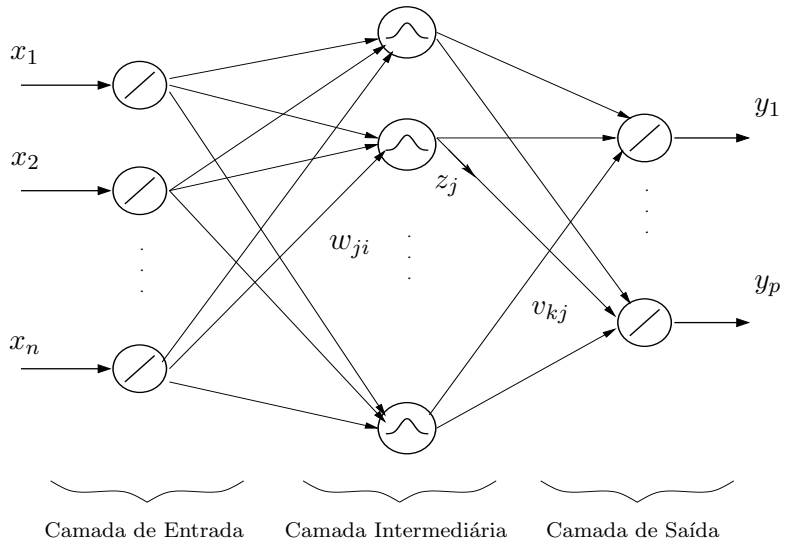


Figura 2.4: Rede com função de ativação de base radial.

Finalmente, a saída y_k desta rede é dada por:

$$y_k = \sum_{j=1}^{M} v_{kj} z_j \tag{2.9}$$

onde v_{kj} são os pesos da camada de saída, com $k = 1, \ldots, p$, sendo p o número de saídas da rede. Deste modo, a rede RBF executa um mapeamento não linear $\Re^n \rightarrow \Re^p$ através de uma combinação linear de funções de base não lineares.

As redes MLP utilizam a função base sigmoidal (equação (2.4)) como funções de ativação. A principal diferença entre uma rede MLP com funções de ativação sigmoidais e uma rede RBF com funções de ativação Gaussianas é a natureza das próprias funções de ativação. Enquanto os neurônios intermediários de uma rede MLP com funções de ativação sigmoidais cobre uma ampla região do espaço de entrada, os neurônios intermediários da rede RBF cobrem pequenas regiões específicas. Desta forma, as redes MLP são mais eficientes que as redes RBF para certas aplicações, como é o caso no problema de aproximação de funções. Já para a resolução de problemas de classificação de padrões, as redes RBF resultam ser mais eficientes que as redes MLP (Hush & Horne 1993).

2.4.2 Redes Neurais Recorrentes

As redes neurais recorrentes são estruturas de processamento capazes de representar uma grande variedade de comportamentos dinâmicos. A presença de realimentação de informação permite a criação de representações internas e dispositivos de memória capazes de processar e armazenar informações temporais e sinais seqüenciais (de Moraes Lima 2000).

Arquiteturas Básicas

A primeira tentativa de incorporar recorrência em redes neurais foi proposta por Hopfield (Hopfield 1982). As redes de Hopfield utilizam como unidades processadoras o modelo do neurônio baseado na proposta de McCulloch e Pitts, ou seja, neurônios com dois únicos estados "on" (1) e "off" (-1 ou 0).

Estes tipos de estruturas de redes neurais são mais apropriadas para aplicações nas quais uma representação binária dos padrões é possível, como é o caso do tratamento de imagens em branco e preto (Lau 1991). A Figura 2.5 ilustra a estrutura desta rede.

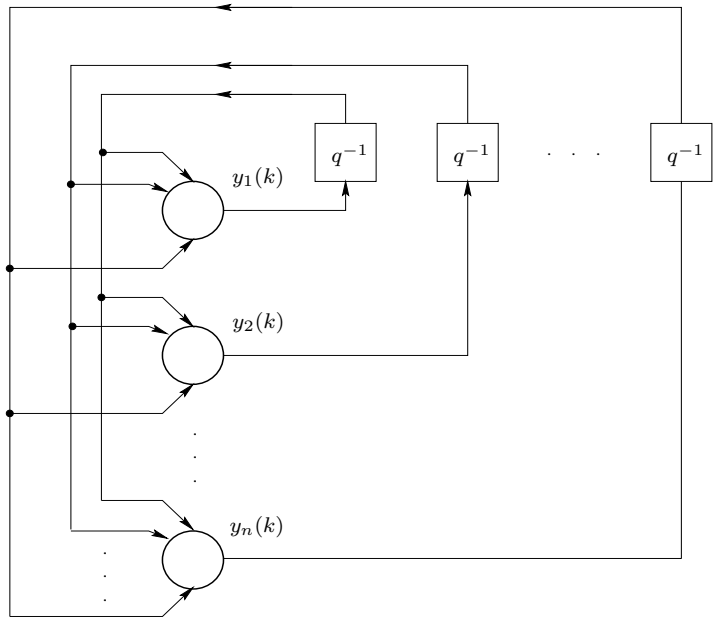


Figura 2.5: Rede de Hopfield, totalmente recorrente.

A rede de Hopfield pode ser vista como uma memória associativa não linear, cuja principal

função é recuperar um padrão como resposta da apresentação de uma versão incompleta ou ruidosa do próprio padrão. Cada padrão memorizado está associado a um *atrator*, pois a saída será igual ao estado correspondente ao *atrator* mais próximo ao padrão de entrada fornecido (Haykin 1994). Devido a este fato, estes tipos de estruturas são menos aplicadas para a resolução de problemas em tempo contínuo, pois a informação contínua da representação do problema deve ser transformada em binário para efetuar o armazenamento desta.

Como o enfoque neste trabalho é o aprendizado de trajetórias e não no projeto de atratores, o estudo de redes do tipo Hopfield em particular, não será de interesse.

As redes neurais recorrentes podem ser classificadas como parcialmente recorrentes ou totalmente recorrentes, dependendo do tipo de recorrência. Redes totalmente recorrentes possuem todas as conexões recorrentes e ajustáveis. Quando apenas uma parte das possíveis conexões recorrentes é admitida ou, então, quando as conexões recorrentes não são ajustáveis, resultam em redes parcialmente recorrentes. Um exemplo de rede totalmente recorrente é a rede de Hopfield (Figura 2.5) e um exemplo de rede parcialmente recorrente é a rede de Elman (Elman 1990), ilustrada na Figura 2.6.

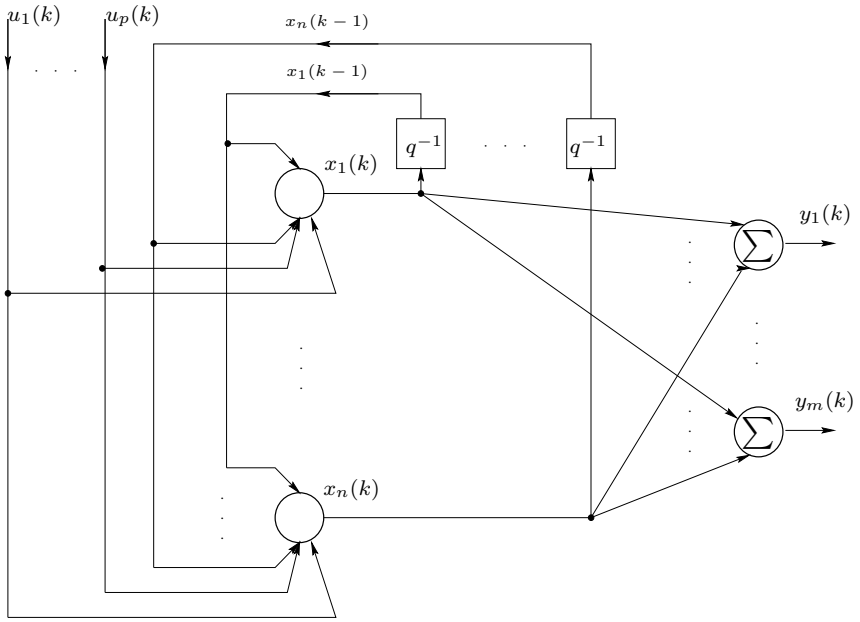


Figura 2.6: Rede de Elman, parcialmente recorrente.

Dentro da classificação das redes parcialmente recorrentes, têm-se também as redes com

recorrência interna local (Figura 2.7) e as redes com recorrência interna global, ilustrada na Figura 2.8.

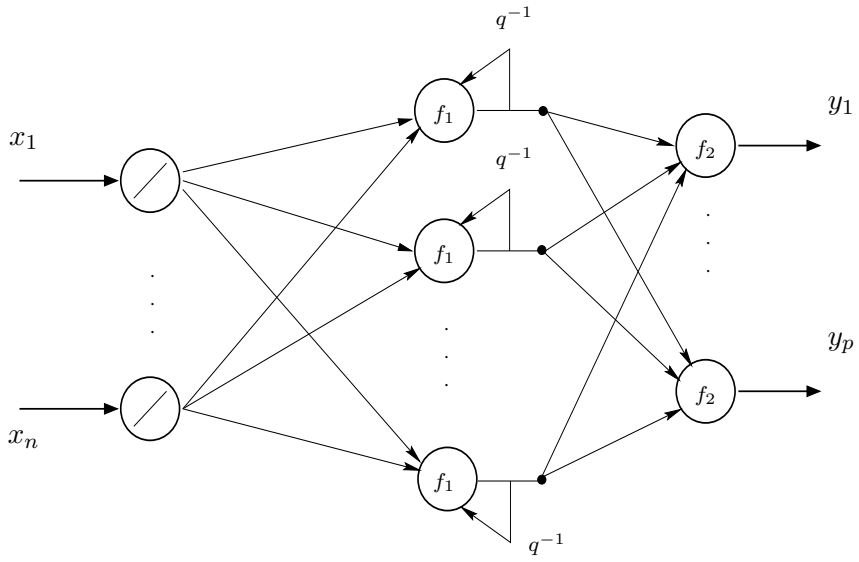


Figura 2.7: Rede neural com recorrência interna local.

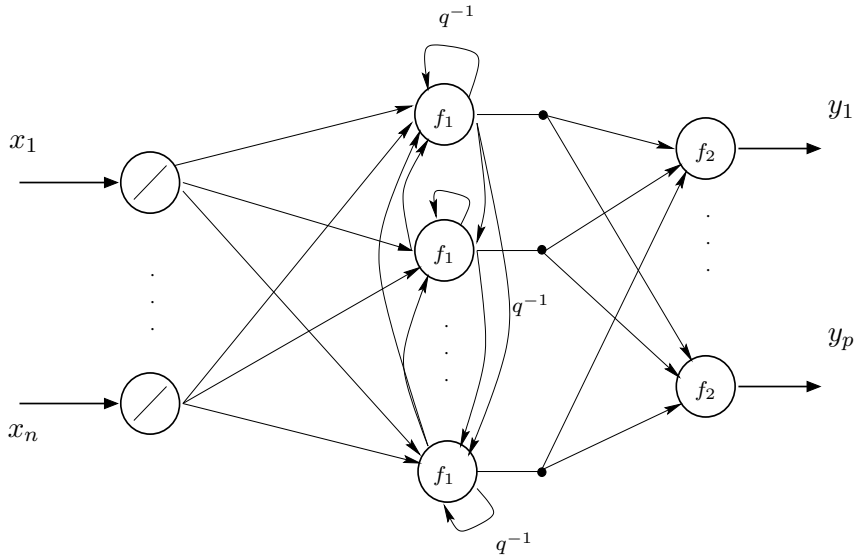


Figura 2.8: Rede neural com recorrência interna global.

Uma forma simples de incorporar recorrência em uma rede estática é gerar um laço de realimentação da saída, utilizando operadores atraso q^{-1} (Figura 2.9). Esta arquitetura particular foi proposta por M. Jordan (Jordan 1986) e por K. S. Narendra (Narendra &

Parthasarathy 1990), em aplicações de modelagem e controle de sistemas dinâmicos.

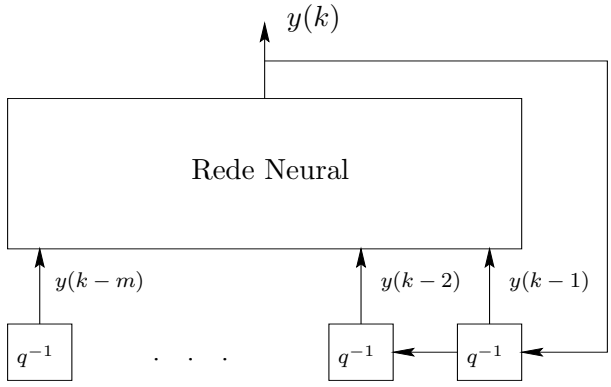


Figura 2.9: Rede neural com recorrência na saída.

Em (Nerrand et al. 1993), é demonstrado que toda rede recorrente pode ser representada em uma forma canônica utilizando uma rede equivalente estática com as seguintes características:

- As saídas são as saídas dos neurônios contendo as saídas desejadas, e os valores das variáveis de estado;
- As entradas são as entradas da rede e valores anteriores das variáveis de estado.

A forma canônica de uma rede neural é, deste modo, expressa da seguinte maneira:

$$\begin{cases} \mathbf{z}(k) = \varphi_1[\mathbf{y}(k-1), \mathbf{u}(k-1)]; \\ \mathbf{y}(k) = \varphi_2[\mathbf{z}(k-1), \mathbf{u}(k-1)] \end{cases}$$

onde $\mathbf{z}(k)$ é o vetor de estados de tamanho M, sendo M a ordem da rede, $\mathbf{y}(k)$ o vetor de saída da rede, e $\mathbf{u}(k)$ o vetor que contém as entradas que não pertencem à realimentação. Este conceito pode ser utilizado com qualquer tipo de neurônio em tempo discreto.

Utilizando a forma canônica das redes neurais, pode-se estabelecer uma comparação entre redes neurais e filtros adaptativos. As redes estáticas são utilizadas para representar filtros transversais e as redes recorrentes são utilizadas para representar filtros recursivos. Em (Nerrand et al. 1993) é também demonstrado que diversos métodos aplicados na teoria clássica

de filtragem adaptativa, assim como alguns dos métodos de treinamento de redes neurais propostos na literatura, são casos especiais de uma classificação geral de métodos de treinamento para redes neurais estáticas.

2.5 Métodos de Treinamento

Define-se aprendizado ou treinamento de redes neurais como o processo pelo qual os parâmetros livres ou adaptáveis da rede são ajustados via um processo contínuo de simulação do ambiente no qual a rede é inserida (Haykin 1994). Este processo pode ser classificado dentro de três grandes grupos:

1. Treinamento supervisionado.
2. Treinamento não supervisionado.
3. Treinamento por reforço.

2.5.1 Treinamento Supervisionado

O treinamento supervisionado caracteriza-se pela disponibilidade de conhecimento ou informação sobre um sistema na forma de padrões entrada-saída (Haykin 1994). Estes padrões formam o conjunto de treinamento; para uma dada entrada a saída da rede neural é comparada com a saída correspondente do conjunto de treinamento. Os parâmetros da rede são ajustados tendo como base o sinal de erro e os padrões de treinamento. O sinal de erro é definido como a diferença para uma mesma entrada entre a saída da rede e a saída desejada do conjunto de treinamento. Estes ajustes são feitos de forma iterativa que prossegue até que o erro seja pequeno o suficiente. O mecanismo de treinamento supervisionado é ilustrado na Figura 2.10.

Aplicando um algoritmo de treinamento supervisionado, uma rede neural adquire conhecimento ou informação relevante sobre um problema a ser resolvido de forma análoga àquela utilizada pelo ser humano e outros animais, ou seja, a partir de exemplos (Von Zuben 1993).

Uma desvantagem deste tipo de treinamento consiste no fato de que a rede neural não pode aprender novas estratégias para situações particulares que não foram consideradas na escolha do conjunto de treinamento.

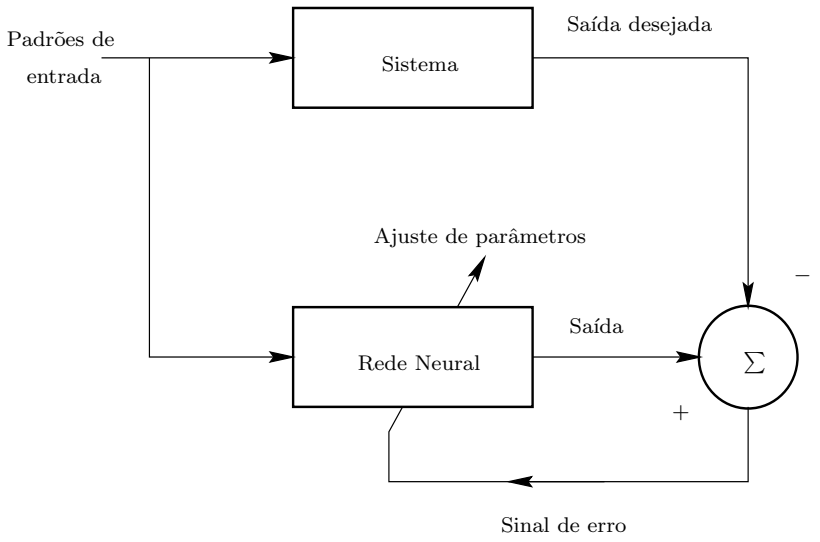


Figura 2.10: Treinamento supervisionado.

Em (Nerrand et al. 1994) treinamento não adaptativo é definido como sendo o processo pelo qual a rede neural é treinada, inicialmente, utilizando um conjunto finito de padrões entrada-saída, e logo utilizada mantendo os parâmetros fixos. O treinamento adaptativo refere-se ao processo de treinamento pelo qual a rede é treinada de forma permanente enquanto está sendo utilizada (conjunto infinito de padrões de treinamento).

No decorrer deste trabalho, denomina-se como treinamento *off-line* o treinamento supervisionado não adaptativo e treinamento supervisionado adaptativo o treinamento *on-line.*

Um algoritmo de treinamento supervisionado altamente popular é baseado o método do gradiente, comumente denominado algoritmo de retropropagação, proposto pela primeira vez por P. J. Werbos (Werbos 1974) e redescoberto e popularizado por Rumelhart e McClelland (Rumelhart & McClelland 1986). O nome de retropropagação deve-se ao fato do sinal de erro ser *retropropagado* camada por camada através da rede. O método supervisionado de retropropagação é um dos mais utilizados na literatura (Warner & Misra 1996). Este algoritmo é detalhado a seguir.

Algoritmo de Retropropagação

O algoritmo de retropropagação consiste basicamente de duas etapas (Ballini 2000): durante a primeira etapa (*forward*), as entradas são apresentadas e propagadas camada por

camada mantendo os pesos fixos; a saída é comparada com a saída desejada, calculando assim, o sinal de erro. Na segunda etapa, o erro é retropropagado (*backward*), sendo os pesos atualizados via o Método do Gradiente Descendente (Haykin 1994). Seja:

- z_j^l a saída do $j-$ésimo neurônio na camada l;
- w_{ji}^l o peso entre o neurônio i da camada $l-1$ e o neurônio j da camada l;
- $\mathbf{x}_p$ o $p-$ésimo padrão de entrada;
- z_i^0 o $i-$ésimo componente do vetor de entrada;
- y_j a $j-$ésima saída desejada;
- M_l o número de neurônios da $l-$ésima camada;
- N o número de padrões de treinamento;
- L o número de camadas da rede neural;
- $f_l(\cdot)$ a função de ativação dos neurônios na camada l.

sendo $z_j^0 = x_j$, $z_0^l = 1$ (polarização) e w_{j0}^l os pesos correspondentes às polarizações. A saída do $j-$ésimo neurônio da camada l é dada por:

$$z_j^l = f_l \left[\sum_{i=0}^{M_{l-1}} w_{ji}^l z_i^{l-1} \right] \tag{2.10}$$

O objetivo é minimizar uma função objetivo definida como o Erro Quadrático Médio, isto é:

$$J(\mathbf{w}) = \frac{1}{N} \sum_{p=1}^{N} J_p(\mathbf{w}) \tag{2.11}$$

onde $J_p(\mathbf{w})$ é o erro quadrático total para o $p-$ésimo padrão, definido por:

$$J_p(\mathbf{w}) = \frac{1}{2} \sum_{q=1}^{M_L} \epsilon_q^2 \tag{2.12}$$

com

$$\epsilon_q = (z_q^L - y_q) \tag{2.13}$$

Para minimizar a função objetivo (2.11) via o método do gradiente descendente, é necessário calcular a derivada parcial de J_p em relação a cada peso da rede e para cada padrão. Assim, as regras para efetuar o ajuste dos pesos podem ser escritos da seguinte maneira:

$$w_{ji}^{l}(k+1) = w_{ji}^{l}(k) - \alpha \frac{\partial J(\mathbf{w})}{\partial w_{ji}^{l}}\bigg|_{\mathbf{w}(k)} \tag{2.14}$$

fazendo $J = J_p$ para $N = 1$, tem-se

$$w_{ji}^{l}(k+1) = w_{ji}^{l}(k) - \alpha \sum_{p=1}^{M_L} \frac{\partial J_p(\mathbf{w})}{\partial w_{ji}^{l}}\bigg|_{\mathbf{w}(k)} \tag{2.15}$$

sendo α a taxa de aprendizado. Utilizando a regra da cadeia têm-se:

$$\frac{\partial J_p(\mathbf{w})}{\partial w_{ji}^{l}} = \frac{\partial J_p(\mathbf{w})}{\partial z_{j}^{l}} \frac{\partial z_{j}^{l}}{\partial w_{ji}^{l}} \tag{2.16}$$

onde,

$$\frac{\partial z_{j}^{l}}{\partial w_{ji}^{l}} = \frac{\partial}{\partial w_{ji}^{l}} \left[f_l \left(\sum_{m=0}^{M_{l-1}} w_{jm}^{l} z_{m}^{l-1} \right) \right] \tag{2.17}$$

Seguindo a regra da cadeia,

$$\frac{\partial z_{j}^{l}}{\partial w_{ji}^{l}} = f_l' \left(\sum_{m=0}^{M_{l-1}} w_{ji}^{l} z_{m}^{l-1} \right) \frac{\partial}{\partial w_{ji}^{l}} \left[\sum_{m=0}^{M_{l-1}} w_{jm}^{l} z_{m}^{l-1} \right] \tag{2.18}$$

para logo obter,

$$\frac{\partial z_{j}^{l}}{\partial w_{ji}^{l}} = f_l' \left(\sum_{m=0}^{M_{l-1}} w_{jm}^{l} z_{m}^{l-1} \right) z_{i}^{l-1} \tag{2.19}$$

Assim,

$$\frac{\partial J_p(\mathbf{w})}{\partial w_{ji}^{l}} = \frac{\partial J_p(\mathbf{w})}{\partial z_{j}^{l}} f_l' \left(\sum_{m=0}^{M_{l-1}} w_{jm}^{l} z_{m}^{l-1} \right) z_{i}^{l-1} \tag{2.20}$$

onde o termo $\partial J_p(\mathbf{w})/\partial z_j^l$ representa a sensibilidade de $J_p(\mathbf{w})$ com relação à saída z_j^l (Hush & Horne 1993). Seja δ_j^l o gradiente local definido por:

$$\delta_j^l = \frac{\partial J_p(\mathbf{w})}{\partial z_j^l} f_l' \left(\sum_{m=0}^{M_{l-1}} w_{jm}^l z_m^{l-1} \right) \tag{2.21}$$

A equação (2.20) pode ser reescrita como:

$$\frac{\partial J_p(\mathbf{w})}{\partial w_{jm}^l} = \delta_j^l z_i^{l-1} \tag{2.22}$$

Assumindo uma rede MLP com uma camada intermediária ($L = 3$), a modo de exemplo, o erro ϵ_q é calculado como:

$$\epsilon_q(k) = z_q^3(k) - y_q(k) \tag{2.23}$$

para $q = 1, \ldots, M_3$. Considerando primeiro, o caso em que o neurônio pertence a camada de saída (ver Figura 2.11) e utilizando a equação (2.12), tem-se que o gradiente local é dado por:

$$\delta_q^3(k) = \epsilon_q(k) f_3' \left(\sum_{m=0}^{M_2} w_{qm}^3 z_m^2 \right) \tag{2.24}$$

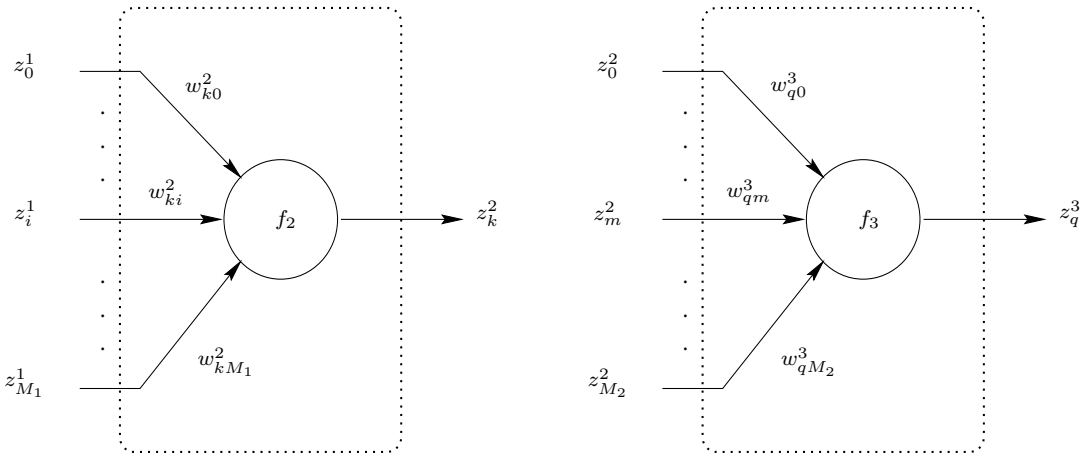


Figura 2.11: Neurônios de uma rede MLP para L=3: do lado esquerdo, neurônio de uma camada intermédia; do lado direito, neurônio da camada de saída

para $q = 1, \ldots, M_3$. Deste modo, a atualização dos pesos da camada de saída pode ser escrito como:

$$\Delta w_{qi}^3(k) = \frac{\partial J_p(\mathbf{w})}{\partial w_{qi}^3(k)}\bigg|_{\mathbf{w}(k)} = \epsilon_q(k) f_3' \left(\sum_{m=0}^{M_2} w_{qm}^3 z_m^2 \right) z_i^2 \tag{2.25}$$

para $i = 0, \ldots, M_2$ e $q = 1, \ldots, M_3$. Considerando agora os neurônios da camada intermediária, δ_j^2 é calculado como:

$$\delta_j^2(k) = \frac{\partial J_p(\mathbf{w})}{\partial z_q^3} \frac{\partial z_q^3}{\partial z_j^2} f_2' \left(\sum_{m=0}^{M_1} w_{jm}^2 z_m^1 \right) \tag{2.26}$$

Calculando as derivadas,

$$\frac{\partial J_p(\mathbf{w})}{\partial z_q^3} = \frac{1}{2} \sum_{q=1}^{M_3} \frac{\partial \epsilon_q^2(k)}{\partial z_q^3} = \sum_{q=1}^{M_3} \epsilon_q(k) \tag{2.27}$$

$$\frac{\partial z_q^3}{\partial z_j^2} = \frac{\partial}{\partial z_j^2} \left[f_3 \left(\sum_{j=0}^{M_2} w_{qj}^3 z_j^2 \right) \right] = f_3' \left(\sum_{j=0}^{M_2} w_{qj}^3 z_j^2 \right) w_{qj}^3 \tag{2.28}$$

Substituindo as equações (2.27) e (2.28) na equação (2.26), tem-se:

$$\delta_j^2(k) = \sum_{q=1}^{M_3} e_q(k) f_3' \left(\sum_{j=0}^{M_2} w_{qj}^3 z_j^2 \right) w_{qj}^3 f_2' \left(\sum_{m=0}^{M_1} w_{jm}^2 z_m^1 \right) \tag{2.29}$$

Finalmente, substituindo a equação (2.24) na equação (2.29), obtém-se:

$$\delta_j^2(k) = \left[\sum_{q=1}^{M_3} \delta_q^3(k) w_{qj}^3 \right] f_2' \left(\sum_{m=0}^{M_1} w_{jm}^2 z_m^1 \right) \tag{2.30}$$

Desta forma, a atualização dos pesos da camada intermediária utilizando as equações (2.30) e (2.20) é da seguinte forma:

$$\Delta w_{ji}^2 = \frac{\partial J_p(\mathbf{w})}{\partial w_{ji}^2}\bigg|_{\mathbf{w}(k)} = \left[\sum_{q=1}^{M_3} \delta_q^3(k) w_{qj}^3\right] f_2' \left(\sum_{m=0}^{M_1} w_{jm}^2 z_m^1\right) z_i^1 \tag{2.31}$$

para $i = 0, \ldots, M_1$ e $j = 0, \ldots, M_2$, sendo, neste caso, $z_i^1 = z_i^0 = x_i$.

Diversas modificações deste algoritmo de treinamento podem ser encontradas na literatura. No caso das redes recorrentes, o algoritmo de retropropagação pode ser utilizado, feitas certas considerações.

Em (Von Zuben 1993) detalha-se o método do gradiente para redes neurais recorrentes. Outros algoritmos de treinamento de redes recorrentes são detalhados em (Atiya & Parlos 2000), (Santos & Von Zuben 1999), (Campolucci et al. 1999), (Pearlmutter 1995) e (Olurotimi 1994).

2.5.2 Treinamento Não Supervisionado

No treinamento não supervisionado ou auto-organizado, não existe um elemento externo que inspecione o processo, ou seja, não existe um conjunto específico de padrões a serem aprendidos pela rede neural, mas sim, uma medida da qualidade de representação que a rede requer para o aprendizado (Figura 2.12). Os parâmetros da rede neural são ajustados em função desta medida. Uma vez que a rede consegue extrair algumas características dos dados de entrada, esta será capaz de gerar representações internas para classificar as características. Métodos de treinamento não supervisionados e aplicações são propostos em (Kohonen et al. 2000) e (Kohonen 1982).

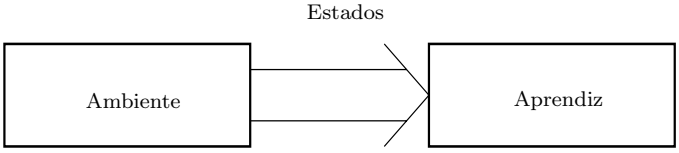


Figura 2.12: Treinamento não supervisionado.

Os métodos não supervisionados são também utilizados numa primeira etapa de treinamento das redes RBF. Como outro exemplo de métodos de treinamento não supervisionado tem-se os distintos métodos de clusterização, que neste caso, são utilizados para definir os centros das funções de ativação sigmoidais dos neurônios da camada intermediária de uma rede RBF (Hush & Horne 1993). Diversos algoritmos de clusterização encontram-se detalhados em (Lin & Lee 1996).

2.5.3 Treinamento por Reforço

O treinamento por reforço é um problema discreto e dinâmico (Jouffe 1998), no qual um sistema de aprendizagem ou agente, deve aprender um comportamento desejado através de tentativas, admitindo-se acertos e erros durante a interação com um determinado ambiente (Onat et al. 1998), (Kaelbing et al. 1996).

O treinamento por reforço é um método de treinamento do tipo "avaliador" e não do tipo "instrutor", como no caso dos métodos supervisionados. No treinamento por reforço, a informação disponível para o treinamento geralmente é pobre e grossa, o sinal de avaliação é um escalar e no caso extremo, existe uma única informação para indicar se a saída da rede está certa ou errada (Lin & Lee 1996).

Segundo (Haykin 1994), o aprendizado por reforço pode ser não associativo ou associativo. No treinamento por reforço não associativo, o sistema de aprendizagem escolhe uma determinada ação ao invés de associar diferentes ações com diferentes estímulos, ou seja, o sinal de reforço é a única entrada que o sistema recebe do ambiente.

No treinamento por reforço associativo, um mapeamento ou associação ações-estímulos é produzido, de forma a maximizar um valor que caracterize o desempenho do sistema de aprendizagem ou o sinal de reforço. Dado que o aprendiz não sabe a *priori* a ação que deve ser tomada, este deve descobrir quais são as ações que produzem uma maior recompensa.

Durante o desenvolvimento deste trabalho esta abordagem será utilizada para a atualização dos pesos no modelo de rede neural proposto. Desta forma, a partir de agora, se entenderá como treinamento por reforço ao treinamento por reforço associativo.

No modelo básico de aprendizado ou treinamento por reforço, o sistema de aprendizagem comunica-se com o ambiente via ações (saídas do sistema) e percepções (entradas ao sistema e o sinal de reforço), como mostra a Figura 2.13.

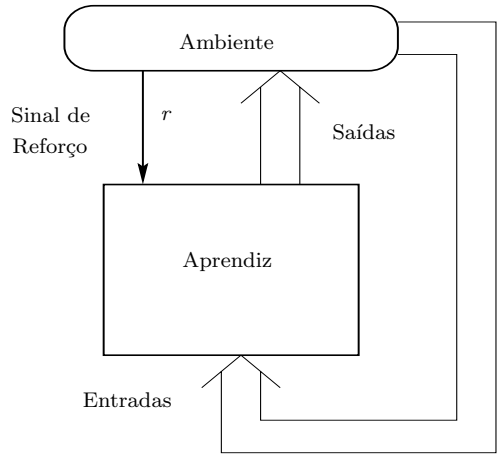


Figura 2.13: Modelo básico de treinamento por reforço.

Em cada iteração, o sistema de aprendizagem recebe como entrada algum sinal indicando o estado atual do ambiente. A seguir, o sistema escolhe a ação a ser tomada, gerando assim a saída a ser avaliada. Esta ação gera uma mudança no estado do ambiente e a transição de estado é comunicada ao aprendiz via um sinal de reforço r. Através de um processo iterativo de recompensa e punição, o sistema escolhe a ação que acrescente o valor do sinal de reforço com o passar do tempo.

Existem várias diferenças entre o treinamento supervisionado e o treinamento por reforço, sendo que a mais importante é a ausência de padrões de treinamento. Assim que o aprendiz escolhe a ação ser tomada, este é informado do sinal de reforço (recompensa ou punição) e do estado atual do ambiente e não é informado em momento algum sobre a ação ótima para os respectivos interesses; é necessário para o aprendiz reunir experiência sobre o ambiente, estados, ações ou punições para, a partir de um certo momento, tomar a decisão correta (Kaelbing et al. 1996).

2.6 Algumas Considerações para o Treinamento

2.6.1 Taxa de Aprendizado

O algoritmo de retropropagação fornece uma aproximação da trajetória no espaço dos pesos em direção ao mínimo da função objetivo (expressão (2.11)) computada via o método do gradiente descendente.

A taxa de aprendizagem α determina o passo na variação dos pesos sinápticos. Assim, quanto menor é α, menor será a variação dos pesos de uma iteração para a próxima e, no caso do algoritmo de retropropagação, a trajetória no espaço dos pesos será mais suave. Por outro lado, se α toma valores muito grandes, embora a velocidade de convergência aumente, a variação dos pesos entre iterações será também maior podendo afetar a convergência do algoritmo (Haykin 1994).

Utilizando o treinamento por reforço, como se verá nos próximos capítulos, dependendo das regras adotadas para atualizar os pesos, é necessário utilizar taxas de aprendizado que regulem a variação dos pesos por iteração e as considerações mencionadas acima para o caso do algoritmo de retropropagação são válidas também para este caso.

2.6.2 Modos de Treinamento

Quando todo o conjunto de treinamento é apresentado a uma rede neural, obtém-se o que se denomina de *época*. O processo de treinamento continua até que seja alcançado um número máximo de épocas ou o erro quadrático médio atinge um valor suficientemente pequeno. Para um determinado conjunto de treinamento, o algoritmo de retropropagação pode ser aplicado de dois modos diferentes (Ballini 2000):

1. **Padrão a padrão**: No aprendizado por retropropagação padrão a padrão, os pesos da rede são atualizados a cada padrão entrada-saída apresentado, ou seja, apenas o gradiente do correspondente padrão é calculado; este é utilizado imediatamente para atualizar os pesos (equação (2.15)), considerando assim, um único padrão por vez para o treinamento. Deste modo, por cada época, os pesos são ajustados N vezes, sendo cada ajuste baseado unicamente no gradiente do erro de um único padrão por vez.

2. **Em lote ou batelada**: Neste modo de treinamento, um único ajuste dos pesos é feito por cada época. Todos os padrões de treinamento são propagados pela rede, e os correspondentes erros são calculados, sendo que o gradiente utilizado para ajustar os pesos será a soma dos gradientes para cada padrão apresentado na época.

É recomendável apresentar os padrões durante o treinamento em ordem aleatória para cada época. Isto faz que a busca no espaço dos pesos seja estocástica, garantindo que a ordem dos padrões de treinamento não represente informação significativa alguma e , assim, condicionar o ajuste dos pesos.

2.6.3 Critérios de Parada

Não existe critério de parada para o algoritmo de retropropagação que garanta a obtenção de uma solução. Contudo, existem alguns critérios que podem ser utilizados para dar término ao processo de treinamento. Segundo (Haykin 1994), dois critérios de parada são comumente utilizados:

1. Considerando as propriedades dos mínimos locais ou globais da superfície de erro. Sendo $\mathbf{w} = \mathbf{w}^*$ um mínimo local ou global, o vetor gradiente calculado para este mínimo será aproximadamente nulo. Assim, considera-se que a convergência tenha sido alcançada quando a norma Euclidiana do vetor gradiente seja menor que um valor especificado suficientemente pequeno.

2. Um segundo critério de parada é quando os valores do erro quadrático médio ou da função objetivo tornam-se estacionários;

Um outro critério a considerar consiste na unificação da capacidade de generalização da rede. Uma rede neural com capacidade de generalização deve ser capaz de, uma vez fornecidos diferentes padrões de treinamento e teste, o mapeamento entre os dados entrada-saída estimados pela rede já treinada é satisfatório. Uma forma de garantir esta característica, é aplicar o mecanismo de validação cruzada.

O mecanismo de validação cruzada se inicia gerando um conjunto de padrões de treinamento e um conjunto de padrões de teste, este último para avaliar a eficiência da rede neural. Durante o processo de treinamento, após cada ajuste de pesos, é necessário avaliar a rede tanto com o conjunto de treinamento como com o conjunto de teste. Quando o erro de teste começar a ter uma tendência crescente, o processo de treinamento se dá por concluído.

Durante o treinamento do modelo de rede neural proposto neste trabalho e dos modelos de redes neurais utilizados para comparação, será utilizado o segundo critério de parada e o número máximo de épocas.

2.6.4 Inicialização dos Pesos

A qualidade e eficiência do aprendizado supervisionado em redes multicamadas, depende não só da estrutura da rede, funções de ativação e regras de aprendizagem mas também dos valores iniciais dos parâmetros ajustáveis. Em geral, não é possível ter um conhecimento prévio de valores ótimos para estes parâmetros, pois isto depende tanto do conjunto de treinamento como da natureza da solução. O conjunto inicial de pesos a ser utilizado influi diretamente na velocidade de aprendizagem e na qualidade da solução final. Uma escolha inicial pouco adequada pode gerar problemas de mínimos locais pobres ou de saturação prematura. Quando existe alguma informação a *priori* sobre os dados, esta informação pode ser utilizada para inicializar os pesos. Quando esta informação não é utilizada, recomenda-se que a inicialização dos parâmetros seja feita aleatoriamente, com distribuição uniforme e magnitude pequena (Haykin 1994), (Ballini 2000).

2.7 Resumo

Neste capítulo foram apresentados os principais conceitos da teoria de redes neurais, assim como uma classificação das estruturas básicas: redes neurais estáticas e redes neurais recorrentes. Neste trabalho o tipo de redes de interesse são redes parcial e totalmente recorrente, com entradas e saídas reais e pesos assimétricos, com capacidade de aprendizado de trajetórias.

Também foi apresentada uma classificação dos métodos de treinamento; dentre estes métodos, um dos mais utilizados é o método do gradiente do erro ou retropropagação. Finalmente, foram mencionadas considerações importantes para o processo de treinamento.

Capítulo 3

Sistemas Fuzzy

3.1 Introdução

Neste capítulo são apresentados os principais conceitos da teoria de conjuntos e sistemas fuzzy, que servirão como base teórica para um melhor entendimento dos próximos capítulos.

A seção 3.2 apresenta as definições básicas da teoria dos sistemas fuzzy, assim como uma classificação destes sistemas do ponto de vista estrutural.

Na seção 3.3 são brevemente descritos os sistemas fuzzy estáticos, detalhando o sistema fuzzy adaptativo proposto em (Wang 1994), que será utilizado depois, para modelagem e controle de sistemas.

Conceitos sobre sistemas fuzzy recorrentes são revistos na seção 3.4. Este capítulo finaliza com um comentário sobre os métodos de treinamento de sistemas fuzzy adaptativos, podendo notar claramente com estas duas ultimas seções, a influência das redes neurais nos sistemas fuzzy.

3.2 Teoria de Conjuntos Fuzzy

Existem duas fontes principais de informações consideradas na engenharia: os *sensores* que oferecem medidas numéricas das variáveis de interesse, e os *especialistas* que fornecem instruções de ordem lingüística e descrições qualitativas do sistema. O primeiro tipo de informação é denominado *informação numérica* e o segundo *informação lingüística*. A informação numérica é representada por números como 35, -0,98, etc; enquanto a informação

lingüística é representada por conceitos como *grande, bom, baixo*, etc.

Em geral, o conhecimento sobre os diversos sistemas ou modelos é fuzzy e a representação do conhecimento utilizando modelos clássicos resulta em uma representação complexa e pouco eficiente.

Como exemplo, considere um copo cheio de água até a metade da sua capacidade. Na teoria de conjuntos clássica, pode-se dizer que o copo **ou** está cheio **ou** está vazio. Utilizando a teoria dos conjuntos fuzzy, pode-se dizer que o copo está 50% cheio **e** 50% vazio, o que resulta ser uma afirmação mais realística.

Um outro exemplo é o seguinte: se sabe que o espaço percorrido por um corpo rígido é proporcional a velocidade deste, multiplicada pelo tempo decorrido; assim, poderia-se dizer: "SE a velocidade do corpo A é *maior* que a do corpo B, ENTÃO o espaço percorrido por A no mesmo intervalo de tempo será *maior* que o percorrido por B", sendo que esta informação lingüística pode ser incorporada para a representação do sistema ou resolução do problema.

O conceito de Conjuntos Fuzzy foi inicialmente proposto por L. Zadeh (Zadeh 1965) como uma generalização da teoria de conjuntos clássica. Um subconjunto fuzzy pode ser considerado como uma função que atribui a elementos de um universo valores de pertinência no intervalo $I = [0, 1]$ ao contrário da teoria de conjuntos da lógica clássica onde os valores de pertinência são 0 ou 1. Assim, voltando ao primeiro exemplo, poderia-se dizer que o copo se encontra cheio com um grau de 0.50 ao mesmo tempo que o copo se encontra vazio com um grau de 0.50.

A teoria de conjuntos fuzzy e a lógica subjacente é uma "ponte" para aproximar a lógica executada pela máquina ao raciocínio humano. Um sistema fuzzy é capaz de capturar informações vagas descritas em uma linguagem natural (variáveis lingüísticas) e converte-las para um formato numérico, de fácil manipulação pelo computador, possuindo assim, habilidades no manuseio de informações qualitativas e imprecisas, o que resulta em um desempenho estável e robusto.

O objetivo dos sistemas fuzzy é combinar de forma efetiva tanto a informação numérica como a lingüística para obter sistemas eficientes e aplicáveis a diversas áreas como economia, computação, engenharia, medicina, etc.

3.2.1 Definição

Um *universo de discurso* é um conjunto clássico que contém todos os objetos de um domínio de interesse representados por $\mathbf{x} = [x_1, \ldots, x_n]$.

Seja $U \subseteq \Re^n$ um universo de discurso. Um subconjunto fuzzy A em U é caracterizado pela sua função de pertinência $\mu_A : U \rightarrow [0, 1]$ com $\mu_A(\mathbf{x})$ representando o grau de pertinência de $\mathbf{x} \in U$ em A.

3.2.2 Funções de Pertinência

Algumas das funções de pertinência comumente utilizada na literatura (Pedrycz & Gomide 1998), (Yager & Filev 1994), são ilustradas na Figura 3.1. Estas funções de pertinência são definidas a seguir:

1. *Função Triangular*:

$$\mu_A(x) = \begin{cases} 0 & \text{se } x < a, \\ \frac{x-a}{m-a} & \text{se } x \in [a, m], \\ \frac{b-x}{b-m} & \text{se } x \in [m, b] \\ 0 & \text{se } x > b, \end{cases} \tag{3.1}$$

   onde m é o valor modal, e a e b são os limites superior e inferior, respectivamente.

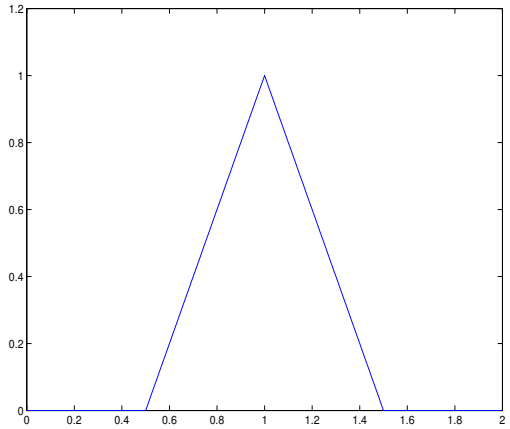
2. *Função Trapezoidal*:

$$\mu_A(x) = \begin{cases} 0 & \text{se } x < a, \\ \frac{x-a}{m-a} & \text{se } x \in [a, m], \\ 1 & \text{se } x \in [m, n], \\ \frac{b-x}{b-m} & \text{se } x \in [n, b] \\ 0 & \text{se } x > b, \end{cases} \tag{3.2}$$

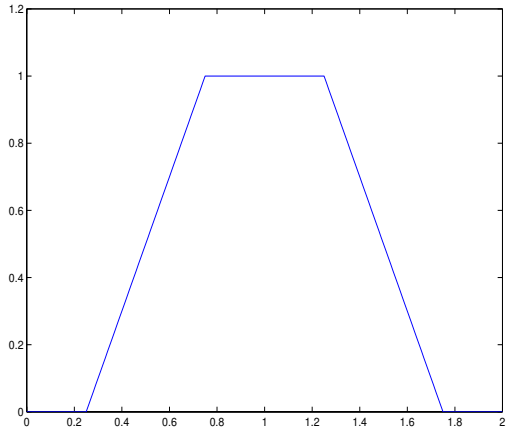
3. *Função Gaussiana*:

$$\mu_A(x) = \exp^{-\beta(x-m)^2} \tag{3.3}$$

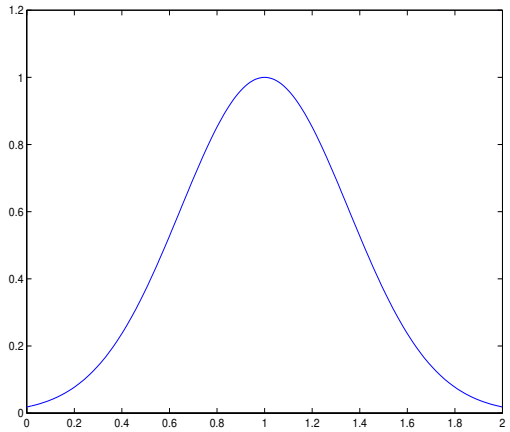
   onde $\beta > 0$.



(1) $m = 1$, $a = 0.5$, $b = 1.5$



(2) $a = 0.25$, $b = 1.75$, $m = 0.75$, $n = 1.25$



(3) $m = 1$, $\beta = 4$

Figura 3.1: Funções de pertinência. (1) Função triangular, (2) Função Trapezoidal, (3) Função Gaussiana.

Funções de pertinência do tipo Gaussiana serão utilizadas na implementação do sistema fuzzy adaptativo proposto por Wang (Wang 1994), e que será apresentado nas próximas seções deste capítulo. Nas estruturas neurofuzzy propostas neste trabalho serão utilizadas funções de pertinência triangulares, pois estas fornecem um bom desempenho a um baixo custo computacional.

3.2.3 Normas Triangulares

As normas triangulares são modelos genéricos das operações de união e intersecção da teoria de conjuntos fuzzy e da conjunção e disjunção na lógica correspondente, devendo apresentar as propriedades de comutatividade, associatividade, monotonicidade e satisfazer as condições de contorno (Pedrycz & Gomide 1998). As normas triangulares são chamadas de t-normas e s-normas, podendo ser formalmente definidas como:

Definição 3.1. *Uma t-norma é uma função binária* $t : [0,1]^2 \rightarrow [0,1]$ *tal que:*

1. *Comutatividade:* $x\ t\ y = y\ t\ x$
2. *Associatividade:* $x\ t\ (y\ t\ z) = (x\ t\ y)\ t\ z$
3. *Monotonicidade: Se* $x \leq y$ *e* $w \leq z$ *então* $x\ t\ w \leq y\ t\ z$
4. *Condições de contorno:* $0\ t\ x = 0, \quad 1\ t\ x = x$

Definição 3.2. *Uma s-Norma, também conhecida como co-norma triangular, é uma função binária* $s : [0,1]^2 \rightarrow [0,1]$*, tal que:*

1. *Comutatividade:* $x\ s\ y = y\ s\ x$
2. *Associatividade:* $x\ s\ (y\ s\ z) = (x\ s\ y)\ s\ z$
3. *Monotonicidade: Se* $x \leq y$ *e* $w \leq z$ *então* $x\ s\ w \leq y\ s\ z$
4. *Condições de contorno:* $0\ s\ x = x, \quad 1\ s\ x = 1$

Pode-se observar claramente que t-normas incluem a operação *min* (intersecção padrão) e s-normas a operação *max* (união padrão). Ao longo deste trabalho e sem perda de generalidade, as normas triangulares a serem utilizadas são o *produto algébrico* e a *soma probabilística*, definidas para a t-norma e a s-norma respectivamente como segue:

$$x \; t \; y = xy \tag{3.4}$$

$$x \; s \; y = x + y - xy \tag{3.5}$$

Outros tipos de normas triangulares encontram-se definidas em (Pedrycz & Gomide 1998).

3.2.4 Classificação de Sistemas Fuzzy

Os sistemas fuzzy podem ser classificados em três tipos (Wang 1994):

1. ***Sistemas fuzzy puros***: Um sistema fuzzy puro é formado por uma base de regras do tipo SE [*antecedente*] ENTÃO [*conseqüente*], e um mecanismo de inferência fuzzy como mostra a Figura 3.2. O mecanismo de inferência fuzzy determina um mapeamento dos conjuntos fuzzy do universo de discurso da entrada em U com os conjuntos fuzzy da saída pertencente ao universo de discurso V. A base de regras fuzzy adota a seguinte forma:

$$R^{(l)}: \quad \text{SE} \quad [x_1 \text{ É } A_1^l] \quad \text{E} \quad \cdots \quad [x_n \text{ É } A_n^l] \quad \text{ENTÃO} \quad [y \text{ É } B^l] \tag{3.6}$$

   onde A_i^l e B^l são os subconjuntos fuzzy em U e V; $\mathbf{x} = [x_1, \ldots, x_n] \in U$ e $y \in V$ são as variáveis de base das variáveis lingüísticas respectivamente, para $l = 1, \ldots, M$, sendo M o número de regras que formam a base de regras do sistema fuzzy.

2. ***Sistemas fuzzy funcionais (Takagi e Sugeno)***: Este sistema tem como principal diferença do sistema fuzzy puro, o cálculo da saída y, a qual é computada como uma função da entrada, como por exemplo uma combinação linear de $\mathbf{x}$ (Takagi & Sugeno 1985), sendo neste caso as regras fuzzy definidas como:

$$R^{(l)}: \quad \text{SE} \quad [x_1 \text{ É } A_1^l] \quad \text{E} \quad \cdots \quad [x_n \text{ É } A_n^l] \quad \text{ENTÃO} \quad [y^l = a_0^l + a_1^l x_1 + \cdots + a_n^l x_n] \tag{3.7}$$

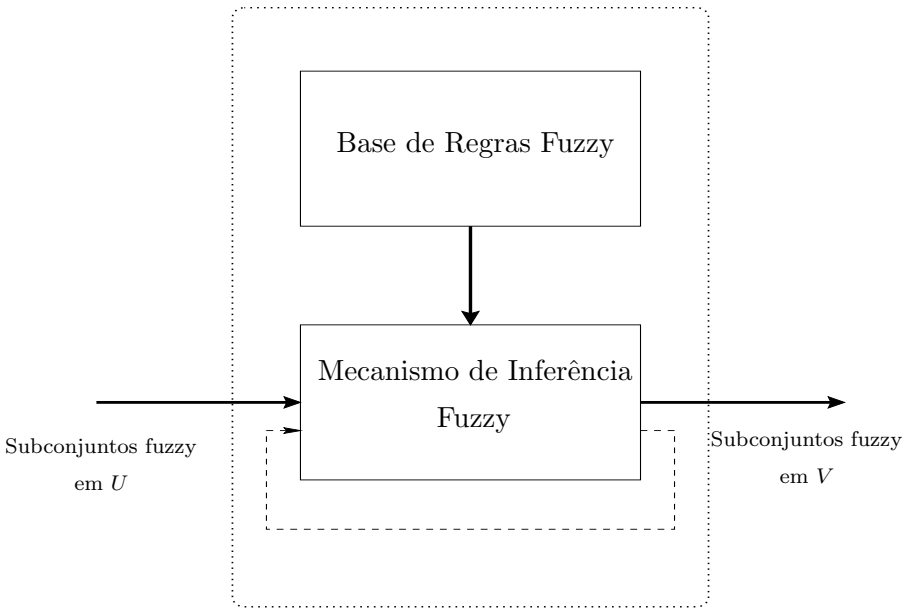


Figura 3.2: Sistema fuzzy "puro".

onde a_i^l são coeficientes reais e y^l é a saída real da regra l, $l = 1, \ldots, M$. Enquanto os antecedentes das regras continuam sendo fuzzy, o conseqüente não. Ou seja, dada uma entrada $\mathbf{x}$, a saída $y = y(\mathbf{x})$ do sistema é calculada como uma média ponderada das saídas y^l de cada regra:

$$y(\mathbf{x}) = \frac{\sum_{l=1}^{M} w^l y^l}{\sum_{l=1}^{M} w^l} \tag{3.8}$$

onde w^l é o grau de ativação da regra $R^{(l)}$ sendo calculado como:

$$w^l = \prod_{i=1}^{n} \mu_{A_i^l}(x_i) \tag{3.9}$$

A Figura 3.3 representa este sistema.

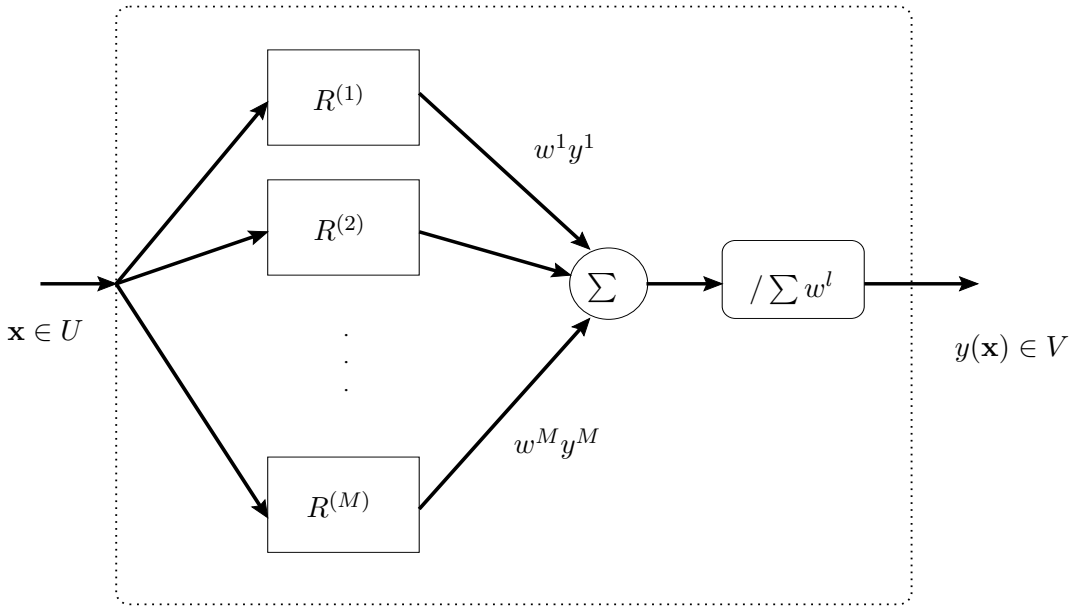


Figura 3.3: Sistema fuzzy do tipo Takagi e Sugeno.

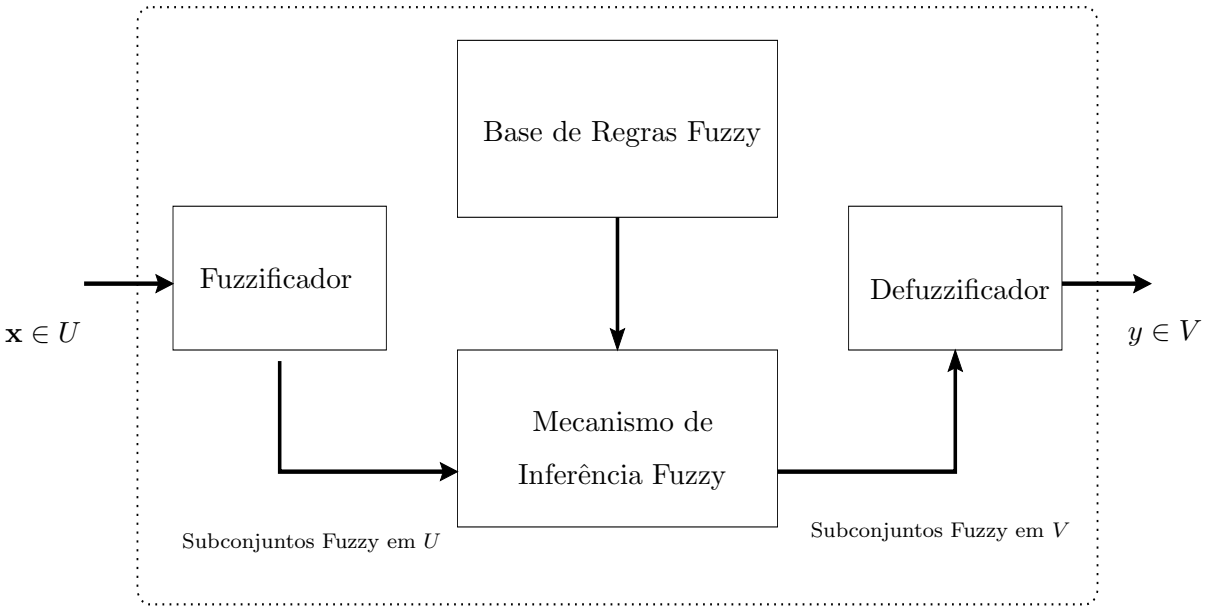


Figura 3.4: Sistema fuzzy com fuzzificador e defuzzificador.

3. ***Sistemas fuzzy com fuzzificador e defuzzificador***: A configuração básica de um sistema fuzzy com fuzzificador e defuzzificador é mostrado na Figura 3.4. Este sistema foi inicialmente proposto por Mamdani (Mamdani 1974).

O mecanismo de fuzzificação mapeia valores reais de $\mathbf{x}$ em U a subconjuntos fuzzy em U e o processo de defuzzificação mapeia subconjuntos fuzzy em V a valores reais de y em V.

Assim como as redes neurais podem ser classificadas como redes neurais estáticas e recorrentes, os sistemas fuzzy também podem ser classificados como sistemas fuzzy estáticos e sistemas fuzzy recorrentes.

3.3 Sistemas Fuzzy Estáticos

Os *sistemas fuzzy estáticos* são definidos como modelos fuzzy com processamento direto das informações, ou seja, sem realimentação.

Um *sistema fuzzy adaptativo* é definido como um sistema fuzzy junto com um algoritmo de aprendizagem. Um exemplo de sistema adaptativo é aquele construído a partir de um conjunto de regras SE-ENTÃO utilizando o princípio dos sistemas fuzzy, sendo os parâmetros da base de regras ou do sistema de inferência ajustados via algum método de otimização, tendo como referência informações numéricas. Maiores detalhes de sistemas fuzzy adaptativos podem ser encontrados em (Bersini & Gorrini 1992), (Bersini & Gorrini 1993) e (Yager & Filev 1994).

Em (Wang 1994), um sistema fuzzy adaptativo é utilizado para modelagem de sistemas não lineares e posteriormente para controle *off-line*, ajustando o número de regras durante o treinamento do sistema. Este sistema foi aplicado para modelagem e controle e será utilizado como um modelo de comparação com a abordagem proposta neste trabalho. A seguir é brevemente apresentado o sistema fuzzy adaptativo proposto por Wang (Wang 1994).

3.3.1 Sistema Fuzzy Adaptativo

Seja $\mathbf{x} \in U$, onde $U \subseteq \Re^n$ é o universo de discurso para $\mathbf{x}$, e A um subconjunto fuzzy em U. O Sistema Fuzzy Adaptativo (**SFA**) (Wang 1994) apresenta as etapas de fuzzificação e defuzzificação, tendo as seguintes características:

1. **Base de regras**: Sejam M regras do tipo SE-ENTÃO com funções de pertinência Gaussianas e M definido pelo usuário. Assim:

$$\mu_{A_i^l}(x_i) = \exp\left[-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l}\right)^2\right] \tag{3.10}$$

onde $\bar{x}_i^l$ e $\sigma_i^l > 0$ são os valores modais e dispersão, respectivamente, ambos ajustáveis.

2. **Mecanismo de Inferência**: Cada regra fuzzy SE-ENTÃO pode ser interpretada como uma relação $A \rightarrow B$, sendo $A = A_1 \times \cdots \times A_n$, com A_i um subconjunto nebuloso de $\Re$ e B um subconjunto nebuloso de V. Esta relação pode ser implementada por exemplo, utilizando diferentes normas triangulares. Dada uma entrada $A'(\mathbf{x})$ pode-se determinar a saída correspondente por:

$$\mu_{B'}(y) = \sup_{\mathbf{x} \in U}[\ \min\ [A'(\mathbf{x}), \mu_{A \rightarrow B}(\mathbf{x}, y)]\ ]$$

onde

$$\mu_{A \rightarrow B}(\mathbf{x}, y) = \mu_A(\mathbf{x})\mu_B(y) \tag{3.11}$$

é uma possível interpretação da regra $R = A \rightarrow B$ (versão simplificada da regra em (3.6)) utilizando a t-norma definida por (3.4).

3. **Fuzzificador**: O mecanismo de fuzzificação executa um mapeamento da entrada $\mathbf{x} \in U$ para um conjunto fuzzy A em U. A fuzzificação é em geral, do tipo unitária (*singleton*) e definida como $\mu_A(\mathbf{x}') = 1$ para $\mathbf{x}' = \mathbf{x}$ e $\mu_A(\mathbf{x}') = 0$ para todo $\mathbf{x}' \in U$ tal que $\mathbf{x}' \neq \mathbf{x}$.

4. **Defuzzificador**: O mecanismo de defuzzificação executa um mapeamento do subconjunto fuzzy B em V para um valor real $y \in V$. A defuzzificação é feita da seguinte maneira:

$$y = f(\mathbf{x}) = \frac{\sum_{l=1}^{M} \bar{y}^l \cdot \mu_{B^l}(\bar{y}^l)}{\sum_{l=1}^{M} \mu_{B^l}(\bar{y}^l)} \tag{3.12}$$

onde $\bar{y}^l$ é o valor para o qual $\mu_{B^l}(y)$ assume o máximo valor, e $\mu_{B^l}(y)$ é dado por (3.11).

Desta forma, a partir de (3.11) e de (3.12), o sistema fuzzy com as características dadas fornece como saída:

$$y = f(\mathbf{x}) = \frac{\sum_{l=1}^{M} \bar{y}^l \prod_{i=1}^{n} \mu_{A_i^l}(x_i)}{\sum_{l=1}^{M} \prod_{i=1}^{n} \mu_{A_i^l}(x_i)} \tag{3.13}$$

com $\mu_{B^l}(\bar{y}^l) = 1$, devido ao tipo de fuzzificação escolhido (unitária). Substituindo a equação (3.10) em (3.13) obtém-se finalmente:

$$y = f(\mathbf{x}) = \frac{\sum_{l=1}^{M} \bar{y}^l \cdot \left[\prod_{i=1}^{n} \exp\left(- \left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l} \right)^2 \right) \right]}{\sum_{l=1}^{M} \left[\prod_{i=1}^{n} \exp\left(- \left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l} \right)^2 \right) \right]} \tag{3.14}$$

A estrutura acima para o sistema fuzzy é utilizada como inicialização para o modelo, pois utilizando técnicas de agrupamento (*clusterização* ou *clustering*) é possível definir durante o treinamento o número de regras M.

O sistema fuzzy proposto em (Wang 1994), é baseado na geração de uma base de regras, cujo número de regras é igual ao número de padrões do conjunto de treinamento, ou seja, com uma regra responsável por cada par entrada-saída, admitindo-se conjuntos de treinamento pequenos.

Quando o sistema a modelar é mais complexo ou requer conjuntos de treinamento maiores, utiliza-se um método de agrupamento (e.g. *Nearest Neighborhood*) para se determinar grupos

e os respectivos centros. Estes centros são utilizados como padrões representativos na geração de regras fuzzy.

Sejam N padrões de treinamento $(\mathbf{x}^l, y^l)$, $l = 1, \ldots, N$, para N pequeno. O objetivo é construir um sistema fuzzy que mapeie cada padrão do conjunto de treinamento, ou seja, $N = M$. Para um dado $\epsilon > 0$, é necessário que o sistema satisfaça a condição $|f(\mathbf{x}^l) - y^l| < \epsilon$ para todo $l = 1, \ldots, N$ e onde $|\cdot|$ representa a distância Euclidiana.

O sistema fuzzy é construído da seguinte maneira:

$$y = f(\mathbf{x}) = \frac{\sum_{l=1}^{N} y^l \cdot \exp\left(-\frac{|\mathbf{x} - \mathbf{x}^l|^2}{\sigma^2}\right)}{\sum_{l=1}^{N} \exp\left(-\frac{|\mathbf{x} - \mathbf{x}^l|^2}{\sigma^2}\right)} \tag{3.15}$$

onde $\exp(-\frac{|\mathbf{x} - \mathbf{x}^l|^2}{\sigma^2}) = \prod_{i=1}^{n} \exp\left(-(\frac{x_i - x_i^l}{\sigma})^2\right)$. Dado que cada regra representa um padrão, o centro de cada regra será o próprio padrão y^l. Assim, se fazemos $y^l = \bar{y}$ então (3.14) pode ser representada por (3.15). Para sistemas mais complexos, ou seja, para $N >> M$, o algoritmo é como segue:

(1.) Definir o primeiro centro $\mathbf{c}^1$ como sendo o primeiro padrão de entrada $\mathbf{x}^l$, $A^l(1) = y^l$ e $B^l(1) = 1$. Selecionar o raio r e a dispersão σ.

(2.) Supor que após k iterações, existem M centros $\mathbf{c}^1$, $\mathbf{c}^2$, ..., $\mathbf{c}^M$. Calcular as distâncias de $\mathbf{x}^k$ para cada um destes centros $|\mathbf{x}^k - \mathbf{c}^l|$, $l = 2, \ldots, M$, e seja $\mathbf{c}^{l_k}$ o centro mais próximo a $\mathbf{x}^k$. Logo:

(2.1) Se $|\mathbf{x}^k - \mathbf{c}^{l_k}| > r$, estabelecer $\mathbf{x}^k$ como um novo centro $\mathbf{c}^{M+1} = \mathbf{x}^k$. Definir $A^{M+1}(k) = y^k$, $B^{M+1}(k) = 1$ e manter $A^l(k) = A^l(k-1)$, $B^l(k) = B^l(k-1)$ para $l = 1, \ldots, M$.

(2.2) Se $|\mathbf{x}^k - \mathbf{c}^{l_k}| < r$, faça:

$$A^{l_k}(k) = A^{l_k}(k-1) + y^k \tag{3.16}$$

$$B^{l_k}(k) = B^{l_k}(k-1) + 1 \tag{3.17}$$

e definir:

$$A^l(k) = A^l(k-1) \tag{3.18}$$

$$B^l(k) = B^l(k-1) \tag{3.19}$$

para $l = 1,\ 2, \ldots, M$ com $l \neq l_k$.

(3.) A saída do sistema fuzzy adaptativo no passo k é calculada da seguinte forma:

$$y = f(x) = \frac{\sum_{l=1}^{M} A^l(k) \cdot \exp\left(-\frac{|\mathbf{x} - \mathbf{c}^l|^2}{\sigma^2}\right)}{\sum_{l=1}^{M} B^l(k) \cdot \exp\left(-\frac{|\mathbf{x} - \mathbf{c}^l|^2}{\sigma^2}\right)} \tag{3.20}$$

se a distância $|\mathbf{x}^k - \mathbf{c}^{l_k}| < r$.Caso contrário, $\mathbf{x}^k$ é definido como um novo centro e $M = M + 1$.

O raio r determina a complexidade do sistema fuzzy adaptativo. Para valores pequenos de r, o número de centros M poderá ser grande, enquanto que para valores maiores de r, o número de centros poderá ser menor e comprometer a precisão.

3.4 Sistemas Fuzzy Recorrentes

Um sistema fuzzy recorrente é um sistema com memória, ou seja, é capaz de representar um comportamento dinâmico. Em um sistema fuzzy recorrente de primeira ordem, uma ou mais variáveis aparecem tanto no antecedente como no conseqüente da seguinte forma (ver Figura 3.5):

$$\text{SE}\ \ [y(k-1)\ \text{É}\ A_i^l]\ \ \text{E}\ \ [x(k)\ \text{É}\ A_j^l]\ \ \text{ENTÃO}\ \ [y(k)\ \text{É}\ B^l] \tag{3.21}$$

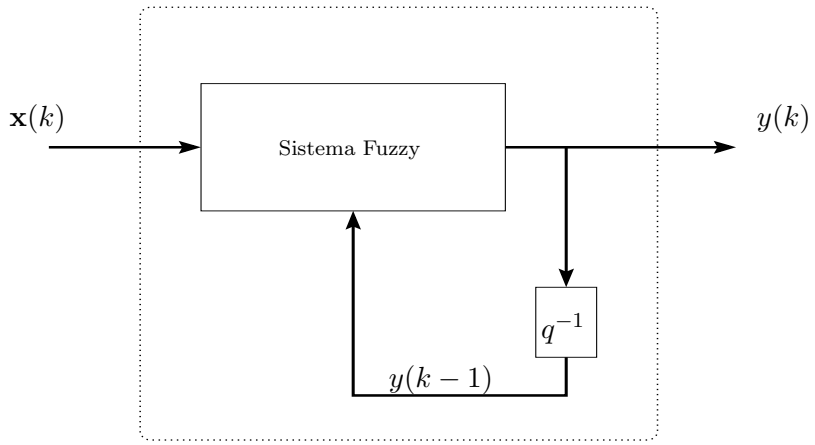


Figura 3.5: Sistema fuzzy recorrente de primeira ordem.

Um sistema fuzzy de primeira ordem pode ser utilizado para aproximar sistemas clássicos de primeira ordem; assim, é necessária a inclusão de variáveis internas e, desta forma, enriquecer a estrutura para que utilizando ainda um sistema fuzzy de primeira ordem seja possível obter um bom desempenho na aproximação de sistemas de ordem superior.

Um modelo de sistema fuzzy recorrente com variáveis internas é mostrado na Figura 3.6. Neste caso, o sistema contém uma única variável interna $\Phi(k)$ e a dinâmica do sistema fuzzy pode ser dividido em dois subsistemas onde:

$$R_1^{(l)}: \text{ SE } [x(k) \text{ É } A_i^l] \text{ E } [y(k-1) \text{ É } A_j^l] \text{ ENTÃO } [\Phi(k) \text{ É } \gamma^l] \tag{3.22}$$

representa a dinâmica relacionada a variável interna (Subsistema fuzzy 1). Logo, a saída do sistema estará definida como:

$$R_2^{(l)}: \text{ SE } [x(k) \text{ É } A_i^l] \text{ E } [\Phi(k) \text{ É } \gamma^l] \text{ E } [y(k-1) \text{ É } A_j^l] \text{ ENTÃO } [y(k) \text{ É } B^l] \tag{3.23}$$

a qual representa a dinâmica do Subsistema fuzzy 2.

Em (Bersini & Gorrini 1994), um sistema fuzzy recorrente treinado por retropropagação é proposto; a principal diferença comparando com outros sistemas fuzzy, é que não se tenta otimizar todos os parâmetros das regras do sistema de inferência, mas sim os termos lingüísticos (Bersini & Gorrini 1992), (Bersini & Gorrini 1993).

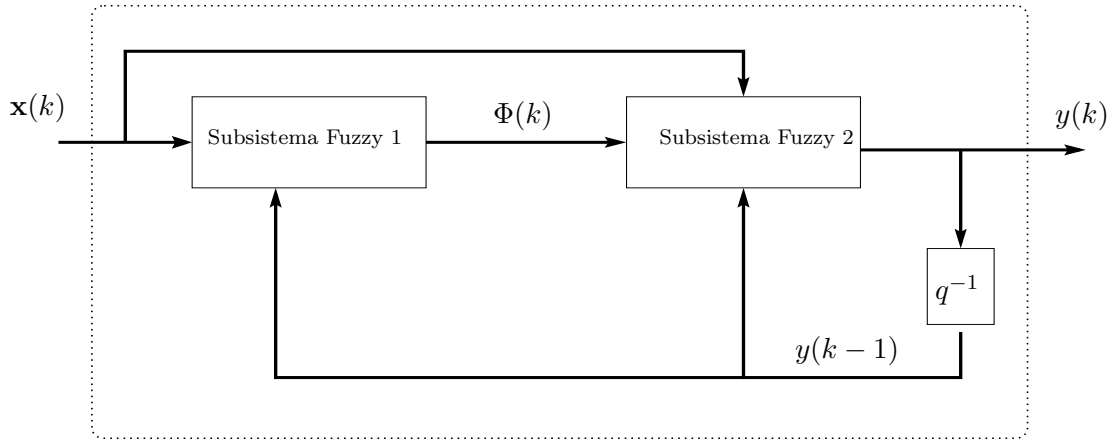


Figura 3.6: Sistema fuzzy recorrente com variáveis internas.

3.5 Métodos de Treinamento de Sistemas Fuzzy

Para o treinamento dos sistemas fuzzy, é possível utilizar os algoritmos de aprendizado apresentados no capítulo 2, ou seja, aprendizado supervisionado, aprendizado não supervisionados e aprendizado por reforço. Grande parte da literatura faz menção ao aprendizado supervisionado com o método de retropropagação, podendo ser possível a adoção de outros métodos, considerando sempre as características próprias de cada sistema.

Os parâmetros do sistema fuzzy definido por (3.14) podem ser ajustados via o algoritmo de retropropagação, uma vez dado pelo usuário um número fixo de regras do tipo SE-ENTÃO, e onde os parâmetros ajustáveis seriam não só $\bar{x}_i^l$ e σ_i^l mas também $\bar{y}^l$.

3.6 Resumo

Neste capítulo foram apresentados os conceitos básicos da teoria de conjuntos e sistemas fuzzy necessários para o desenvolvimento dos próximos capítulos, assim como, uma classificação quanto a estrutura e a dinâmica envolvida.

Capítulo 4

Sistemas Híbridos Neurofuzzy

4.1 Introdução

Como foi visto nos dois últimos capítulos, as redes neurais artificiais e os sistemas fuzzy proporcionam modelos computacionais complementares. Como resultado de sua combinação, surgem os denominados *sistemas neurofuzzy*. Estes sistemas agregam as propriedades de aproximação universal e aprendizagem das redes neurais com a facilidade de manipulação de informação lingüística e dedução dos sistemas fuzzy em um único sistema, resultando um modelo eficiente e robusto, com capacidade para tratar as incertezas existentes na informação. Acrescentando recorrência aos sistemas neurofuzzy estáticos, é possível obter sistemas capazes de, além de tratar informação lingüística e possuir capacidade de aprendizado, tratar sistemas dinâmicos, caso onde é necessário manipular relações temporais.

Neste capítulo, duas estruturas de redes neurofuzzy recorrentes e uma rede neurofuzzy estática são propostas. A estrutura base é uma rede neurofuzzy com recorrência interna global nos neurônios lógicos, que constituem uma camada intermediária. Eliminando laços de recorrência entre diferentes neurônios lógicos desta estrutura obtém-se uma segunda estrutura, esta com recorrência interna local. Finalmente, retirando a recorrência local obtém-se uma estrutura de rede neurofuzzy estática. Os detalhes destas estruturas e de outras que, posteriormente, serão utilizadas para comparação e testes, serão apresentados nas próximas seções.

4.2 Redes Neurais vs. Sistemas Fuzzy

Os sistemas neurofuzzy têm atraído o interesse da comunidade científica nestes últimos anos pois estes sistemas têm demonstrado um melhor desempenho em diferentes aplicações quando comparadas com sistemas puros (Buckley & Hayashi 1994), (Buckley & Hayashi 1995). Diversas estruturas foram propostas na literatura (Zhang & Morris 1999), (Lee & Teng 2000), (Lin & Wai 2001).

Segundo A. Abraham (Abraham n.d.), um sistema de inferência fuzzy é capaz de utilizar o conhecimento especialista, armazenando informação na base de regras associada ao sistema e realizando o raciocínio aproximado para inferir o valor da saída correspondente. Para construir um sistema de inferência fuzzy (*Fuzzy Inference System* - FIS), é necessário definir o número de conjuntos fuzzy que constituem as partições dos universos de discruso, os operadores lógicos e a base de conhecimento.

As redes neurais se mostram deficientes para representar o conhecimento de forma explícita em sua estrutura, pois estes modelos não são capazes de definir automaticamente as regras utilizadas para as tomadas de decisões (Fullér 1995). Por outro lado, os FIS valem-se amplamente de suas características com relação a manipulação de termos lingüísticos. A Tabela 4.1 apresenta uma comparação entre as redes neurais artificiais e os sistemas de inferência fuzzy.

Tabela 4.1: Comparação entre as redes neurais artificiais e os sistemas de inferência fuzzy.

Redes Neurais Artificiais	**Sistemas de Inferência Fuzzy**
Conhecimento a priori não utilizado	Conhecimento a *priori* pode ser incorporado
Capacidade de aprendizado	Utiliza o conhecimento lingüístico
Caixa preta	De fácil interpretação (regras $SE - ENT\tilde{A}O$)
Algoritmos de aprendizado complexos	Fácil interpretação e implementação
Dificuldade para extração de conhecimento	Conhecimento disponível

Os sistemas neurofuzzy combinam as vantagens destas duas abordagens, obtendo-se um sistema com capacidade de aprendizado, capaz de aproveitar informação lingüística e numérica através da base de regras do sistema de inferência e utilizar conhecimento *a priori* para definir

a estrutura do sistema.

O aproveitamento do conhecimento *a priori* na resolução de problemas, resulta em uma forma de compensar as deficiências que as redes neurais possuem, quando comparadas a outras abordagens e sistemas inteligentes (Castro et al. 2002), (Benítez et al. 1997). Para obter uma maior compreensão da tomada de decisões de uma rede neural, diversos estudos tem sido feitos (Alexander & Mozer 1999). Técnicas de extração de regras a partir de redes neurais são também de grande interesse para este fim (Fu 1994), (Setiono 2000).

Entretanto, a capacidade de aprendizado é, sem dúvida, a característica mais importante das redes neurais que os sistemas fuzzy herdam, gerando as redes neurofuzzy. É através da aprendizagem, que estes dois componentes da inteligência computacional, quando combinados, transformam-se em um único sistema *neurofuzzy* que supera as deficiências individuais.

Uma limitação das redes neurofuzzy estáticas é a sua restrita aplicação devido à sua estrutura não recorrente ou à falta de processos eficientes de aprendizado quando as conexões de realimentação são introduzidas.

Como visto no Capítulo 2, estruturas neurais recorrentes podem ser classificadas em duas grandes classes, isto é, redes parcial (Figura 2.7) ou totalmente recorrentes (Figura 2.8). A presença de realimentação numa rede neural permite a criação de representações internas e mecanismos de memória capazes de processar e armazenar relações temporais (Santos & Von Zuben 1999). Esta classificação pode ser aplicada também a sistemas neurofuzzy.

Embora a análise e a síntese de sistemas recorrentes sejam mais complexas, estes modelos têm demonstrado desempenho superior em diversas aplicações (Ku & Lee 1995), particularmente em identificação de sistemas não lineares e estimação de parâmetros, reconhecimento de padrões, controle de processos, previsão de vazões, aproximação de funções e previsão de séries financieras entre outras. Como conseqüência da sua estrutura, as redes recorrentes são mais difíceis de serem analisadas, seu processo de treinamento é mais trabalhoso, e os algoritmos de aprendizagem são mais complexos e lentos (Lee & Teng 2000).

Acrescentando recorrência em redes neurofuzzy estáticas, é possível obter sistemas que,

além de poder aproveitar o conhecimento a priori de informações lingüísticas, possam tratar de forma eficiente relações temporais e não linearidades na dinâmica.

As relações temporais são induzidas por realimentação global ou parcial, fornecendo assim os elementos de memória que expandem a capacidade da redes para capturar representações temporais.

A seguir, serão apresentados os neurônios lógicos denominados neurônios lógicos fuzzy, unidades computacionais que formam a estrutura das redes neurofuzzy estática e recorrente a serem detalhadas no decorrer deste capítulo.

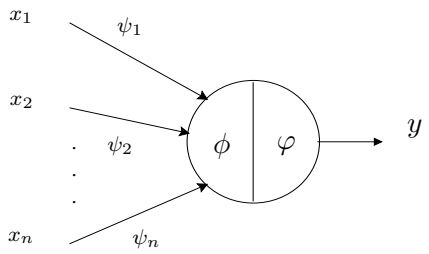
4.3 Neurônios Lógicos Fuzzy

Os neurônios lógicos fuzzy produzem um mapeamento $[0,1]^n \rightarrow [0,1]$, realizando uma operação lógica sobre as entradas, utilizando para isso $t-$normas e $s-$normas (ver Seção 3.2.3).

Um método de treinamento para neurônios fuzzy baseado em operações lógicas é apresentado em (Gupta & QI 1992). Em (Yamakawa et al. 1992), um modelo de neurônio denominado de *neo fuzzy neuron*, assim como o correspondente algoritmo de aprendizado é descrito. Neste modelo, cada sinapse caracteriza-se por um conjunto de regras fuzzy associado a pesos que atuam como conseqüentes (números reais) das regras. Este modelo foi aplicado a problemas de identificação de sistemas não lineares (Caminhas 1997).

A possibilidade de uma maior aproximação entre os sistemas artificiais e os sistemas nervosos biológicos, constitui uma das principais motivações para o desenvolvimento de neurônios lógicos fuzzy (Figueiredo 1997). Em (Figueiredo & Gomide 1997) é proposta uma rede neurofuzzy baseado em um modelo genérico de neurônio. Este modelo de neurônio, ilustrado na Figura 4.1, é composto pelas funções sinápticas ψ_i associadas às n componentes do vetor de entrada, a função de agregação ϕ e uma função de codificação φ não necessariamente linear para o cálculo da saída do neurônio.

Este modelo define uma classe de neurônios fuzzy bastante geral, pois dependendo da



$$y = \varphi(\phi(\psi_1(x_1), \ldots, \psi_n(x_n)))$$

Figura 4.1: Neurônio fuzzy genérico.

escolha conveniente das funções (operadores) é possível obter diferentes propostas, inclusive, o modelo de neurônio artificial clássico.

Dois tipos de neurônios lógicos fuzzy são utilizados neste trabalho: o neurônio lógico AND e o neurônio lógico OR, sendo estes casos particulares do modelo genérico descrito.

4.3.1 Neurônios Lógicos AND e OR

O neurônio lógico do tipo AND, é obtido escolhendo, para o modelo genérico de neurônio fuzzy, a φ como a função identidade, ϕ como uma $t-$norma e ψ_i como uma $s-$norma.

O neurônio AND efetua uma combinação das entradas $\mathbf{x}$ através do operador OR ($s-$norma) sobre as conexões do neurônio. Os resultados são agregados utilizando o operador lógico AND ($t-$norma), isto é:

$$y = AND[(x_1 \; OR \; w_1), \ldots, (x_i \; OR \; w_i), \ldots, (x_n \; OR \; w_n)]$$

Usando a notação das normas triangulares, o neurônio lógico AND, ilustrado na Figura 4.2, pode ser escrito como:

$$y = \underset{i=1}{\overset{n}{\mathbf{T}}} [x_i \, \mathbf{s} \, w_i] \tag{4.1}$$

onde $\mathbf{x} = [x_1, \; x_2, \cdots, \; x_n]$ é a entrada, $\mathbf{w} = [w_1, \; w_2, \cdots, \; w_n]$ são os pesos das conexões e $\mathbf{T}$ é uma $t-$norma.

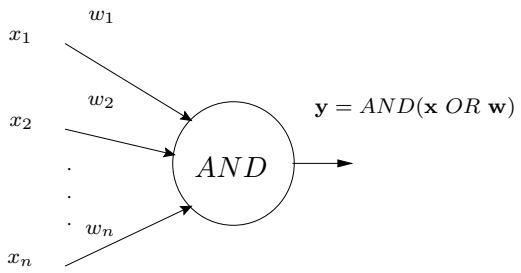


Figura 4.2: Neurônio lógico fuzzy AND.

O neurônio lógico de agregação do tipo OR, ilustrado na Figura 4.3, é obtido como resultado da escolha no modelo genérico de neurônio fuzzy de φ como a função identidade, ϕ como uma $s-$norma e ψ_i como uma $t-$norma (ver Figura 4.1).

Este neurônio lógico fuzzy constitui uma estrutura dual em relação ao neurônio AND. Os sinais de entrada são combinados através do operador lógico AND, e os resultados agregados utilizando o operador lógico OR, isto é:

$$y = OR[(x_1\ AND\ w_1), \ldots, (x_i\ AND\ w_i), \ldots, (x_n\ AND\ w_n)].$$

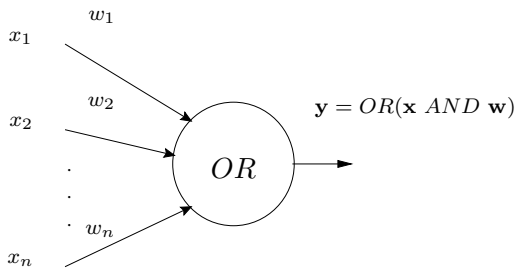


Figura 4.3: Neurônio lógico fuzzy OR.

Usando a notação de normas triangulares, o neurônio OR é definido como:

$$y = \underset{i=1}{\overset{n}{\mathbf{S}}} [w_i\ \mathbf{t}\ x_i]. \tag{4.2}$$

Os neurônios lógicos apresentados realizam um mapeamento não linear estático dentro de um hipercubo unitário, fazendo que todos os sinais de entrada e saída, bem como, as conexões do neurônio sejam codificadas no intervalo $[0, 1]$. Isto implica que os valores da

saída y, para todas as possíveis entradas, cobrem um subconjunto do intervalo unitário, mas não necessariamente o intervalo inteiro. Ou seja, para o neurônio OR, os valores de y estão no intervalo $[0, \overset{n}{\underset{i=1}{\mathbf{S}}} w_i]$, já os valores de y para o neurônio AND cobrem o intervalo $[\overset{n}{\underset{i=1}{\mathbf{T}}} w_i, 1]$ (Pedrycz & Gomide 1998).

Para que estes neurônios lógicos sejam capazes de estabelecer relações dinâmicas entre as entradas e a saída, é necessário considerar conexões de realimentação. Neurônios lógicos fuzzy recorrentes são definidos a seguir.

4.3.2 Neurônios Lógicos Fuzzy Recorrentes

Um neurônio lógico com recorrência local considera a entrada $\mathbf{x}$ e os respectivos pesos $\mathbf{w}$ junto como uma entrada que é a realimentação da sua própria saída y ponderada pelo respectivo peso de recorrência r.

Sejam $\mathbf{x}(t)$ e $y(t)$ a entrada e saída respectivamente, no neurônio no instante t, $\mathbf{w}$ os pesos relacionados a entrada e r o peso do sinal de recorrência $y(t-1)$. Assim, o neurônio lógico com recorrência local do tipo AND é definido como:

$$y(t) = AND[(x_1(t)\ OR\ w_1), \ldots, (x_i(t)\ OR\ w_i), \ldots, (x_n(t)\ OR\ w_n)]\ AND\ [y(t-1)\ OR\ r]$$

Utilizando as normas triangulares, esta definição pode ser re-escrita como:

$$y(t) = \overset{n+1}{\underset{i=1}{\mathbf{T}}} [w_i\ \mathbf{s}\ x_i(t)] \tag{4.3}$$

onde $w_{n+1} = r$ e $x_{n+1}(t) = y(t-1) = q^{-1}y(t)$, sendo q^{-1} o operador atraso. Esta equação pode ser referenciada como uma equação à diferenças fuzzy. A Figura 4.4 ilustra o neurônio lógico AND recorrente.

Dado um conjunto de M neurônios lógicos estáticos, acrescentando laços entre todas as unidades lógicas, estes passam a ser denominados neurônios lógicos com recorrência global. A Figura 4.5 ilustra este tipo de neurônio:

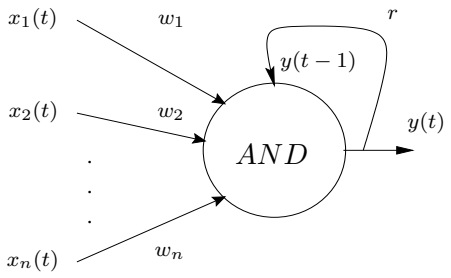


Figura 4.4: Neurônio lógico AND com recorrência local.

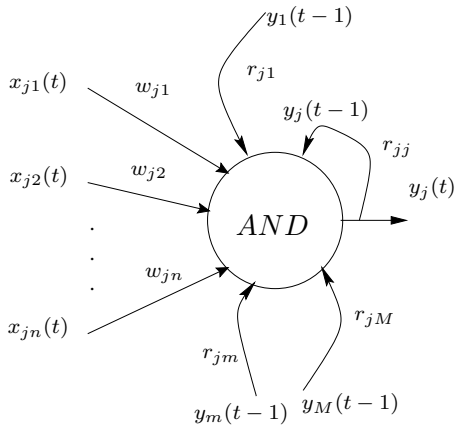


Figura 4.5: Neurônio lógico AND com recorrência global.

sendo r_{ji} o peso da conexão de realimentação entre o $i-$ésimo e o $j-$ésimo neurônio AND, x_{ji} a $i-$ésima entrada ao $j-$ésimo neurônio AND, $y_j(t)$ a saída do $j-$ésimo neurônio e $y_j(t-1) = q^{-1}y_j(t)$, $i, j = 1, \ldots, M$ e q^{-1} é o operador atraso.

Utilizando normas triangulares, a dinâmica do neurônio com recorrência global pode ser definida como segue:

$$y_j(t) = \overset{n+M}{\underset{i=1}{\mathbf{T}}} [w_{ji}\, \mathbf{s}\, x_{ji}(t)] \tag{4.4}$$

sendo $w_{j\ n+l} = r_{jl}$ e $x_{j\ n+l} = y_l(t-1)$, para $l = 1, \ldots, M$.

De forma análoga, neurônios lógicos com recorrência interna local ou global do tipo OR podem ser definidos e, assim como nos neurônios clássicos, uma função de ativação não linear pode ser acrescentado na saída dos neurônios lógicos.

Os neurônios lógicos estáticos e recorrentes distribuídos em camadas, constituem estruturas que formam a proposta deste trabalho detalhadas a seguir.

4.4 Rede Neurofuzzy com Recorrência Global (RNFR-Glob)

A estrutura da rede neurofuzzy com recorrência interna global é ilustrada na Figura 4.6 (Luna et al. 2002). Esta estrutura neurofuzzy é composta por duas partes: um sistema de inferência fuzzy e uma rede neural clássica.

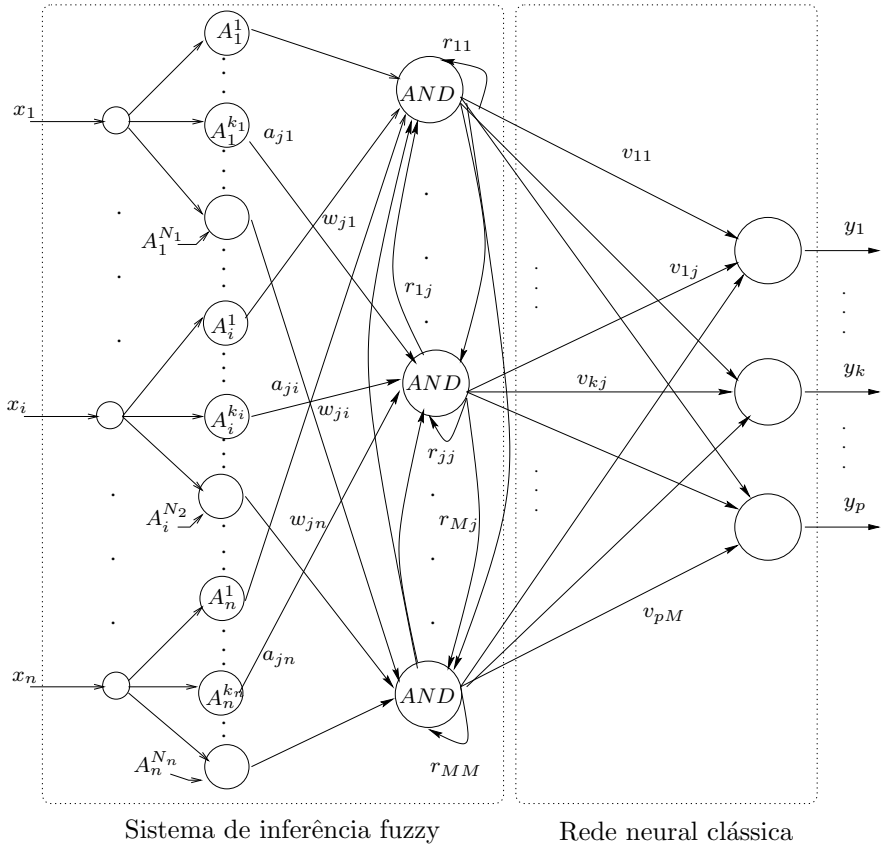


Figura 4.6: Rede neurofuzzy com recorrência interna global (**RNFRGlob**).

O sistema de inferência fuzzy é composto pelas camadas de entrada e intermediária. A camada de entrada consiste de neurônios cujas funções de ativação são as funções de pertinência dos conjuntos fuzzy que formam a partição do espaço de entrada. Para cada componente $x_i(t)$

do vetor de entrada $n-$dimensional $\mathbf{x}(t)$ existem N_i conjuntos fuzzy $A_i^{k_i}, k_i = 1, \ldots, N_i$ cujas funções de pertinência são as correspondentes funções de ativação dos neurônios da camada de entrada. A variável t denota o tempo discretizado, isto é, $t = 1, 2, \ldots$ e será omitida no decorrer deste trabalho para simplificar a notação.

Deste modo, os graus de pertinência associados aos padrões de entrada são:

$$a_{ji} = \mu_{A_i^{k_i}}(x_i)$$

com $i = 1, \ldots, n$ e $j = 1, \ldots, M$; onde M é o número de neurônios da segunda camada.

Os neurônios da camada intermediária são neurônios lógicos com recorrência interna global do tipo AND (Figura 4.5), cujas entradas a_{ji} são ponderadas pelos pesos w_{ji} e as conexões de realimentação são ponderadas pelos pesos $r_{jl}, l = 1, \ldots, M$.

A implementação dos conectivos de conjuntos fuzzy envolvem normas triangulares, isto é, os operadores AND e OR são implementados através de $t-$normas e $s-$normas. Dado que as normas trabalham só com valores no intervalo $[0, 1]$, tanto os pesos w_{ji}, r_{jl} como os graus de pertinência a_{ji}, devem pertencer a este intervalo, produzindo assim, um mapeamento não-linear no hipercubo unitário.

A estrutura da rede neurofuzzy codifica de forma implícita um conjunto $R = \{R_j, j = 1, \ldots, M\}$ de regras do tipo *Se-Então* tal que:

R_j : Se (x_1 é $A_1^{k_1}$ com certeza w_{j1}) ... AND (x_i é $A_i^{k_i}$ com certeza w_{ji}) ...
AND (x_n é $A_n^{k_n}$ com certeza w_{jn}) AND $z_{j1}(t-1)$ com certeza r_{j1} ...
AND $z_{jl}(t-1)$ com certeza r_{jl} ... AND $z_{jM}(t-1)$ com certeza r_{jM}
então, z é $z_j(t)$.

onde:

$$z_j = \mathop{\mathbf{T}}_{i=1}^{n+M} (w_{ji} \,\mathbf{s}\, a_{ji}) \tag{4.5}$$

Portanto, existe uma similaridade entre a estrutura que a primeira e a segunda camada constituem e um sistema de inferência fuzzy associado.

A segunda parte da estrutura consiste de uma rede neural clássica, composta por um neurônio não recorrente como mostra a Figura 4.7. A saída y_k é a ativação causada pelas das entradas z_j ponderadas pelos pesos $v_{kj}, j = 1, \ldots, M$ e $k = 1, \ldots, p$.

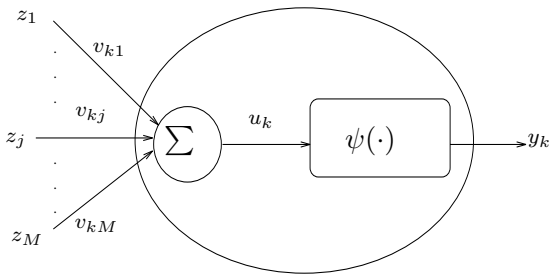


Figura 4.7: Neurônio clássico estático.

Esta rede neural clássica codifica uma função de agregação para a geração da saída do sistema.

A dinâmica da rede neurofuzzy pode ser resumida da seguinte maneira:

1. $x_1, \ldots, x_i, \ldots, x_n$ são os n componentes do vetor de entrada $\mathbf{x}$;

2. N_i é o número de conjuntos fuzzy que constitui a partição do universo da $i-$ésima entrada;

3. $A_i^{k_i}$ é o k_i-ésimo conjunto fuzzy associado à partição do universo do $i-$ésimo componente de $\mathbf{x}$, com $k_i = 1, \ldots, N_i$;

4. c_{ir} são os centros da $i-$ésima componente de $\mathbf{x}$, com $r = 1, \ldots, N_i$. Estes centros definirão as funções de pertinência associadas aos conjuntos fuzzy $A_i^{k_i}$.

5. A variável j indexa os neurônios lógicos AND. Para a estrutura apresentada na Figura 4.6, j é determinada utilizando a seguinte equação:

$$j = f(K) = k_n + \sum_{i=2}^{n}(k_{(n-i+1)} - 1)\left(\prod_{\tau=1}^{i-1} N_{(n+1-\tau)}\right) \tag{4.6}$$

onde, $K = (k_1,\ \ldots,\ k_i,\ \ldots,\ k_n)$, sendo k_i a regra ativada pelo componente x_i de $\mathbf{x}$.

A modo de exemplo, suponha-se uma rede neurofuzzy recorrente com duas entradas ($n = 2$) e uma saída ($p = 1$), como se apresenta na Figura 4.8.

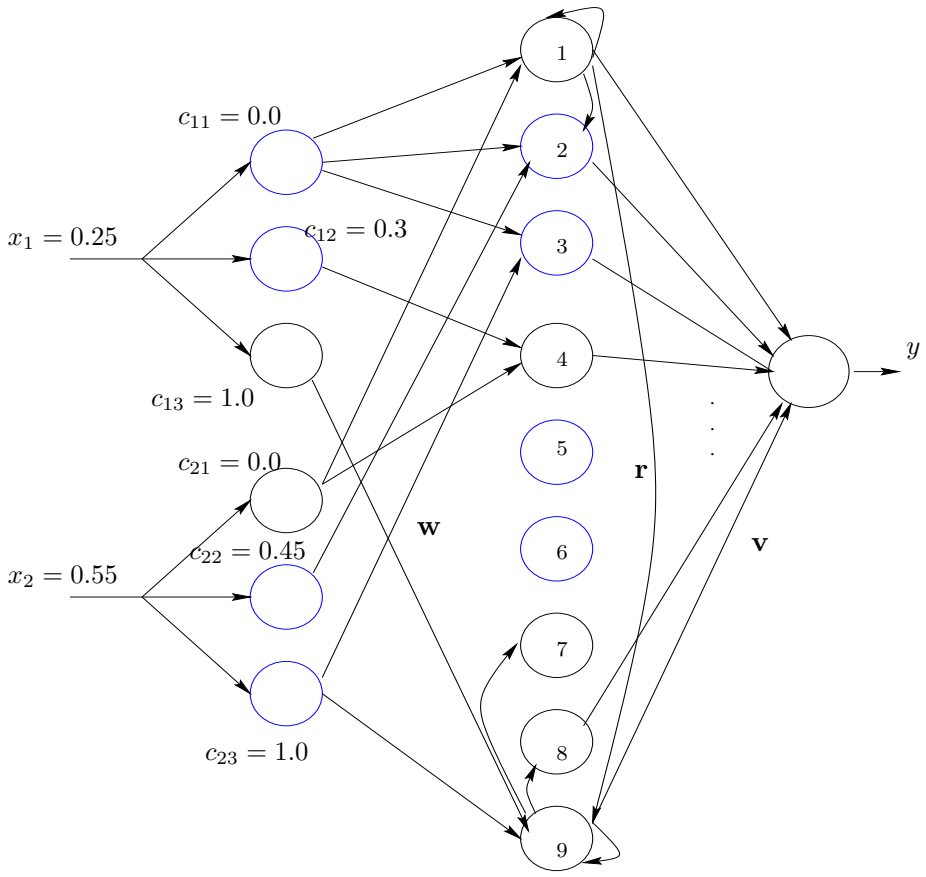


Figura 4.8: Exemplo para determinar os neurônios AND ativos de rede neurofuzzy recorrente.

A partição do universo para cada entrada x_i é composta de três conjuntos fuzzy com funções de pertinência triangulares como se ilustra na Figura 4.9, ou seja, $N_1 = N_2 = 3$. Sejam os centros definidos por $\mathbf{c_1} = [0.0\ 0.3\ 1.0]$ e $\mathbf{c_2} = [0.0\ 0.45\ 1.0]$.

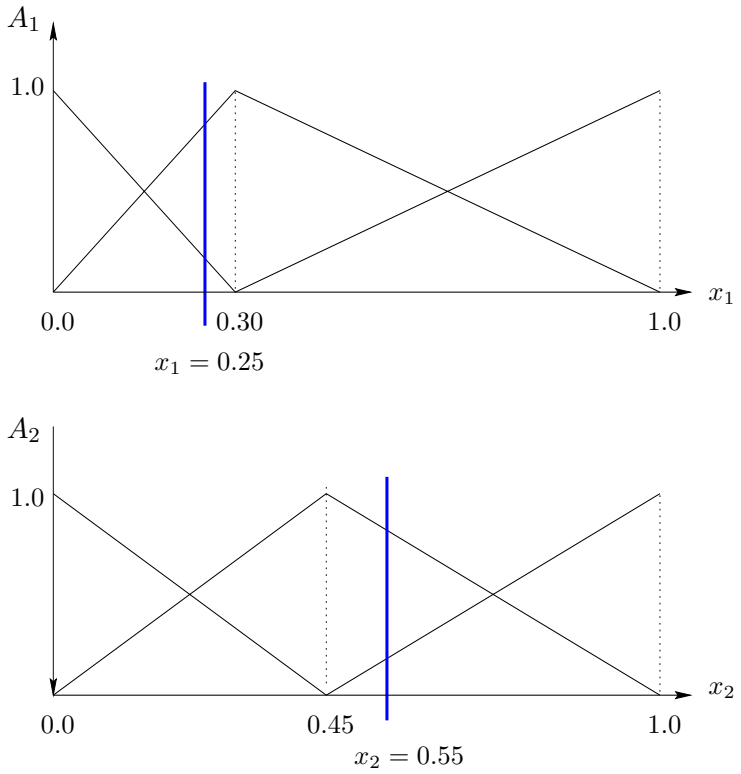


Figura 4.9: Partição do espaço de entrada para o exemplo de rede neurofuzzy recorrente ilustrado na Figura 4.8.

Seja para um instante de tempo t dado, o padrão de entrada igual a $\mathbf{x} = [0.25\ 0.55]$. Dado que $c_{11} < x_1 < c_{12}$ e $c_{22} < x_2 < c_{23}$, os conjuntos fuzzy ativos serão A_1^1 e A_1^2 para x_1 e A_2^2 e A_2^3 para x_2. Assim, os conjuntos fuzzy ativos para a primeira entrada serão os conjuntos [1 2] e para a segunda entrada, os conjuntos fuzzy ativos serão os conjuntos [2 3].

Combinando estes indices, tem-se para este caso, um total de $e^2 = 4$ vetores K definidos da seguinte maneira:

$$K^1 = [1\ 2] \qquad K^2 = [1\ 3]$$

$$K^3 = [2\ 2] \qquad K^4 = [2\ 3]$$

Cada vetor K gerado, representa um neurônio AND ativo na camada intermediaria. Desta forma, utilizando a expressão (4.6) e cada um dos vetores K^1, K^2, K^3, K^4, os

neurônio AND ativos serão os neurônios 2, 3, 5 e 6 respectivamente.

6. $a_{ji} = \mu_{A_i^{k_i}}(x_i)$ é o grau de pertinência de x_i no conjunto fuzzy $A_i^{k_i}$, sendo a_{ji} a $i-$ésima entrada para o neurônio j da camada intermediária;

7. z_j é a $j-$ésima saída da camada intermediária, dada pela expressão (4.5);

8. y_k é a $k-$ésima saída da rede, dada por:

$$y_k = \psi(u) = \psi\left(\sum_{j=1}^{M}(v_{kj}z_j)\right) \tag{4.7}$$

onde, $\psi : \Re^n \rightarrow [0,1]$ é uma função contínua e monotonicamente crescente que, neste trabalho, sem perda de generalidade, está sendo considerada como: $\psi(u) = 1/(1 + \exp(-u))$. Assim, $y_k \in [0,1]$.

9. w_{ji} é o peso entre o $j-$ésimo neurônio and e o $i-$ésimo neurônio da camada de entrada;

10. v_{kj} é o peso entre a saída y_k da rede e o $j-$ésimo neurônio AND;

11. r_{jl} é o peso da conexão recorrente entre o $l-$ésimo neurônio AND e o $j-$ésimo neurônio AND da mesma camada;

4.5 Rede Neurofuzzy com Recorrência Local (RNFR-Loc)

Considerando que as conexões entre neurônios AND distintos não existam, a rede neurofuzzy recorrente da Figura 4.6, torna-se uma estrutura com recorrência interna local (Luna, Ballini & Gomide 2003b), constituída por neurônios lógicos como na Figura 4.4. A estrutura da rede neurofuzzy com recorrência interna local é ilustrada na Figura 4.10.

Os pesos da conexão de realimentação r_j, associado ao $j-$ésimo neurônio lógico AND, devem pertencer ao intervalo unitário pois o processamento do neurônio lógico baseia-se em normas triangulares.

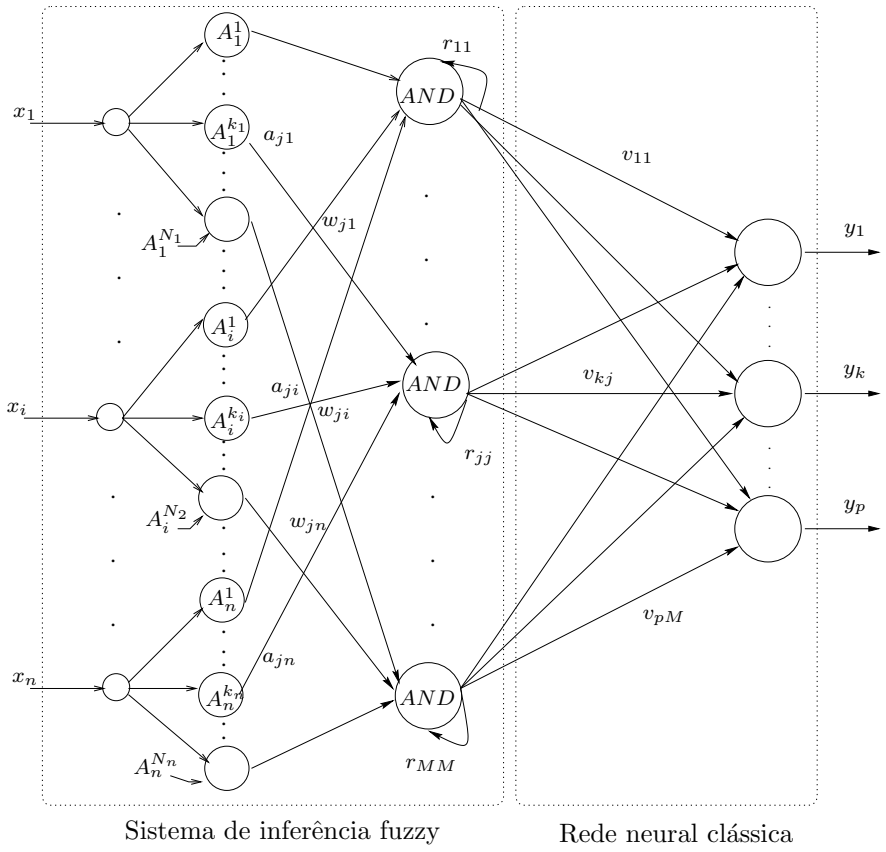


Figura 4.10: Rede neurofuzzy com recorrência interna local (**RNFRLoc**).

Os neurônios ativos são identificados utilizando a expressão (4.6) e a saída z_j é definida como:

$$z_j = \mathop{\mathbf{T}}_{i=1}^{n+1} (w_{ji} \,\mathbf{s}\, a_{ji}) \tag{4.8}$$

onde $w_{in+1} = r_j$ e $a_{in+1} = z_j(t-1) = q^{-1}z_j(t)$.

A estrutura recorrente **RNFRLoc** codifica um conjunto de regras fuzzy do tipo *se-então* $R = \{R_j, j = 1, \ldots, M\}$ da forma seguinte:

R_j : Se (x_1 é $A_1^{k_1}$ com certeza w_{1j}) $\ldots$ AND (x_i é $A_i^{k_i}$ com certeza w_{ji}) $\ldots$
AND (x_n é $A_n^{k_n}$ com certeza w_{jn}) AND $z_j(t-1)$ com certeza r_j
então, z é $z_j(t)$.

sendo z_j definido por (4.8). Estas regras formam um sistema de inferência fuzzy recorrente, onde as saídas z_j atuam como variáveis internas.

A Tabela 4.2, apresenta uma analogia entre o modelo proposto em (Bersini & Gorrini 1994) e os modelos aqui propostos, sendo R_j (ver definição acima) as regras fuzzy análogas às regras que formam os sistemas de inferência definidos pelas expressões (3.22) e (3.23). Esta analogía é válida tanto para **RNFRGlob** como para **RNFRLoc**.

Tabela 4.2: Analogia entre sistemas neurofuzzy recorrentes propostos e sistemas fuzzy recorrentes.

Análise qualitativa	
Sistema fuzzy recorrente	**Sistemas neurofuzzy recorrente**
2 mecanismos de inferência, um para Φ (ver Figura 3.6) e o segundo para y	1 único mecanismo de inferência
Mecanismos de inferência Takagi-Sugeno via neurônios Π	Mecanismo de inferência utilizando neurônios lógicos AND
Graus de certeza para as componentes dos antecedentes $w_{ji} = 1$	Graus de certeza w_{ji} para as funções de pertinência ativas no intervalo $[0, 1]$
Funções de pertinência triangulares, isósceles e de parâmetros ajustáveis	Funções de pertinência triangulares, não uniformemente distribuídas e fixas
Cálculo da saída via método do centro de gravidade	Cálculo da saída via soma ponderada e função de ativação
Ajuste das funções de pertinência	Ajuste dos graus de certeza do mecanismo de inferência e dos pesos da saída
Método de treinamento via retropropagação	Aprendizado via combinação de retropropagação e reforço associativo

4.6 Rede Neurofuzzy Estática (RNFEst)

A estrutura da rede neurofuzzy estática **RNFEst** é ilustrada na Figura 4.11.

As regras R_j, $j = 1, \ldots, M$ do sistema de inferência fuzzy estático codificadas na estrutura da rede neurofuzzy **RNFEst**, tem a seguinte forma:

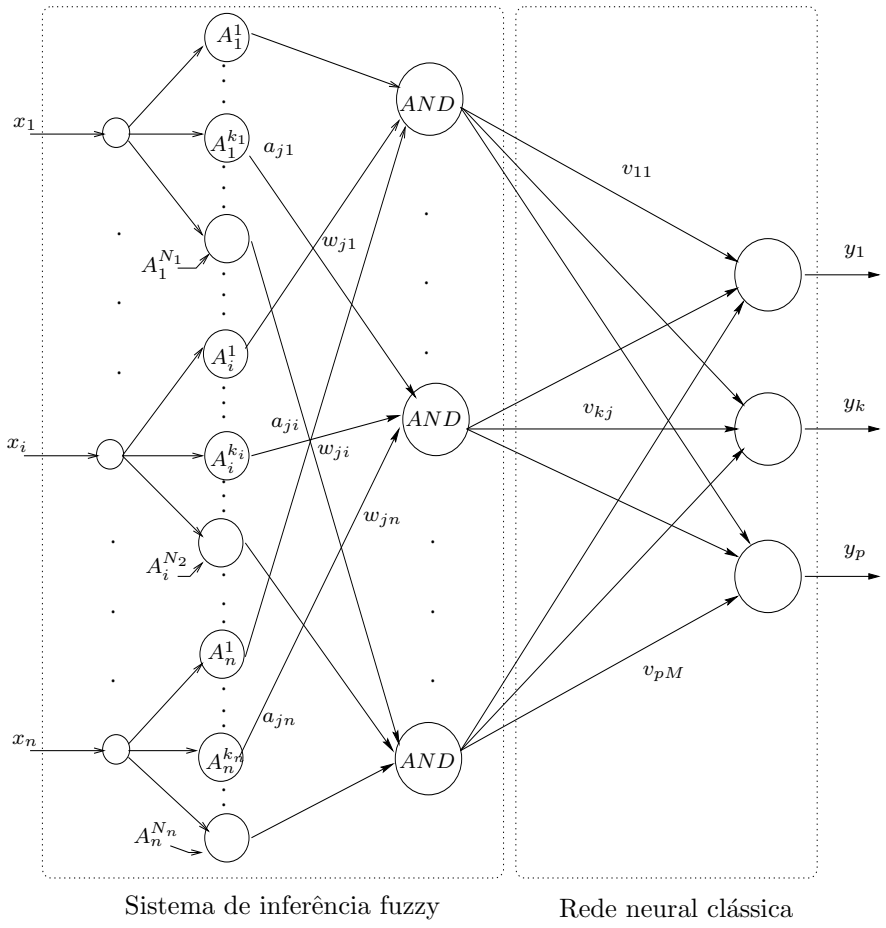


Figura 4.11: Rede neurofuzzy estática (**RNFEst**).

$$R_j \;:\; \text{Se } (x_1 \text{ é } A_1^{k_1} \text{ com certeza } w_{j1}) \ldots AND\ (x_i \text{ é } A_i^{k_i} \text{ com certeza } w_{ji}) \ldots$$
$$AND\ (x_n \text{ é } A_n^{k_n} \text{ com certeza } w_{jn}) \text{ então, } z \text{ é } z_j.$$

sendo z_j a saída do $j-$ésimo neurônio lógico dado por:

$$z_j = \mathop{\mathbf{T}}_{i=1}^{n} (w_{ji}\, \mathbf{s}\, a_{ji}) \tag{4.9}$$

Dado que todas as conexões entre os neurônios lógicos não existem (não são consideradas), os neurônios que constituem a camada intermediária adotam o modelo da Figura 4.2.

As redes neurofuzzy apresentadas produzem uma partição do espaço de entrada $n-$dimensional em M regiões fuzzy, sendo cada uma destas regiões governadas por uma regra $Se -$

Então. Isto é, o antecedente de cada regra define uma região fuzzy, enquanto o conseqüente define a saída do sistema para esta região. Além disso, estas arquiteturas tem como vantagens a geração automática da topologia da rede, flexibilidade quanto à utilização de diversas normas triangulares e a possibilidade de extração de regras diretamente da topologia (Caminhas et al. 1999) e (Iyoda 2000).

4.7 Processo de Aprendizagem

Assim como as redes neurais clássicas, as redes neurofuzzy recorrentes e estática apresentadas, possuem capacidade de aprendizado. O algoritmo de aprendizado proposto está baseado em dois métodos: o método do gradiente da função erro (Hush & Horne 1993), (Rumelhart & McClelland 1986) para os pesos da camada de saída e o método de aprendizado por reforço associativo (Barto & Jordan 1987), (Caminhas 1997) para os pesos da camada intermediária.

Os passos que do algoritmo de treinamento proposto são enumerados a seguir.

Algoritmo de Aprendizado

1. Gerar das funções de pertinência;

2. Inicializar dos pesos;

3. Até que a condição de parada não seja satisfeita, fazer:

    3.1 Apresentar um padrão $\mathbf{x}$ à rede, geralmente escolhido aleatoriamente;

    3.2 Efetuar a fuzzificação;

    3.3 Determinar os neurônios AND ativos e calcular a saída da rede $\mathbf{y}$;

    3.4 Atualizar os pesos w_{ji}, v_{kj} e r_{jl} para a rede com recorrência global **RNFRGlob**, (r_j para a rede com recorrência local **RNFRLoc**);

    3.5 Testar a condição de parada (máximo erro permitido ou número máximo de iterações).

A seguir, cada um dos passos do algoritmo acima será detalhado.

4.7.1 Geração das Funções de Pertinência

As funções de pertinência assumidas para os três modelos neurofuzzy são funções triangulares, podendo ser adotadas em geral outras formas como as definidas na Seção 3.2.2. Estas funções de pertinência, normais e complementares, são caracterizadas por três parâmetros x_{imin}, x_{imax} e c_{ir}, sendo x_{imin} e x_{imax} os valores mínimo e máximo associado ao $i-$ésimo componente do vetor de entrada $\mathbf{x}$ e c_{ir}, o centro da função de pertinência, com $r = 1, \ldots, N_i$, onde N_i é o número de conjuntos nebulosos que compõem a partição do universo do $i-$ésimo componente de $\mathbf{x}$.

Dois tipos de partições são utilizadas: partição uniforme, ilustrada na Figura 4.12 e partição não uniforme (Figura 4.13).

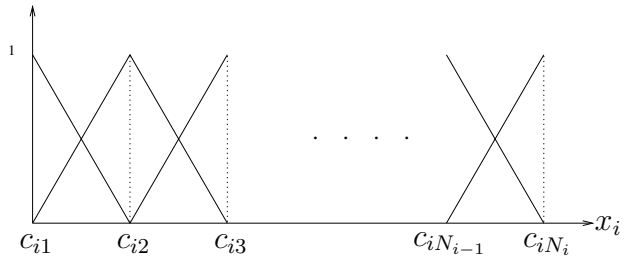


Figura 4.12: Partição uniforme.

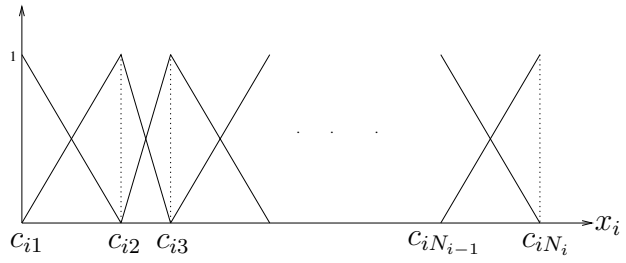


Figura 4.13: Partição não uniforme.

Dado o número de conjuntos nebulosos (N_i), a partição uniforme é gerada da seguinte forma:

$$\triangle_i = \frac{x_{imax} - x_{imin}}{N_i - 1} \qquad i = 1, \ldots, n \tag{4.10}$$

Os centros c_{ir} são definidos como:

$$c_{ir} = x_{imin} + (r-1) \cdot \triangle_i \qquad\qquad r = 1, \ldots, N_i \tag{4.11}$$

Quando ocorre uma concentração de dados em uma determinada região do espaço de entrada, pode ser interessante assumir uma partição não uniforme, resultando em um menor número de centros em regiões onde a concentração de informação seja baixa (Caminhas et al. 1999). Neste caso, técnicas de agrupamento para determinar os centros c_{ir}, com $r = 1, \ldots, N_i$ são necessárias. O algoritmo de agrupamento utilizado neste trabalho é baseado na rede neural auto-organizada de Kohonen (Kohonen 1982), com algoritmo de treinamento LVQ (*Learning Vector Quantization*), proposto em (Caminhas 1997).

A rede auto-organizada utilizada para a determinação dos centros c_{ir} é ilustrada na Figura 4.14. Os pesos da rede são os centros associados as funções de pertinência. O número, inicialmente sobre-estimado, de unidades na saída da rede N_i^0 é corrigido via aprendizagem. Este parâmetro fornece uma estimativa do número de centros para a partição correspondente a x_i.

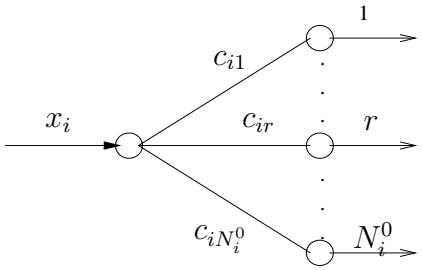


Figura 4.14: Rede auto-organizada para determinar os centros das funções de pertinência durante a geração de partições não uniformes.

O algoritmo de treinamento proposto é um método de treinamento não supervisionado e competitivo, onde cada neurônio possui um índice de desempenho sendo que, para cada iteração, somente o neurônio vencedor é ajustado. No final do treinamento, os neurônios com menor índice de desempenho são excluídos, resultando um número final de centros menor ou igual a N_i^0.

O procedimento para a geração automática da partição não uniforme é apresentado a seguir.

Algoritmo de Agrupamento via Rede Neural Auto-Organizada

1. Inicialização:

   1.1 Dado N_i^0, os pesos iniciais c_{ir} são inicializados considerando uma partição uniforme para $r = 1, \ldots, N_i^0$, ou seja, os centros são inicializados utilizando 4.10 e 4.11, para $r = 1, \ldots, N_i^0$. Esta forma de inicialização dos pesos contribui na aceleração da convergência.

   1.2 Sejam $id_i(r)$ os índices de desempenho associado a cada neurônio. No início do treinamento, os índices de desempenho são todos nulos, ou seja:

$$id_i(r) = 0, \qquad r = 1, \ldots, N_i^0$$

2. Até que $|c_{ir}(t+1) - c_{ir}(t)| \leq \varepsilon, \forall r$ fazer:

   2.1 Apresentar um padrão de entrada x_i e atualizar o peso do neurônio vencedor da seguinte maneira:

$$c_{iL}(t+1) = c_{iL}(t) + \alpha(t)[x_i - c_{iL}(t)] \tag{4.12}$$

   sendo L o índice do neurônio vencedor, cujo peso da conexão possui o valor mais próximo do padrão de entrada, isto é:

$$L = \arg\{ \min_r |x_i - c_{ir}| \} \tag{4.13}$$

   2.2 Ajustar o valor do passo $\alpha(t)$, onde $\alpha(t)$ é uma função decrescente;

   2.3 Atualizar o índice de desempenho do neurônio vencedor:

$$id_i(L) = id_i(L) + 1 \tag{4.14}$$

3. Excluir os neurônios que conseguiram um índice de desempenho baixo, ou seja $id_i < \delta$, sendo δ um limiar inteiro positivo. Seja Nne_i o número de neurônios excluídos para a $i-$ésima componente x_i. O número final e centros N_i será:

$$N_i = N_i^0 - Nne_i \tag{4.15}$$

Uma vez determinados os centros $c_{ir}, i = 1, \ldots, n$ e $r = 1, \ldots, N_i$, as funções de pertinência $\mu_{A_i^r}(x_i)$ para cada padrão de entrada $\mathbf{x}$ são calculados da seguinte forma:

$$\mu_{A_i^r}(x_i) = \begin{cases} \alpha_{esq} \cdot (x_i - c_{ir}) + 1, & c_{ir-1} \leq x_i < cir \\ \alpha_{dir} \cdot (x_i - c_{ir}) + 1, & c_{ir} < x_i \leq c_{ir+1} \\ 0, & \text{caso contrário.} \end{cases} \tag{4.16}$$

onde $\alpha_{esq} = \frac{1}{c_{ir} - c_{ir-1}}$ e $\alpha_{dir} = \frac{1}{c_{ir} - c_{ir+1}}$, $i = 1, \ldots, n$ e $r = 1, \ldots, N_i$.

4.7.2 Inicialização dos Pesos

Os pesos da camada de saída v_{kj} são inicializados aleatoriamente com valores no intervalo $[-1, 1]$, $k = 1, \ldots, p$ e $j = 1, \ldots, M$.

Os pesos da recorrência r_j para a rede **RNFRLoc** e r_{lj} para a rede **RNFRGlob** assim como os pesos w_{ji}, são inicializados com valores aleatórios no intervalo $[0, 1]$, para $i = 1, \ldots, n$, $j = 1, \ldots, M$ e $l = 1, \ldots, M$. Testes inicializando estes pesos com zeros e uns foram feito. Os resultados de simulação não foram influenciados por estas inicializações.

4.7.3 Determinação dos Neurônios AND Ativos

Para cada padrão de entrada $\mathbf{x}$ apresentado à rede (**RNFRGlob**, **RNFRLoc** ou **RNFEst**), existe no máximo dois conjuntos fuzzy com graus de pertinência diferentes de zero, denominados conjuntos ativos que, por sua vez, definem os neurônios AND ativos. Assim, de M neurônios AND na segunda camada, no máximo 2^n estarão ativos, sendo estes, identificados da seguinte maneira:

Dado um padrão de entrada $\mathbf{x} = [x_1, \ldots, x_i, \ldots, x_n]$, seja $K^1 = (k_1^1, \ldots, k_i^1, \ldots, k_n^1)$ o vetor cujos componentes são os índices da primeira função de pertinência diferente de zero para cada componente de $\mathbf{x}$. Seja $K^2 = (k_1^2, \ldots, k_i^2, \ldots, k_n^2)$ um vetor tal que:

$$k_i^2 = \begin{cases} k_i^1 + 1 & \text{,se } \mu_{A_i^{k_i^1}(x_i)} \neq 1 \\ k_i^1 & \text{,caso contrário} \end{cases} \tag{4.17}$$

onde, $\mu_{A_i^{k_i^1}}(x_i)$ representa o grau de pertinência de x_i ao subconjunto fuzzy indexado por k_i^1.

O número de neurônios AND ativos Na é 2^{Pa} sendo $2^{Pa} \leq 2^n$, onde Pa é o número de elementos tal que $k_i^1 \neq k_i^2$, para $i = 1, \ldots, n$. Os neurônios AND ativos são determinados utilizando a expressão (4.6).

4.7.4 Fuzzificação

Nesta etapa, os graus de pertinência para os subconjuntos ativos definidos em K^1 para os padrões de entrada são calculados.

No caso em que $k_i^1 \neq k_i^2$, tem-se:

$$\mu_{k_i^2}(x_i) = 1 - \mu_{k_i^1}(x_i)$$

isto devido à complementaridade das funções de pertinência. Assim, somente é necessário efetuar o cálculo de $2n$ graus de pertinência.

4.7.5 Atualização dos Pesos

O processo de atualização de pesos das conexões das redes neurofuzzy propostas é baseado no método do gradiente da função erro e no método de treinamento por reforço associativo proposto em (Barto & Jordan 1987), (Caminhas 1997). O método do gradiente é utilizado para atualizar os pesos da camada de saída. Os pesos das conexões que formam o sistema de inferência fuzzy são atualizados através do método de treinamento por reforço associativo.

O primeiro passo é avaliar a saída da rede para um dado padrão de entrada $\mathbf{x}$ utilizando primeiramente para o cálculo das saídas dos neurônios da camada intermediária a Equação

(4.9) para a rede **RNFEst**, a Equação (4.8) para a rede **RNFRLoc**, a Equação (4.5) para a rede **RNFRGlob** e para o cáculo das saídas, a Equação (4.7), isto é, calcular as saídas de cada neurônio em cada camada para cada uma das redes.

O objetivo do processo de treinamento (supervisionado) é minimizar o erro quadrático médio entre a saída atual da rede $\widehat{\mathbf{y}}$ e a saída desejada $\mathbf{y}$, para cada padrão de entrada, isto é, minimizar:

$$\epsilon = \frac{1}{p}\sum_{k=1}^{p}(y_k - \widehat{y}_k)^2 \tag{4.18}$$

onde, $\widehat{y_k}$ é o valor da saída do $k-$ésimo neurônio e y_k é a correspondente saída desejada para o padrão $\mathbf{x}$, $k = 1, \ldots, p$.

Assim, se v_{kj} é um peso conectado à $k-$ésima unidade de saída, tem-se:

$$\Delta v_{kj} = \eta(y_k - \widehat{y}_k)\psi'(u_k)z_j \tag{4.19}$$

onde, $\psi'(u_k) = \psi(u_k)(1 - \psi(u_k))$ é a derivada da função de ativação avaliada em u_k, sendo $u_k = \sum_{i=1}^{M} v_{kj}z_j$ e η a taxa de aprendizado.

O processo de atualização dos pesos da camada intermediária é baseado em duas abordagens: método de treinamento por reforço associativo proposto por A.G. Barto e M.I. Jordan em (Barto & Jordan 1987) e o método de ajuste de pesos de uma rede neurofuzzy estática aplicada a classificação de padrões proposta em (Caminhas 1997) e (Caminhas et al. 1999). Ambas abordagens tem como base para o ajuste dos pesos um mecanismo de recompensa e punição.

Para Caminhas (Caminhas 1997), a estrutura neurofuzzy possui uma dualidade com um sistema de inferência fuzzy baseado em regras do tipo $Se - Então$, onde cada regra possui um grau de certeza. Quando um padrão é classificado corretamente pela rede, os graus de certeza associados às regras ativadas são aumentados (recompensa), caso contrário, estes são diminuidos (punição).

Deve-se observar que, se uma regra é ativada corretamente um maior número de vezes,

o grau de certeza desta será cada vez maior. No caso da rede neurofuzzy estática proposta (Figura 4.11), as regras que constituem o sistema de inferência possuem a seguinte forma:

$$\begin{aligned} R_j \quad : \quad & \text{Se } (x_1 \text{ é } A_1^{k_1} \text{ com certeza } w_{j1}) \ldots \textit{ and } (x_i \text{ é } A_i^{k_i} \text{ com certeza } w_{ji}) \ldots \\ & \textit{and } (x_n \text{ é } A_i^{k_n} \text{ com certeza } w_{jn}) \text{ então, } z \text{ é } z_j. \end{aligned}$$

onde os graus de certeza para cada componente do antecedente correspondem aos pesos da camada intermediária w_{ji}. Dado que as normas triangulares são utilizadas para o processamento das entradas e dos pesos no neurônio lógico, o valor máximo dos graus de certeza (pesos) será "1" (certeza total - recompensa) e o valor mínimo será "0" (totalmente errado - punição).

Em (Barto & Jordan 1987), um sinal de reforço "δ" é utilizado para o ajuste dos pesos da camada intermediária de uma rede estática multicamada. Durante este processo, não é necessário o cálculo de derivadas, ao contrário do algoritmo de retropropagação. Este sinal de reforço é transmitido por igual a todas as unidades intermediárias e as variações dos pesos para cada iteração são orientadas no sentido que gerou uma resposta com sucesso (recompensa), caso contrário, a variação do peso será feita no sentido contrário (punição).

Assim, combinando as equações que definem a dinâmica destas duas abordagens, e assumindo que o sinal de reforço adota valor "1" quando a rede consegue um bom desempenho e "0" caso contrário, a variação dos pesos da camada intermediária pode ser calculada da seguinte maneira:

$$\triangle\Theta = \begin{cases} \alpha_1 \cdot (1 - \Theta), & \text{para } \delta = 1 \\ \alpha_2 \cdot (0 - \Theta) = -\alpha_2 \cdot \Theta, & \text{para } \delta = 0 \end{cases} \tag{4.20}$$

sendo Θ o peso associado à regra ou regras ativadas pelo padrão apresentado. A Equação (4.20) representa um caso particular para δ binário; sendo necessária uma generalização desta equação para um sinal de reforço $0 \leq \delta \leq 1$, como foi proposto em (Barto & Jordan 1987).

Seja $\delta = 1 - \epsilon$, onde ϵ é dado por (4.18). Para valores baixos de ϵ, o sinal de reforço é

maior (recompensa), os pesos (graus de certeza) são aumentados. Valores altos de ϵ indicam um desempenho pobre da rede, fornecendo um sinal de reforço baixo e indicando que as regras ativadas não foram as corretas, tendo então que diminuir os graus de certeza destas regras.

Desta forma, a regra de atualização dos pesos da camada intermediária que faz parte do sistema de inferência fuzzy pode ser definida como segue:

$$\triangle\Theta = \alpha_1 \cdot \delta \cdot (1 - \Theta) - \alpha_2 \cdot (1 - \delta) \cdot \Theta \tag{4.21}$$

onde, $0 < \alpha_1 << \alpha_2 < 1$.

Finalmente, substituíndo o parâmetro Θ da Equação (4.21), pelo parâmetro correspondente, as regras a utilizar para a atualização dos pesos da camada intermediária das redes neurofuzzy propostas serão, para os pesos entre a camada de entrada e a camada intermediária:

$$\Delta w_{ji} = \delta\alpha_1[1 - w_{ji}] - (1 - \delta)\alpha_2 w_{ji} \tag{4.22}$$

e, para os pesos de recorrência da rede **RNFRGlob**:

$$\Delta r_{jl} = \delta\alpha_3[1 - r_{jl}] - (1 - \delta)\alpha_4 r_{jl} \tag{4.23}$$

Já para os pesos da rede **RNFRLoc**:

$$\Delta r_j = \delta\alpha_3[1 - r_j] - (1 - \delta)\alpha_4 r_j \tag{4.24}$$

onde, $0 < \alpha_1 << \alpha_2 < 1$ e $0 < \alpha_3 << \alpha_4 < 1$ são as taxas de aprendizado, com $j = 1,\ \ldots,\ M$ e $i = 1,\ \ldots,\ n$.

Os neurônios da camada de saída das redes neurofuzzy propostas, com funções de ativação sigmoide como já foi dito em seções anteriores, fornece uma saída normalizada ($\hat{\mathbf{y}} \in [0, 1]$). Isto garante que o sinal de reforço ϵ seja um sinal unitário, como é necessário para a sua utilização na atualização dos pesos $\mathbf{w}$ e $\mathbf{r}$, segundo o método de treinamento por reforço descrito neste trabalho e proposto em (Barto & Jordan 1987). É por este motivo que os

conjuntos de treinamento utilizados nas simulações, são compostos por padrões com saídas **y** normalizadas no intervalo unitário.

4.8 Rede Neurofuzzy Recorrente - Lee (RNFRLoc_Lee)

A rede neurofuzzy recorrente proposta em (Lee & Teng 2000) é ilustrada na Figura 4.15. Esta estrutura, assim como as estruturas propostas, desempenha o papel de um mecanismo de inferência fuzzy, utilizando uma rede neural recorrente particular com quatro camadas.

As características de cada camada da rede são as seguintes:

1. *Primeira camada*: A primeira camada é a camada de entrada da rede, tendo neurônios com funções de ativação lineares, que transmitem diretamente o vetor de entrada **x**, $n-$dimensional para a próxima camada.

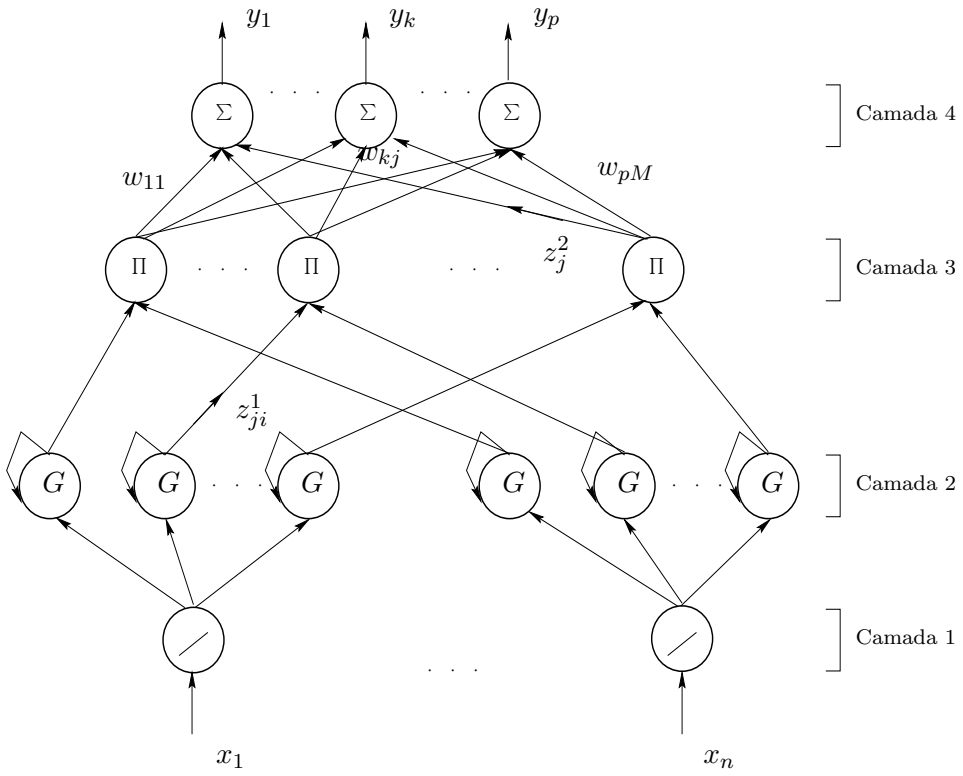


Figura 4.15: Rede neurofuzzy recorrente **RNFRLoc_Lee**.

2. *Segunda camada*: A segunda camada é formada por $n \times M$ neurônios, sendo M o número de conjuntos nebulosos que compõem a partição de x_i. A função de ativação de cada

um destes neurônios, desempenha o papel de função de pertinência e de unidade de memória. As funções de pertinência adotadas são do tipo Gaussianas:

$$z_{ji}^1 = \exp\left\{-\frac{(u_{ji} - c_{ji})^2}{(\sigma_{ji})^2}\right\} \tag{4.25}$$

sendo z_{ji}^1 a saída do $j-$ésimo neurônio associado a $i-$ésima entrada e onde c_{ji} e σ_{ji} são o valor modal e a dispersão das funções de pertinência. u_{ji} é o sinal de entrada para esta camada, $i = 1, \ldots, n$, $j = 1, \ldots, M$. A recorrência está incorporada em cada neurônio desta camada. Assim, a entrada u_{ji} para cada neurônio na segunda camada no tempo t, está definida como:

$$u_{ji}(t) = x_i(t) + z_{ji}^1(t-1) \cdot r_{ji} \tag{4.26}$$

sendo r_{ji} o peso sináptico do laço de recorrência no neurônio ji (o $j-$ésimo neurônio correspondente a $i-$ésima entrada), e $z_{ji}^1(t) = q^{-1} z_{ji}^1(t-1)$ como se mostra na Figura 4.16. Cada neurônio nesta camada possui três parâmetros ajustáveis: c_{ji}, σ_{ji} e r_{ji}.

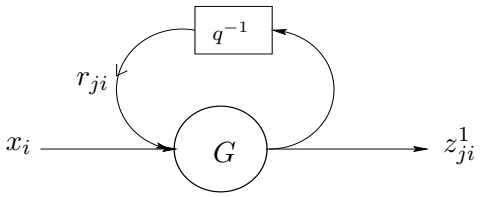


Figura 4.16: Neurônio recorrente.

3. *Terceira camada*: Cada neurônio na terceira camada representa uma regra do sistema sistema. A saída z_j^2 de cada neurônio é definida como:

$$z_j^2 = \prod_{i=1}^{M} z_{ji}^1 = \exp\left\{-\sum_{i=1}^{M} \frac{(u_{ji} - c_{ji})^2}{(\sigma_{ji})^2}\right\} \tag{4.27}$$

A saída z_j^2 do $j-$ésimo neurônio, representa o grau de ativação da regra correspondente.

4. *Quarta camada*: Os neurônios desta camada possuem funções de ativação lineares. Esta camada efetua a defuzzificação no sistema fuzzy, sendo que a saída (número real) é

calculada como uma combinação linear dos conseqüentes obtidos para cada regra, isto é:

$$y_k = \sum_{j=1}^{M} z_j^2 w_{kj} \tag{4.28}$$

onde y_k é a $k-$ésima saída da rede e w_{kj} são os pesos sinápticos ajustáveis, com $j = 1, \ldots, M$ e $k = 1, \ldots, p$, onde p é o número de saídas da rede.

Finalmente, substituindo (4.25) e (4.27) em (4.28) obtém-se a saída y_k da rede, isto é:

$$y_k = \sum_{j=1}^{M} \left[w_{kj} \prod_{i=1}^{n} \exp \left\{ - \frac{[x_i(t) + z_{ji}^1(t-1) \cdot r_{ji} - c_{ji}]^2}{(\sigma_{ji})^2} \right\} \right] \tag{4.29}$$

onde c_{ji}, σ_{ji}, r_{ji} e w_{kj} são os parâmetros ajustáveis da rede, tendo assim, um total de $(3nM) + (Mp)$ parâmetros a atualizar, sendo estes atualizados via o método de retro-propagação (Haykin 1994).

4.9 Sistema de Inferência Neurofuzzy Adaptativo (ANFIS)

O Sistema de inferência neurofuzzy adaptativo (*Adaptive NeuroFuzzy Inference System* - **ANFIS**) é uma rede neural proposta por Jang (Jang 1993), (Jang et al. 1997). Dado um conjunto de padrões entrada-saída, o **ANFIS** constrói um sistema de inferência neural fuzzy equivalente. Os parâmetros associados com as funções de pertinência são ajustados via um algoritmo de aprendizado. O ajuste destes parâmetros é efetuado utilizando o algoritmo de retropropagação ou uma combinação deste com um algoritmo do tipo quadrados mínimos (*Least Squares*).

Esta estrutura implementa sistemas do tipo Takagi - Sugeno (Takagi & Sugeno 1985), com funções lineares ou constantes nos consequentes das regras que formam o sistema, tendo estas regras pesos unitários.

A Figura 4.17 mostra a rede neurofuzzy equivalente e a Figura 4.18 ilustra o mecanismo de inferência associado ao modelo. Este sistema, a modo de exemplo, possui duas entradas x, y, uma saída f e duas regras da forma:

$$\begin{aligned} &Regra\ 1 : SE\ (x\ \acute{e}\ A_1)\ E\ (y\ \acute{e}\ B_1)\ ENT\tilde{A}O\ f_1 = p_1 x + q_1 y + r_1 \\ &Regra\ 2 : SE\ (x\ \acute{e}\ A_2)\ E\ (y\ \acute{e}\ B_2)\ ENT\tilde{A}O\ f_2 = p_2 x + q_2 y + r_2 \end{aligned} \tag{4.30}$$

Considerando o exemplo ilustrado, a saída da camada 1 é composta pelos graus de pertinência do padrão de entrada (x, y) com relação aos subconjuntos fuzzy que formam a partição de x e y, respectivamente. Assumindo funções de pertinência triangulares, estes possuem três parâmetros ajustáveis $a_i,\ m_i,\ b_i$ (ver Seção 3.2.2), com $i = 1, \ldots, N_i$, sendo N_i o número de partições do $i-$ésimo componente do vetor de entrada.

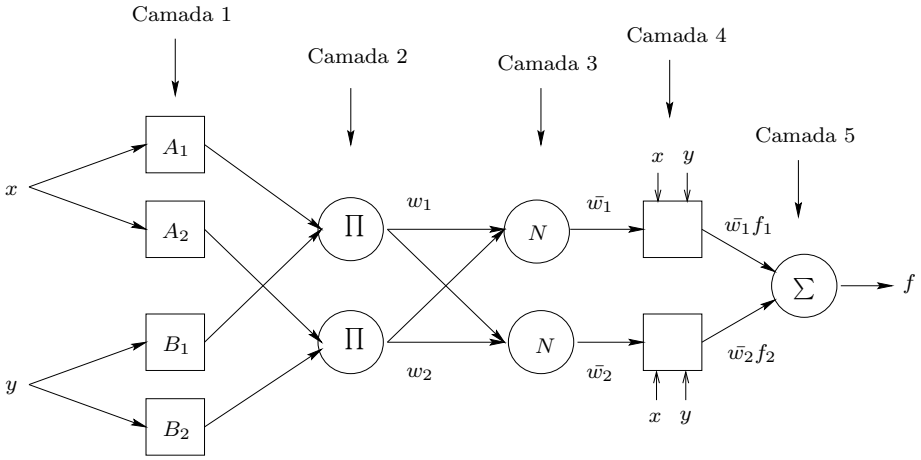


Figura 4.17: Estrutura do ANFIS.

Estas variáveis ajustáveis são os parâmetros associados aos antecedentes das regras. No exemplo, $N_i = 2, \forall i$.

A segunda camada calcula o grau de ativação de cada regra. No caso do exemplo ilustrado na Figura 4.17 e, considerando a $t-$norma como o produto algébrico (neurônio Π), tem-se:

$$w_i = \mu_{A_i}(x)\mu_{B_i}(y), \qquad i = 1, 2 \tag{4.31}$$

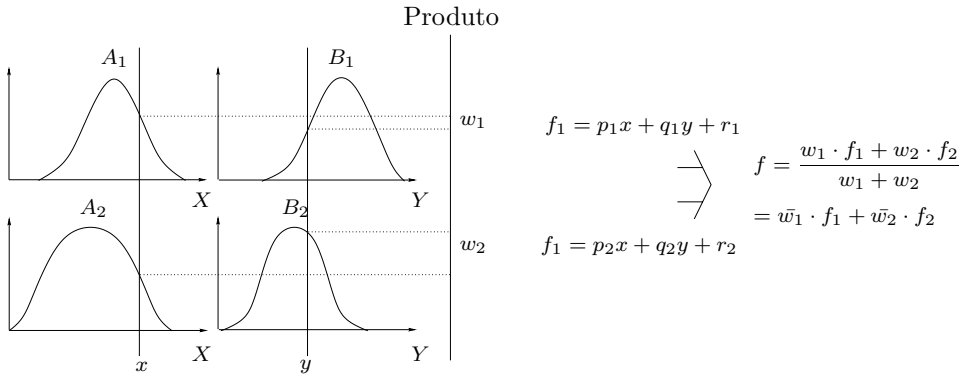


Figura 4.18: Mecanismo de inferência - ANFIS.

sendo que qualquer outro tipo de operador $t-$norma pode ser utilizado na implementação do neurônio Π.

Na terceira camada, os graus de relevância de cada regra são normalizados via neurônios N, isto é:

$$\bar{w_i} = \frac{w_i}{w_1 + w_2} \qquad\qquad i = 1, 2 \tag{4.32}$$

As funções de ativação dos neurônios que formam a quarta camada é definida como:

$$z_{4,i} = \bar{w_i} f_i = \bar{w_i}(p_i x + q_i y + r_i) \tag{4.33}$$

sendo $p_i,\ q_i,\ r_i$ os parâmetros associados aos conseqüentes das regras.

Finalmente, na quinta camada, a saída f da rede neurofuzzy é calculada da seguinte maneira:

$$f = \sum_i \bar{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{4.34}$$

O processo de treinamento é composto de duas etapas. Na primeira etapa (*forward*), os parâmetros dos antecedentes são fixos e os parâmetros dos conseqüentes são determinados via o algoritmo dos quadrados mínimos. Na segunda etapa (*backward*), os sinais de erro são retropropagados e os parâmetros dos antecedentes são atualizados via o método do gradiente descendente (retropropagação). Maiores detalhes podem ser encontrados em (Jang et al. 1997).

4.10 Resumo

Este capítulo apresentou os modelos de neurônios lógicos utilizados nas estruturas neurofuzzy propostas neste trabalho: uma rede neurofuzzy estática (**RNFEst**) e duas redes neurofuzzy recorrentes (**RNFRGlob** e **RNFRLoc**). O processo de treinamento a ser utilizado para o ajuste dos pesos das redes neurofuzzy também foi descrito. Este algoritmo possui duas etapas: a geração da partição do espaço de entrada e o ajuste dos pesos. O algoritmo de ajuste dos pesos baseia-se no método do gradiente, o qual é utilizado para atualizar os pesos da camada de saída das estruturas neurofuzzy e o método de treinamento por reforço associativo com mecanismo de recompensa e punição, utilizado para atualizar os pesos da camada intermediária. Finalmente, foram apresentadas duas estruturas de redes neurofuzzy (rede neurofuzzy recorrente (**RNFRLoc_Lee**) proposta em (Lee & Teng 2000) e a rede neurofuzzy adaptativa **ANFIS** proposta em (Jang 1993)), que serão utilizadas para comparação com as estruturas aqui propostas.

Capítulo 5

Identificação e Controle de Sistemas

5.1 Introdução

As redes neurais e os sistemas fuzzy, quando combinados, produzem um sistema neurofuzzy onde a capacidade de aprendizado e aproximação universal, tratamento de informações lingüísticas e de processamento de conhecimento, são herdadas das duas abordagens. Segundo o Capítulo 4, estes sistemas podem ser classificados em duas classes: sistemas neurofuzzy estáticos e sistemas neurofuzzy recorrentes, sendo que estes últimos tem o propósito de processar informação com dependência temporal através de estruturas mais eficientes e mais compactas.

Este capítulo apresenta resultados de simulações em aplicações de identificação e controle de sistemas dinâmicos não lineares, utilizando as estruturas estática e recorrentes propostas, assim como, estruturas neurais e fuzzy puras sugeridas na literatura.

Os resultados obtidos mostram que as redes neurofuzzy são eficazes para aproximar modelos de sistemas dinâmicos não lineares, fornecendo modelos mais simples e com erros pequenos de aproximação, demonstrando assim, a potencialidade dos sistemas neurofuzzy.

Alguns conceitos relacionados a teoria de identificação e controle de sistemas dinâmicos não lineares são apresentados para, a seguir, fazer uma breve descrição das redes neurais a serem utilizadas durante os testes. Finalmente, resultados de simulações e comparações são apresentados.

5.2 Identificação e Controle de Sistemas

5.2.1 Caracterização de Sistemas

Em (Narendra & Parthasarathy 1990), quatro classes de modelos para representar sistemas em aplicações de identificação e controle são definidos e sugeridos. A Figura 5.1 ilustra o modelo genérico ARMA para sistemas lineares. Neste modelo, a saída resultante é uma função linear das entradas e de valores passados da própria saída da planta, descrito pela Equação:

Modelo I :

$$y(t+1) = \sum_{i=0}^{n-1} \alpha_i y(t-i) + g[u(t), u(t-1), \ldots, u(t-m+1)] \tag{5.1}$$

Os demais modelos de sistemas não lineares, ilustrados nas Figuras 5.2, 5.3 e 5.4, são expressos da seguinte maneira:

Modelo II :

$$y(t+1) = f[y(t), y(t-1), \ldots, y(t-n+1)] + \sum_{i=0}^{m-1} \beta_i(t-i)u(t-i) \tag{5.2}$$

Modelo III :

$$y(t+1) = f[y(t), \ldots, y(t-n+1)] + g[u(t), \ldots, u(t-m+1)] \tag{5.3}$$

Modelo IV :

$$y(t+1) = f[y(t), \ldots, y(t-n+1); u(t), \ldots, u(t-m+1)] \tag{5.4}$$

Nos modelos de I - IV, $u(t)$ e $y(t)$ representam a entrada e a saída de um sistema SISO (*Single Input - Single Output*) no instante t, m é o número de atrasos em u, n é o número de atrasos em y com $m \leq n$. Estas expressões (5.1) - (5.4) podem ser generalizadas para o caso de sistemas MIMO (*Multiple Input - Multiple Output*), considerando $\mathbf{u}$ e $\mathbf{y}$ vetores s e $p-$dimensionais, respectivamente.

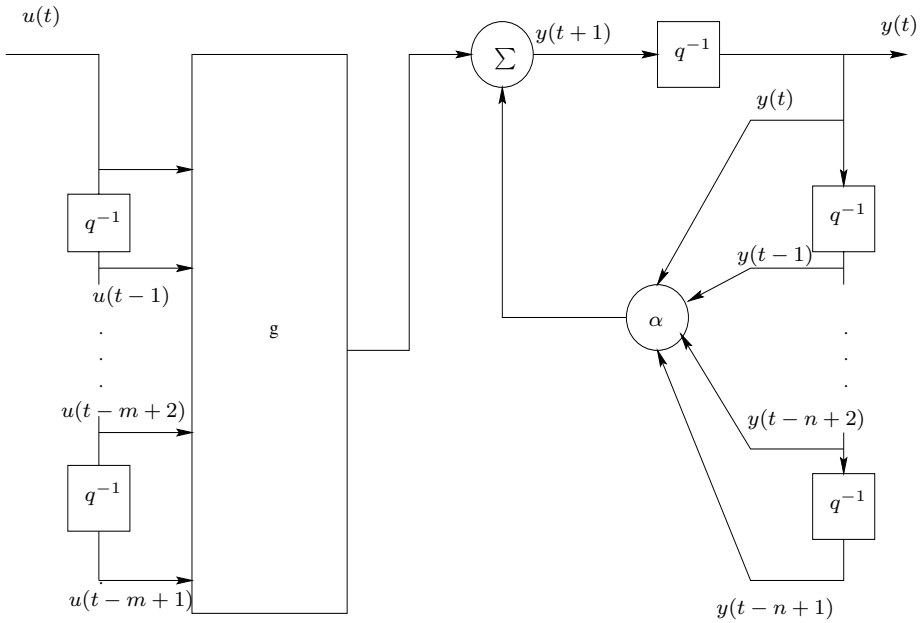


Figura 5.1: Modelo I.

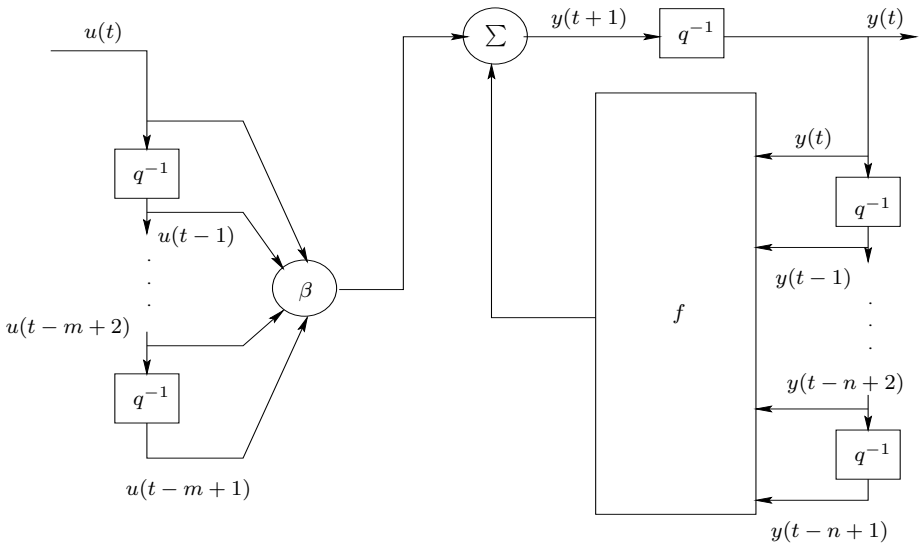


Figura 5.2: Modelo II.

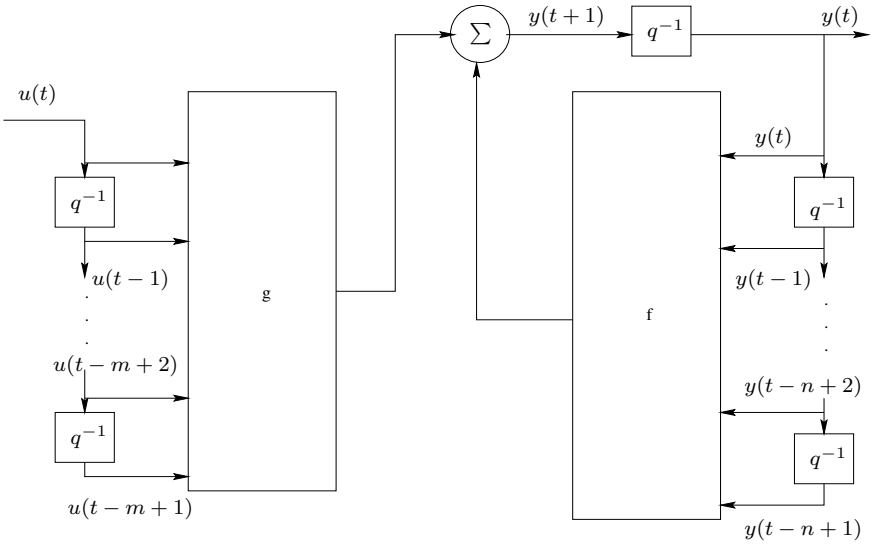


Figura 5.3: Modelo III.

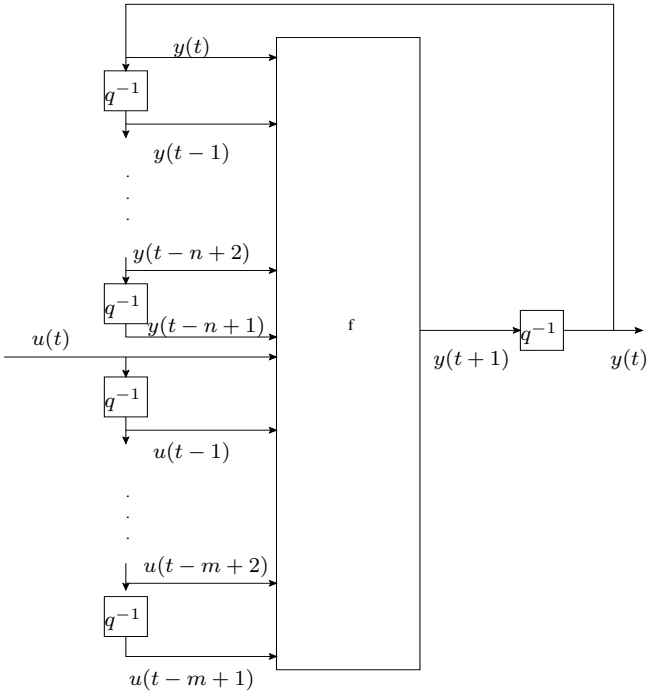


Figura 5.4: Modelo IV.

A função f é um mapeamento $f: \Re^n \rightarrow \Re$ nos modelos II e III e um mapeamento $f: \Re^{n+m} \rightarrow \Re$ no modelo IV. Assume-se que as funções f e g são diferenciáveis com relação aos seus argumentos.

Nos quatro modelos apresentados, a saída do sistema $y(t+1)$ depende dos seus n valores passados assim como dos m valores passados da entrada u. O modelo I é linear com relação a y. O modelo II é linear somente com relação à entrada u. No modelo III, existe uma independência entre a não linearidade com relação a entrada e a não linearidade com relação a saída. O modelo IV é uma generalização dos modelos mencionados.

Na identificação de um sistema desconhecido, assume-se que o mesmo é BIBO estável (*Bounded Input - Bounded Output*), isto é, que toda entrada limitada, produza uma saída limitada. Sendo assim, o sistema pode ser identificado a partir de dados entrada-saída.

5.2.2 Metodologia de Identificação

Seja um sistema representado por S, um mapeamento do espaço de entrada para o espaço da saída. Seja C a família de modelos de sistemas não lineares tal que $S \in C$. O problema de identificação consiste em determinar uma coleção de modelos $\hat{C} \subset C$ e um mapeamento $\hat{S} \in \hat{C} \subset C$ tal que $\hat{S}$ aproxime S, (Narendra & Parthasarathy 1990). Em outras palavras, o problema de aproximação consiste em determinar um $\hat{S}$ tal que:

$$\|\hat{S} - S\| \leq \epsilon \tag{5.5}$$

para algum $\epsilon > 0$ e uma norma $\|\cdot\|$ definida no espaço de estados.

Um caso particular do problema de identificação em redes neurais, consiste em selecionar um modelo neural $\hat{S}$ determinado e ajustar os seus pesos, de modo a otimizar uma função objetivo, que em geral, é baseada no erro entre a saída do sistema real e a saída do modelo.

Portanto, o modelo deve ser escolhido de modo que possua um comportamento idêntico ou o mais próximo possível ao comportamento do sistema real a ser modelado. No caso onde o modelo escolhido é uma rede neural, os parâmetros devem ser ajustados tal que, para as mesmas condições iniciais, tanto o modelo como o sistema apresentam as mesmas saídas para qualquer entrada. Duas estruturas para o processo de identificação de sistemas com redes neurais e, portanto, com redes neurofuzzy podem ser utilizadas:

1. ***Estrutura paralela***: a Figura 5.5, ilustra o modelo paralelo para um sistema da classe

de modelos do tipo IV com um número de atrasos $n = 1$ para y e $m = 1$ para u. O modelo é descrito pela seguinte equação:

$$\widehat{y}(t+1) = \hat{f}[\widehat{y}(t), u(t)] \tag{5.6}$$

Este modelo não garante a convergência do sistema, sendo por isto utilizado o modelo série-paralelo.

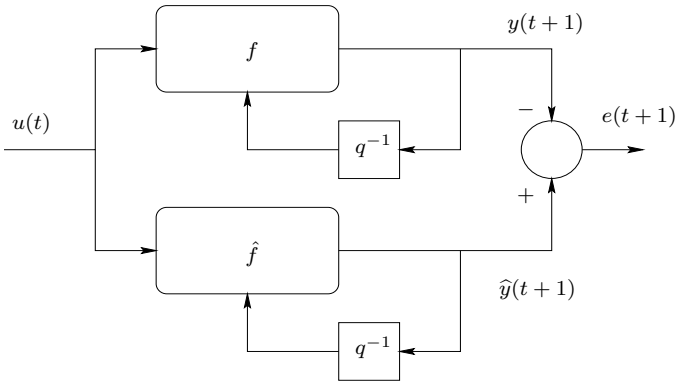


Figura 5.5: Estrutura paralela para identificação de sistemas.

2. ***Estrutura série-paralela***: neste caso, a saída real do sistema e não a saída do modelo é realimentada, como ilustra a Figura 5.6. Assim, o modelo é descrito pela equação:

$$\hat{y}(t+1) = \hat{f}[y(t), u(t)] \tag{5.7}$$

Este procedimento será utilizado para os modelos de redes neurais puras e híbridas durante o processo de treinamento, sendo válido para as quatro classes de sistemas, isto é, para modelos do tipo I a IV.

As vantagens de utilizar o modelo serie-paralelo são as seguintes:

2.1 Dado que o sistema é assumido ser BIBO estável, todos os sinais de entrada utilizados durante o processo de identificação no modelo de rede neural ou, qualquer outra abordagem, serão limitados (*bounded*).

2.2 Devido a não existência do laço de realimentação no modelo de identificação, é possível utilizar o método de retropropagação para o ajuste dos parâmetros diminuindo assim, o custo computacional.

2.3 Assumindo que o erro na saída da rede neural tende a valores pequenos, tal que $y(t) \approx \widehat{y}(t)$, o modelo série-paralelo pode ser substituído pelo modelo paralelo sem conseqüências sérias.

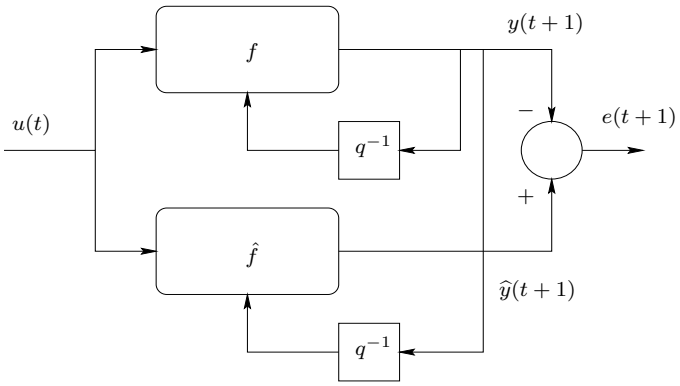


Figura 5.6: Estrutura série-paralela para identificação de sistemas.

Segundo (de Moraes Lima 2000), o processo de identificação de sistemas não lineares pode ser realizado de duas formas básicas: analiticamente e computacionalmente, podendo ser estas utilizadas separadamente ou combinadas:

1. ***Identificação analítica***: envolve a análise da dinâmica do sistema e o desenvolvimento de um modelo matemático para o sistema.

2. ***Identificação computacional***: envolve a coleta estatística das características de entrada-saída da planta e sua utilização na determinação de um modelo que aproxime o comportamento observado. Esta tarefa é formada pelos quatro passos básicos descritos a seguir:

   2.1 *Planejamento experimental*: determinação de como os dados serão coletados, ou seja, definir o método de amostragem a ser utilizado;

2.2 *Seleção da estrutura do modelo*: seleção da estrutura do modelo e posterior determinação dos parâmetros livres ou ajustáveis;

2.3 *Estimação de parâmetros*: ajuste dos parâmetros livres utilizando as estatísticas obtidas dos dados;

2.4 *Validação*: avaliação do desempenho do modelo para os dados de teste.

Em geral, uma rede neural é utilizada principalmente quando existe pouco conhecimento da planta. Para resolver um problema de identificação de sistemas utilizando redes neurais, é necessário garantir que os dados amostrem toda a região de operação do sistema no espaço de estados (de Moraes Lima 2000).

O processo de identificação computacional foi aplicado na obtenção dos resultados de simulação, que serão descritos no próximo capítulo.

A abordagem utilizada neste trabalho para identificação é ilustrada na Figura 5.7, abordagem que pode ser classificada como um mecanismo computacional clássico de treinamento supervisionado (Hunt et al. 1992) (ver Seção 2.5.1). A rede neurofuzzy é conectada em paralelo com o sistema não linear e o sinal de erro entre as saídas do sistema e da rede é utilizado para a atualização dos parâmetros ajustáveis do modelo. Este procedimento é também utilizado para o treinamento de sistemas neurais puros e sistemas fuzzy adaptativos.

Além de definir os passos necessários para o processo de identificação computacional a ser utilizado, é preciso classificar inicialmente o sistema não linear, em cada uma das quatro classes de modelos de sistemas definidos na Seção 5.2.1.

Segundo (Cybenko 1989), o teorema de aproximação universal é definido como segue:

Seja $\varphi(\cdot)$ uma função não constante, limitada e monotonicamente crescente. Seja I_p o hipercubo unitário $p-$dimensional. O espaço de funções contínuas em I_p é definido como $C(I_p)$. Logo, dada qualquer função $f \in C(I_p)$ e $\epsilon > 0$, existe um inteiro M e um conjunto de constantes reais α_i, θ_i e w_{ji}, onde $i = 1, \ldots, M$ e $j = 1, \ldots, p$ tal que:

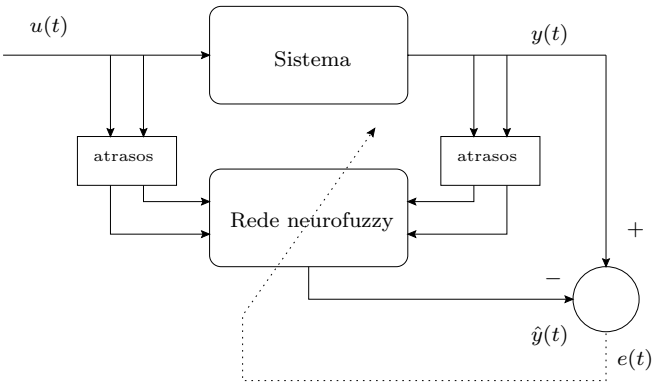


Figura 5.7: Identificação utilizando redes neurais.

$$N(x_1, \ldots, x_p) = \sum_{j=1}^{p} \alpha_i \varphi \left(w_{ji} x_i - \theta_i \right) \tag{5.8}$$

onde $N(x_1, \ldots, x_p)$ é a realização aproximada da função $f(\cdot)$; isto é:

$$|N(x_1, \ldots, x_p) - f(x_1, \ldots, x_p)| < \epsilon \tag{5.9}$$

para todo $\{x_1, \ldots, x_p\} \in I_p$.

Este teorema é diretamente aplicável as redes neurais, pois satisfaz todas as condições impostas. Considerando novamente o exemplo da Equação (5.6), a estrutura deste sistema corresponde ao modelo IV de classes de sistemas não lineares. O objetivo da rede neural será aproximar a função $f(\cdot)$, isto é, fazer $N \approx f(\cdot)$, sendo N o modelo fornecido pela rede neural ou neurofuzzy.

Finalmente, é importante que o conjunto de dados entrada-saída utilizado durante o treinamento seja representativo do espaço de ação do sistema. Isto garantirá que o sistema treinado, forneça uma resposta próxima a desejada; este conceito é denominado de excitação persistente (Miller et al. 1990).

5.2.3 Metodologia de Controle

Um problema típico de controle adaptativo consiste em controlar a saída de um sistema com estrutura conhecida mas cujos parâmetros são desconhecidos. Para tratar este problema, a teoria de sistemas adaptativos em geral considera a planta a ser controlada como linear e invariante no tempo, com um conjunto de parâmetros não conhecidos p. Se p é conhecido, os parâmetros θ do controlador podem ser escolhidos e assim, com estes dois grupos de parâmetros, atingir o objetivo, isto é, que o processo tenha uma saída desejada. Se p é desconhecido, θ deve ser ajustado *on-line* utilizando toda informação acessível da planta a ser controlada (Miller et al. 1990).

Diversas estruturas de controle relacionadas com redes neurais têm sido propostas na literatura (Hunt et al. 1992). Segundo J. Suykens (Suykens et al. 1995), as estratégias de controle mais adotadas aplicando redes neurais são: controle neural adaptativo, controle neural ótimo, controle via o método de treinamento por reforço e controle preditivo.

Existem duas abordagens a ser consideradas no controle adaptativo: controle direto e controle indireto:

1. ***Controle direto***: os métodos utilizados para ajustar os parâmetros do controlador tem como objetivo minimizar alguma norma do sinal de erro com relação a saída (Figura 5.8). A saída desejada é fornecida por um modelo de referência.

2. ***Controle indireto***: neste caso, conforme ilustra a Figura 5.9, a dinâmica do sistema é identificada por uma rede neural N_i. Os parâmetros do controlador N_c são ajustados assumindo N_i como a planta a ser controlada.

Neste trabalho, adota-se a abordagem de controle adaptativo direto, isto é, utiliza-se um modelo de referência (Narendra & Parthasarathy 1990). A rede neurofuzzy é inicialmente treinada *off-line* para identificar as não linearidades do sistema a ser controlado. A seguir, é efetuado um treinamento *on-line* para refinar os pesos do controlador, utilizando o erro de saída para o treinamento, isto é, a diferença entre a saída desejada fornecida pelo modelo de referência e a saída real da planta.

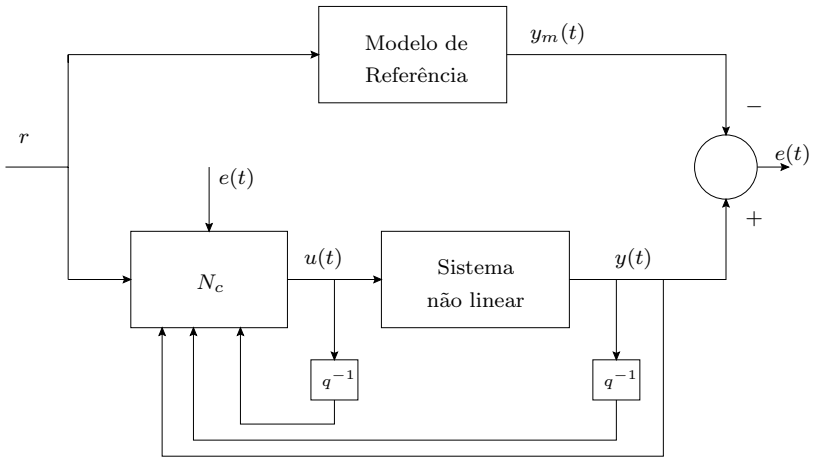


Figura 5.8: Controle direto.

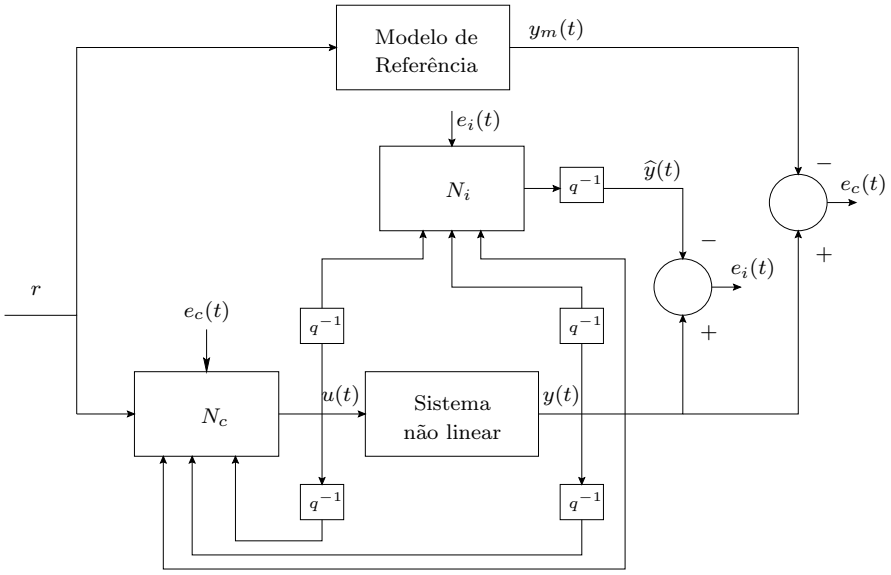


Figura 5.9: Controle indireto.

As técnicas de controle mencionadas neste capítulo são aquelas de interesse direto para este trabalho. Detalhes sobre outros métodos de controle com redes neurais podem ser encontradas em (Miller et al. 1990), (Ronco & Gawthrop 1997) e (de Moraes Lima 2000).

5.3 Modelos de Redes Neurais Implementados

Os modelos de redes neurais implementados para os estudos de simulação foram os seguintes:

1. Redes neurais/neurofuzzy estáticas:

   1.1 **MLP**: rede neural clássica multicamadas, com duas camadas intermediárias e M_1 e M_2 neurônios em cada camada, respectivamente, incluindo a polarização, com funções de ativação idênticas, sigmoidais e pesos ajustados via o método de retropropagação, padrão a padrão.

   1.2 **RNFEst**: rede neurofuzzy estática, ilustrada na Figura 4.8, com M neurônios lógicos do tipo AND na camada intermediária, utilizando para processamento as normas triangulares produto algébrico e a soma probabilística, funções de pertinência triangulares para as partições do espaço de entrada e os pesos ajustados via o método de treinamento de retropropagação e reforço associativo, conforme Seção 4.7.

   1.3 **SFA**: sistema fuzzy adaptativo proposto em (Wang 1994), com método de agrupamento do tipo *Nearest Neighborhood* (Seção 3.3.1).

   1.4 **RNEst**: rede neural baseada na **RNF1**, com M neurônios clássicos na camada intermediária, com funções de ativação sigmoidais e N_i conjuntos nebulosos associados às partições das entradas, sendo estes conjuntos definidos por funções de pertinência triangulares para cada componente da entrada; os pesos são ajustados via o método de retropropagação.

   1.5 **ANFIS**: sistema de inferência neurofuzzy adaptativo (Jang 1993), cuja estrutura foi detalhada na Seção 4.9. Os pesos são ajustados utilizando o algoritmo de retropropagação combinado com o algoritmo de quadrados mínimos.

2. Redes neurais/neurofuzzy recorrentes:

   2.1 **RNFRGlob**: rede neurofuzzy com recorrência interna global, ilustrada na Figura 4.5. A estrutura, detalhada na Seção 4.4, utiliza o algoritmo de aprendizado baseado no método do gradiente e aprendizado por reforço associativo.

   2.2 **RNFRLoc**: rede neurofuzzy com recorrência interna local, ilustrada na Figura 4.7, com M neurônios lógicos do tipo AND, utilizando como $t-$norma o produto algébrico e a $s-$norma soma probabilística, com algoritmo de aprendizado baseado no método do gradiente e no método por reforço associativo, conforme Seção 4.7.

   2.3 **RNFRLoc_Lee**: rede neurofuzzy recorrente proposto em (Lee & Teng 2000), com M neurônios na terceira camada, sendo M definido a *priori* pelo usuário (Seção 4.8) Os pesos são ajustados via o método do gradiente.

   2.4 **RNRLoc**: rede neural com recorrência interna local, baseada na estrutura da rede neural estática **RNEst**, com pesos ajustados via o método de retropropagação.

   2.5 **RNRLoc_Cheng**: rede neural clássica com recorrência interna local (Cheng et al. 1997), com polarização, M neurônios na camada intermediária com funções de ativação sigmoidais e os pesos ajustados utilizando o método de retropropagação.

Estes modelos, exceto o modelo **ANFIS**, foram implementados em C++, em ambiente Windows NT e processador Intel Pentium II, 233 MHz, 256 Mb de memória RAM. O modelo **ANFIS** utilizado é o disponível no *toolbox* do Matlab.

A Tabela 5.1 resume os parâmetros característicos dos modelos implementados e o número de parâmetros ajustáveis por iteração sendo, Na_{max}, o número de neurônios ativos na camada intermediária para cada padrão apresentado, n o número de entradas e p o número de saídas. No caso da rede **MLP**, $Na_{max} = [M_1,\ M_2]$ refere-se ao número de neurônios das camadas intermediárias correspondentes (todos os neurônios são ativados em cada iteração). Já para a rede recorrente **RNR2**, $Na_{max} = M$.

Tabela 5.1: Parâmetros característicos das redes neurais.

Tipo Estrutura	**Estrutura**	**Parâmetros / iteração**
Rede neural estática	RNEst	$Na_{max} \cdot (n+p)$
Rede neural recorrente	RNRLoc	$Na_{max} \cdot (n+p+1)$
Rede neurofuzzy estática	RNFEst	$Na_{max} \cdot (n+p)$
Rede neurofuzzy com recorrência global	RNFRGlob	$Na_{max} \cdot (n+p+Na_{max})$
Rede neurofuzzy com recorrência local	RNFRLoc	$Na_{max} \cdot (n+p+1)$
Rede neural recorrência local-Cheng	RNRLoc_Cheng	$Na_{max} \cdot (n+p+2)+p$
Sistema fuzzy adaptativo	SFA	—
Rede neurofuzzy recorrência local-Lee	RNFRLoc_Lee	$3 \cdot n \cdot Na_{max} + Na_{max} \cdot p$
Sist. infer. neurofuzzy adaptativo	ANFIS	$Na_{max} \cdot (3 \cdot n+1)$
Rede neural estática multicamada	MLP	$M_1 \cdot (n+1) + M_2 \cdot (M_1+1) + (M_2+1) \cdot p$

5.4 Resumo

Este capítulo apresentou as metodologias de identificação e controle a serem utilizadas nos resultados de simulação, assim como os modelos neurais estáticos e recorrentes implementados neste trabalho.

Capítulo 6

Resultados de Simulação

6.1 Identificação de Sistemas

Nesta seção, são apresentados resultados de identificação de quatro sistemas dinâmicos não lineares. Os exemplos são classificados entre as distintas estruturas de sistemas definidas na Seção 5.2.1. Estes sistemas foram também utilizados (Narendra & Parthasarathy 1990), (Wang 1994), (Lee & Teng 2000), (de Moraes Lima 2000) e (Caminhas 1997). Os resultados dos testes e indicadores de desempenho com relação a estrutura e tempo de processamento das arquiteturas descritas acima são apresentados a seguir.

EXEMPLO 6.1

Este sistema é definido pela equação a diferenças:

$$y(t+1) = 0.3y(t) + 0.6y(t-1) + f[u(t)] \tag{6.1}$$

A dinâmica deste sistema consta de uma dependência linear de valores passados da entrada e uma dependência não linear com relação a entrada ao sistema, $u(t)$, correspondendo estas características ao modelo I da Seção 5.2.1. $y(t+1)$ é a saída do sistema e $f(\cdot)$ é a função não linear, suposta desconhecida, definida como segue:

$$f(u) = 0.6\text{sen}(\pi u) + 0.3\text{sen}(3\pi u) + 0.1\text{sen}(5\pi u) \tag{6.2}$$

sendo $u = u(t)$, para $t = 1, \ldots$. Adotou-se como condições iniciais $y(0) = y(-1) = 0.0$. O

objetivo das redes neurofuzzy, seguindo a abordagem de (Narendra & Parthasarathy 1990), é determinar um modelo para a função $f(\cdot)$. Para isto, utilizou-se um conjunto de treinamento para todas as estruturas, composta por 1000 padrões entrada-saída $[u(t), y(t+1)]$, utilizando a Equação 6.1 e com valores para $u(t)$ gerados de forma aleatória e com distribuição uniforme no intervalo $[-1,\ 1]$. Em (Narendra & Parthasarathy 1990) utilizou-se 50000 padrões de treinamento com uma taxa de aprendizado constante $\eta = 0.25$.

A Tabela 6.1, caracteriza os parâmetros que definem a complexidade das estruturas avaliadas, assim como, o número de parâmetros ativos, ou seja, o número de neurônios ativos por iteração. Tipo de agrupamento refere-se ao método adotado para a geração dos centros associados as funções de pertinência dos componentes do vetor de entrada, n é o número de entradas, p é o número de saídas, N_i é o número de partições obtido para x_i, $i = 1, \ldots, n$ e Na_{max} é o número máximo de neurônios da camada intermediária ativos por iteração.

Após concluído o treinamento, as estruturas são testadas aplicando um sinal de entrada $u(t) = \text{sen}(2\pi t/250)$, para $t = 0, \ldots, 599$.

A Tabela 6.2, mostra os erros quadráticos médios (EQM) de treinamento, de teste e os tempos de processamento, T_{proc}, em segundos, necessário para o treinamento de cada uma das redes neurais e neurofuzzy consideradas neste trabalho.

Tabela 6.1: Características das redes neurais - Exemplo 6.1.

Estrutura	n	p	Tipo de Agrupamento	N_i	Na_{max}	Parâmetros / iteração
RNEst	1	1	LVQ	[8]	2	4
RNRLoc	1	1	LVQ	[8]	2	6
RNFEst	1	1	LVQ	[8]	2	4
RNFRGlob	1	1	LVQ	[12]	2	8
RNFRLoc	1	1	LVQ	[8]	2	6
RNRLoc_Cheng	1	1	—	[30]	30	121
SFA	1	1	KNN	[64]	—	—
RNFRLoc_Lee	1	1	—	[10]	10	60
ANFIS	1	1	—	[10]	10	40
MLP	1	1	—	[20,10]	[20,10]	261

Tabela 6.2: Erros quadráticos médios (EQM) - Exemplo 6.1.

Estrutura	Épocas	T_{proc} (s)	EQM Treinamento $(\times 10^{-3})$	EQM Teste $(\times 10^{-2})$
RNEst	500	20	0.004	0.247
RNRLoc	500	42	0.002	2.330
RNFEst	25	1	0.013	1.863
RNFRGlob	32	3	0.138	0.762
RNFRLoc	20	1	0.020	1.557
RNRLoc_Cheng	1000	82	0.097	1.670
SFA	1000	0	19.270	2.241
RNFRLoc_Lee	400	36	0.001	0.096
ANFIS	300	—	0.950	0.020
MLP	1000	152	0.060	2.272

Os resultados obtidos para as estruturas **RNFRGlob**, **RNFRLoc**, **RNFEst** e **RNRLoc_Cheng**, são ilustrados nas Figura 6.1 e Figura 6.2. Estas figuras, assim como as Tabelas 6.1 e 6.2, representam os melhores resultados obtidos para cada estrutura após varias execuções, alterando os parâmetros associados a cada estrutura como taxas de aprendizado e valores modais, inicialização dos pesos, número máximo de centros para as redes neurofuzzy, tamanho dos conjuntos de treinamento e considerando o melhor desempenho para cada modelo.

Analisando os resultados para este sistema, pode-se dizer que os sistemas neurofuzzy propostos (**RNFEst**, **RNFRGlob** e **RNFRLoc**) apresentam erros quadráticos médios na ordem de 10^{-2} para as redes **RNFEst** e **RNFRLoc** e ordem de 10^{-3} para o modelo fornecido pela rede **RNFRGlob**. Além disso, as redes neurofuzzy são parcimoniosas com 8, 8 e 12 centros, com um período de treinamento menor (25, 20 e 32 épocas) para cada modelo respectivamente e por conseqüência, um baixo tempo de processamento (1, 1 e 3 segundos).

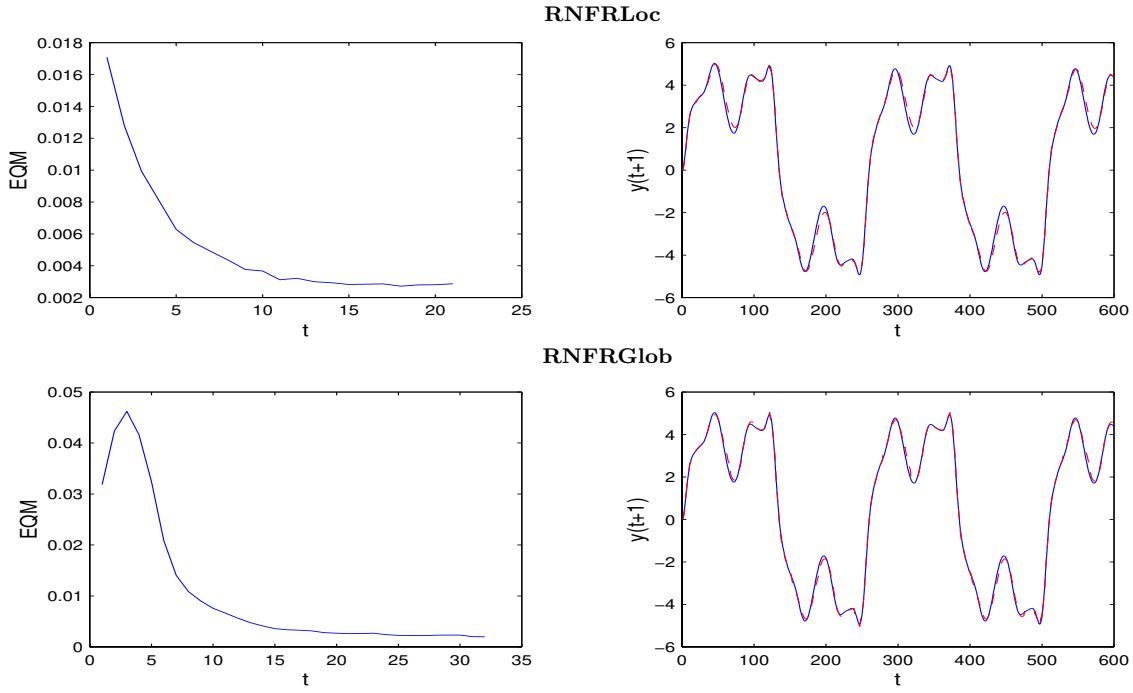


Figura 6.1: Identificação, Exemplo 6.1: (—) saída desejada, ($\cdots$) saída da rede neural.

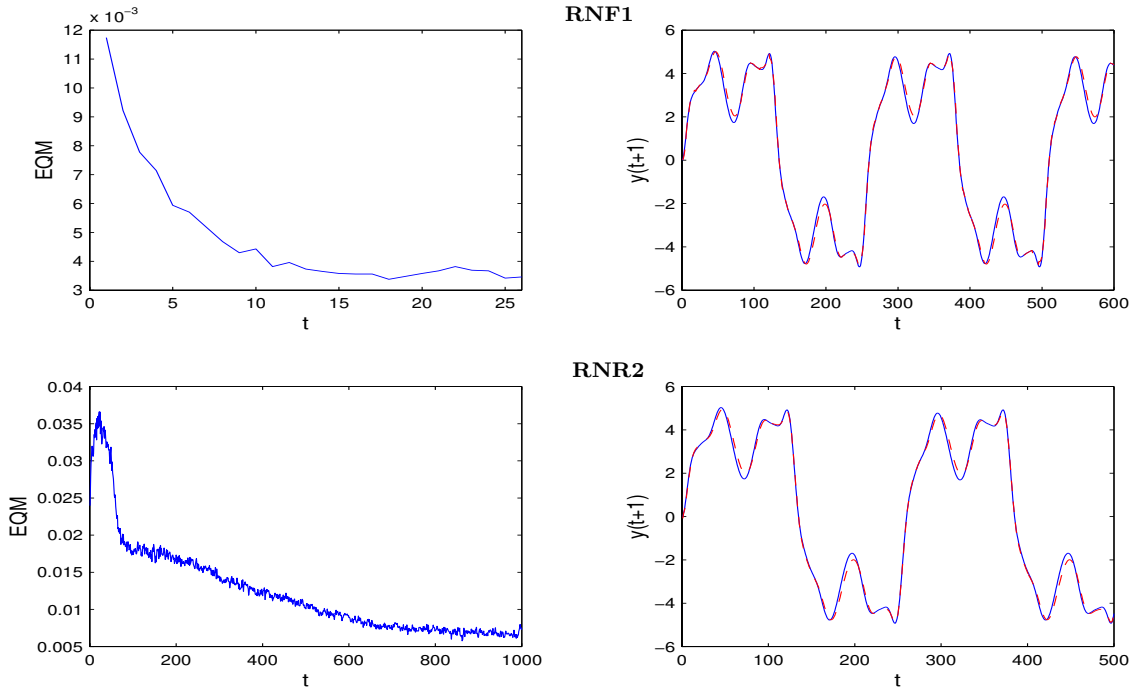


Figura 6.2: Identificação, Exemplo 6.1: (—) saída desejada, ($\cdots$) saída da rede neural.

A rede neurofuzzy **RNFRLoc_Lee** (Lee & Teng 2000) apresenta um EQM de teste menor, mas cabe ressaltar que para isto, a estrutura precisou de um maior número de parâmetros ativos por iteração e um período de treinamento mais longo (400 épocas, 36 segundos). Da mesma forma, a rede neurofuzzy **ANFIS** obteve um menor EQM de teste, mas o modelo precisou de 300 épocas, acarretando um tempo de processamento maior.

O **SFA** (Wang 1994), conseguiu o seu melhor desempenho com 64 funções de pertinência gaussianas para um EQM de teste na ordem de 10^{-2}. Já a rede multicamadas **MLP**, embora tenha utilizado uma estrutura mais complexa, não conseguiu igualar o desempenho das redes híbridas neurofuzzy, embora tenha conseguido um EQM de treinamento similar.

Entre as redes propostas, a rede neurofuzzy com recorrência global (**RNFRGlob**), obteve o melhor desempenho no teste, enquanto que os sistemas neurais estáticos (**RNEst**) e recorrentes não fuzzy (**RNRLoc**) precisaram de um maior número de épocas para a convergência (Tabela 6.2).

EXEMPLO 6.2

O sistema a ser identificado corresponde a um sistema definido por uma equação a diferenças de segunda ordem não linear, sendo classificado como o modelo II na caracterização dos sistemas:

$$y(t+1) = f[y(t),\ y(t-1)] + u(t) \tag{6.3}$$

onde,

$$f[y(t),\ y(t-1)] = \frac{y(t)y(t-1)[y(t)+2.5]}{1+y^2(t)+y^2(t-1)} \tag{6.4}$$

Este sistema não linear possui dois pontos de equilíbrio $(0,0)$ e $(2,2)$, respectivamente no espaço de estados. Isto é, a saída do sistema convergirá a seqüência $\{0\}$ ou $\{1\}$ para uma entrada $u(t) = 0$. Além disso e segundo (Narendra & Parthasarathy 1990), para qualquer

entrada $|u(t)| \leq 5$, a saída da planta é uniformemente limitada para condições iniciais $(0, 0)$ e $(2, 2)$ e satisfaz a inequação $|y(t)| \leq 13$.

O conjunto de treinamento para todos os modelos a serem ajustados, foi composto por 1000 padrões gerados a partir da Equação (6.3), com um sinal de entrada gerado de forma aleatória e com distribuição uniforme no intervalo $[-1, 1]$. (Narendra & Parthasarathy 1990) utilizou um conjunto de treinamento com 100000 padrões aleatórios e uniformemente distribuídos no intervalo $[-2, 2]$. As condições iniciais consideradas foram de $y(0) = y(-1) = 0.0$.

A Tabela 6.3 mostra as entradas utilizadas para cada estrutura na construção do modelo série-paralelo da planta. Pode-se observar que **RNFRLoc_2** refere-se a um modelo da estrutura **RNFRLoc** considerando duas entradas, entanto que **RNFRLoc_1** refere-se a um modelo de estrutura **RNFRLoc** utilizando uma única entrada. A Tabela 6.4 caracteriza as estruturas obtidas para a identificação da planta, assim como a sua complexidade.

Tabela 6.3: Entradas das redes neurais - Exemplo 6.2.

Estrutura	n	Entradas Utilizadas
RNEst	2	$y(t)$, $y(t-1)$
RNRLoc_2	2	$y(t)$, $y(t-1)$
RNFEst	2	$y(t)$, $y(t-1)$
RNFRGlob_2	2	$y(t)$, $y(t-1)$
RNFRLoc_1	1	$y(t)$
RNFR2Loc_2	2	$y(t)$, $y(t-1)$
RNRLoc_Cheng	1	$y(t)$

com $t = 0, \ldots, 99$ para ambos testes. A Tabela 6.5, mostra os erros quadráticos médios de treinamento e de teste, o tempo de processamento T_{proc} associado ao treinamento e o número de épocas para cada estrutura.

Tabela 6.4: Características das redes neurais - Exemplo 6.2.

Estrutura	n	p	Tipo de Agrupamento	N_i	Na_{max}	Parâmetros / iteração
RNEst	2	1	LVQ	[9, 9]	4	12
RNRLoc_2	2	1	LVQ	[13, 13]	4	16
RNFEst	2	1	LVQ	[16, 16]	4	12
RNFRGlob_2	2	1	LVQ	[12, 12]	4	28
RNFRLoc_2	2	1	UNIFORME	[10, 10]	4	16
RNFRLoc_1	1	1	UNIFORME	[14]	2	6
RNRLoc_Cheng	2	1	—	[30]	30	121
SFA	2	1	KNN	[70]	—	—
RNFRLoc_Lee_2	2	1	—	[30, 30]	30	210
RNFRLoc_Lee_1	1	1	—	[50]	50	200
ANFIS	2	1	—	[7,7]	7	49
MLP	2	1	—	[20, 10]	[20, 10]	281

Tabela 6.5: Erros quadráticos médios (EQM) - Exemplo 6.2.

Estrutura	Épocas	T_{proc} (s)	EQM Treinamento $(\times 10^{-3})$	EQM Teste1 $(\times 10^{-2})$	EQM Teste2 $(\times 10^{-2})$
RNEst	500	31	0.028	0.199	0.040
RNRLoc_2	500	42	0.011	0.180	0.249
RNFEst	400	32	0.003	0.522	0.574
RNFRGlob_2	750	195	0.009	1.041	0.936
RNFRLoc_2	400	47	0.001	0.728	0.760
RNFRLoc_1	500	80	0.005	1.954	2.509
RNRLoc_Cheng	500	99	0.013	0.111	0.108
SFA	300	1	8.396	0.321	0.364
RNFRLoc_Lee_2	400	113	0.006	0.011	0.010
RNFRLoc_Lee_1	500	131	1.909	7.118	6.434
ANFIS	500	—	16.152	6.130	3.860
MLP	500	124	0.012	0.002	0.002

As taxas de aprendizado para a rede neurofuzzy com recorrência local (**RNFRLoc**) foram de $\alpha_1 = 0.0001$, $\alpha_2 = \alpha_3 = 0.001$, $\alpha_4 = 0.005$ para o ajuste dos pesos da camada intermediária e de recorrência e $\eta = 0.2$ para o ajuste dos pesos da camada de saída. Já para a rede **RNFEst**, $\eta = 0.075$;.

Analisando as Tabelas 6.4 e 6.5, pode-ser observar mais uma vez, que para EQMs de treinamento e de teste da mesma ordem, as redes neurofuzzy propostas necessitaram de um número reduzido de parâmetros por iteração, quando comparadas com as outras estruturas. No entanto, a rede neurofuzzy com recorrência global **RNFRGlob_2** e a rede com recorrência local de duas entradas (**RNFRLoc_2**), assim como a rede neurofuzzy (**RNFEst**) conseguiram melhores resultados nos testes quando comparados com os resultados obtidos com a rede **RNFRLoc_1**, que utilizou uma única entrada.

Das estruturas neurofuzzy propostas e as suas modificações (**RNEst** e **RNRLoc**), a rede neurofuzzy com recorrência local (**RNFRLoc**) com duas entradas e as redes **RNEst** e **RNRLoc**, conseguiram os menores erros de teste. Isto demonstra que, nem sempre é necessária uma recorrência global para tratar sistemas dinâmicos não lineares, sendo preciso unicamente recorrência local. Não foi necessário utilizar uma partição não uniforme, resultando assim, num processo de geração da estrutura simplificado e eficiente.

O tempo de processamento das estruturas neurofuzzy propostas foi baixo (Tabela 6.3). Cabe observar que, embora a rede **RNFRGlob_2** tenha demorado 195 segundos no treinamento, esta foi a estrutura que precisou de um maior número de épocas (750). A ordem dos erros erros de teste para as redes **RNFEst** e **RNFRLoc_2** foi na ordem de 10^{-3} e para as redes **RNFRGlob_2** e **RNFRLoc_1** na ordem de 10^{-2}.

O sistema fuzzy estático adaptativo **SFA**, a rede neural multicamada **MLP** e a rede neural recorrente **RNRLoc_Cheng** conseguiram erros de treinamento baixos. No entanto, o **SFA** precisou de uma estrutura mais complexa e as estruturas **MLP** e **RNRLoc_Cheng** além disso, necessitaram de um tempo maior de execução. Deve-se observar que, no caso da estrutura **SFA**, de um total de 300 padrões que foram utilizados durante o processo de treinamento, foram geradas 70 centros para cobrir o espaço de entrada do sistema.

Comparando as estruturas **RNFRLoc_1** e **RNFRLoc_Lee_1**, com uma entrada, nota-se que a rede neurofuzzy com recorrência local proposta (**RNFRLoc**), possui uma maior capacidade de tratamento da informação de forma dinâmica, pois esta, para o mesmo conjunto de teste obteve um melhor desempenho e um menor EQM de treinamento, com 14 centros definidos via uma partição uniforme, enquanto que no caso das estruturas **RNFRLoc_lee_2** e **RNFRLoc_Lee_1** foram ajustados 50 centros via o método de retropropagação (Tabela 6.4). Além disso, o tempo de processamento foi menor embora tenha sido necessário um maior número de épocas para finalizar o treinamento.

As curvas de erro de treinamento e os resultados de simulação do primeiro teste para as redes neurofuzzy propostas são ilustrados nas Figura 6.3 para **RNFEst** e **RNFRGlob_2**, Figura 6.4 para **RNFRLoc_2** e **RNFRLoc_1**. Finalmente, resultados de simulação do segundo teste são ilustrados nas Figuras 6.5

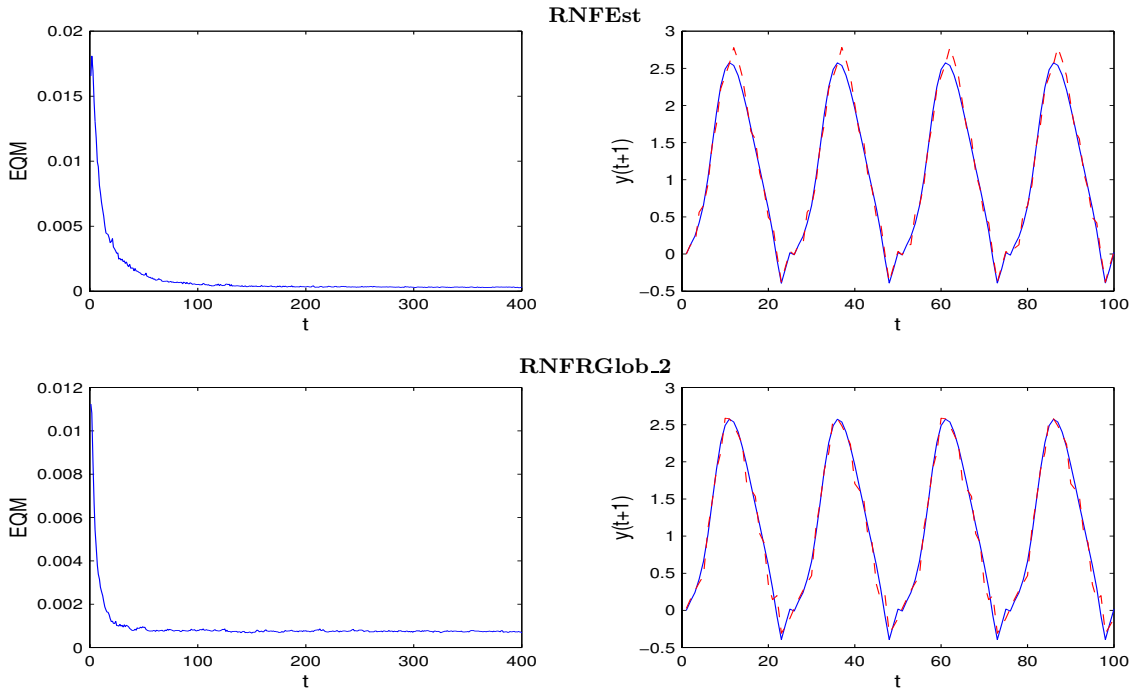


Figura 6.3: Identificação, Exemplo 6.2: (—) saída desejada, ($\cdots$) saída da rede neural.

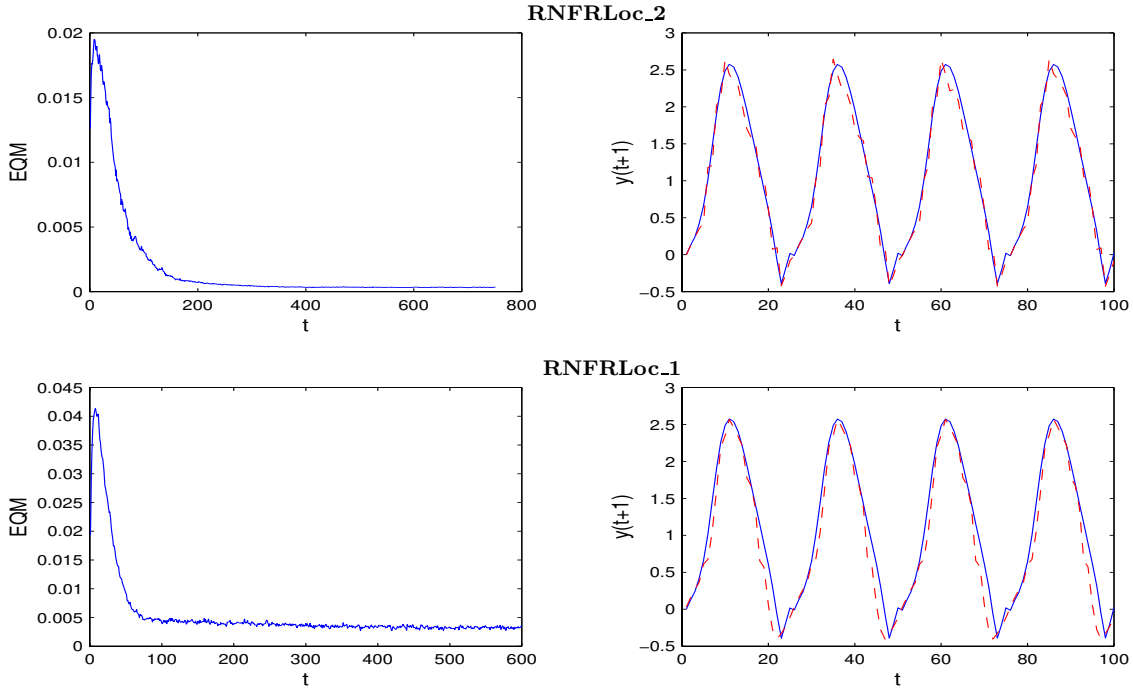


Figura 6.4: Identificação, Exemplo 6.2: (—) saída desejada, (· · ·) saída da rede neural.

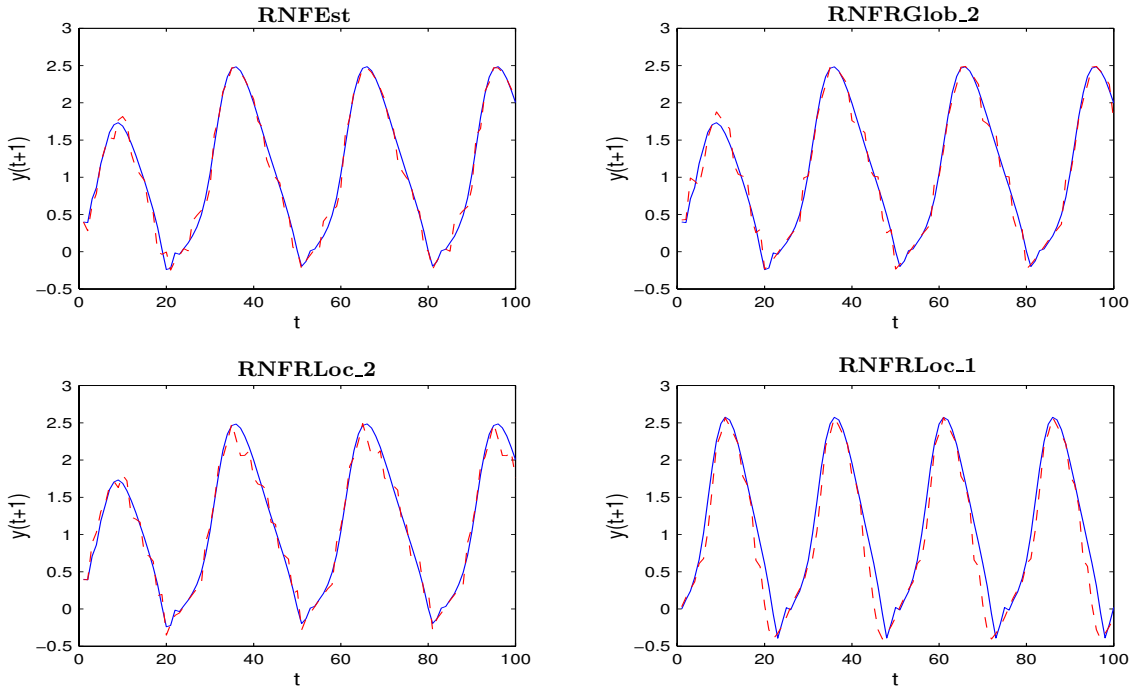


Figura 6.5: Segundo teste de identificação, Exemplo 6.2: (—) saída desejada, (· · ·) saída da rede neural.

EXEMPLO 6.3

O sistema a ser identificado é um sistema multi-variável classificado como uma versão MIMO do modelo II da caracterização de sistemas em tempo discreto. O sistema é descrito pela seguinte equação:

$$\begin{bmatrix} y_1(t+1) \\ y_2(t+1) \end{bmatrix} = \begin{bmatrix} \dfrac{y_1(t)}{1+y_2^2(t)} \\ \\ \dfrac{y_1(t)y_2(t)}{1+y_2^2(t)} \end{bmatrix} + \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \tag{6.7}$$

A abordagem utilizada será a mesma que aquela adotada para os exemplos anteriores. (Narendra & Parthasarathy 1990) utilizou duas redes neurais multicamadas para a identificação deste sistema. Neste trabalho, será utilizado somente uma única estrutura para a construção do modelo de identificação série-paralelo.

O conjunto de treinamento para a estrutura **SFA1** é formado por 4000 padrões gerados a partir da Equação (6.7), com valores aleatórios de distribuição uniforme dos sinais de entrada no intervalo $[-1.5, 1.5]$. Tentou-se utilizar um conjunto menor, mas os resultados obtidos não foram bons.

O conjunto de treinamento para os outros modelos é composto por 2000 padrões, gerados utilizando a Equação (6.7) e um sinal de entrada aleatório e com distribuição uniforme no intervalo $[-1, 1]$. Todos estes padrões foram apresentados em ordem aleatória para cada época no processo de treinamento.

O conjunto de teste para avaliar os modelos série-paralelos obtidos é composto de 100 pontos gerados utilizando o sinal de entrada $u(t)$ definido pela seguinte equação:

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = \begin{bmatrix} 0.5\ \mathrm{sen}(2\pi t/25) \\ 0.4\cos(2\pi t/25) \end{bmatrix} \tag{6.8}$$

isto para $t = 0, \ldots, 99$.

As características das redes neurais são mostradas na Tabela 6.6, onde no caso do **ANFIS**, os dados para o modelo associado saída y_1 e a saída y_2 são separados por uma "\" (dados modelo y_1 \ dados modelo y_2), devido à limitação do modelo implementado no toolbox do Matlab (*Fuzzy Logic Toolbox, For Use with MATLAB®* 2000).

A Tabela 6.6 apresenta o tipo de agrupamento utilizado em cada modelo, assim como o número de conjuntos nebulosos que conformam a partição do espaço de entrada, o número máximo de neurônios ativos na camada intermediária e o número máximo de parâmetros utilizados por iteração.

Tabela 6.6: Características das redes neurais - Exemplo 6.3.

Estrutura	n	p	Tipo de Agrupamento	N_i	Na_{max}	Parâmetros / iteração
RNEst	2	2	LVQ	[21, 23]	4	16
RNRLoc	2	2	LVQ	[16, 16]	4	20
RNFEst	2	2	UNIFORME	[35, 35]	4	16
RNFRGlob	2	2	UNIFORME	[38, 38]	4	32
RNFRLoc	2	2	UNIFORME	[34, 36]	4	20
RNRLoc_Cheng	2	2	—	[50]	50	302
SFA	2	2	KNN	[1317]	—	—
RNFRLoc_Lee	2	2	UNIFORME	[30, 30]	30	240
ANFIS	2	2	—	[9,9]\[9,9]	9\9	63\63
MLP	2	2	—	[20, 10]	[20, 10]	292

Os EQMs de treinamento e de teste, assim como o número de épocas e tempo de processamento são apresentados na Tabela 6.7.

Dos modelos propostos e as suas variações, a rede neurofuzzy com recorrência global (**RNFRLoc**) e a rede neural (**RNEst**) foram as que obtiveram melhor desempenho para as duas saías do sistema y_1 e y_2. No entanto, os outros modelos neurofuzzy conseguiram erros da ordem de 10^{-2} a 10^{-3}. Cabe observar que durante a série de execuções na busca dos melhores resultados, as estruturas recorrentes propostas conseguiram um bom desempenho para a saída y_1 antes do que para a saída y_2, sendo necessário refinar gradativamente o número de conjuntos nebulosos na partição da entrada até obter uma saída aceitável, Isto pode ser

Tabela 6.7: Erros quadráticos médios (EQM) - Exemplo 6.3.

Estrutura	Épocas	T_{proc} (s)	EQM Treinamento $(\times 10^{-3})$	EQM Teste	
				y_1 $(\times 10^{-2})$	y_2 $(\times 10^{-2})$
RNEst	1500	394	0.215	0.471	0.386
RNRLoc	500	213	0.020	1.285	1.055
RNFEst	1500	224	0.002	0.851	0.447
RNFRGlob	800	118	0.001	1.113	0.302
RNFRLoc	1600	419	0.001	0.684	0.587
RNRLoc_Cheng	400	241	0.024	0.067	0.024
SFA	4000	26	6.075	0.354	0.256
RNFRLoc_Lee	400	229	0.002	0.018	0.011
ANFIS	200\200	—	22.000 \ 2.459	0.455	0.188
MLP	500	426	0.019	0.022	0.024

justificado pelo fato de que a saída y_2 é mais complexa que a saída y_1 (de Moraes Lima 2000).

Dos modelos ajustados e testados, somente as rede **RNRLoc_Cheng**, **RNFRLoc_Lee** e **MLP** proporcionaram erros de teste na ordem de 10^{-4}. Deve-se observar que a rede multicamadas precisa de uma quantidade maior de recursos computacionais por iteração (ver Tabela 6.6).

Embora o número de conjuntos fuzzy que caracterizam a partição do espaço de entrada tenha sido elevado, deve-se considerar que um máximo de 4 neurônios estiveram ativos por iteração para as estruturas recorrentes propostas e que somente uma única rede para a identificação do sistema. Em (Narendra & Parthasarathy 1990), duas redes neurais multicamadas foram utilizadas para a identificação do mesmo sistema.

O sistema fuzzy adaptativo, de um total de 4000 padrões utilizados para o treinamento, necessitou definir 1317 funções de pertinência gaussianas para obter um bom desempenho.

A Figura 6.6 ilustra o erro de treinamento e os resultados de simulação para a rede **RNEst** e **RNFEst**. As curvas de erro e resultados de teste para as redes neurofuzzy recorrentes **RNFRGlob** e **RNFRLoc** são apresentados na Figura 6.7. Finalmente, na Figura 6.8 mostra

os resultados de simulação obtidos para as redes recorrentes **RNRLoc_Cheng** e **RNFR-Loc_Lee**.

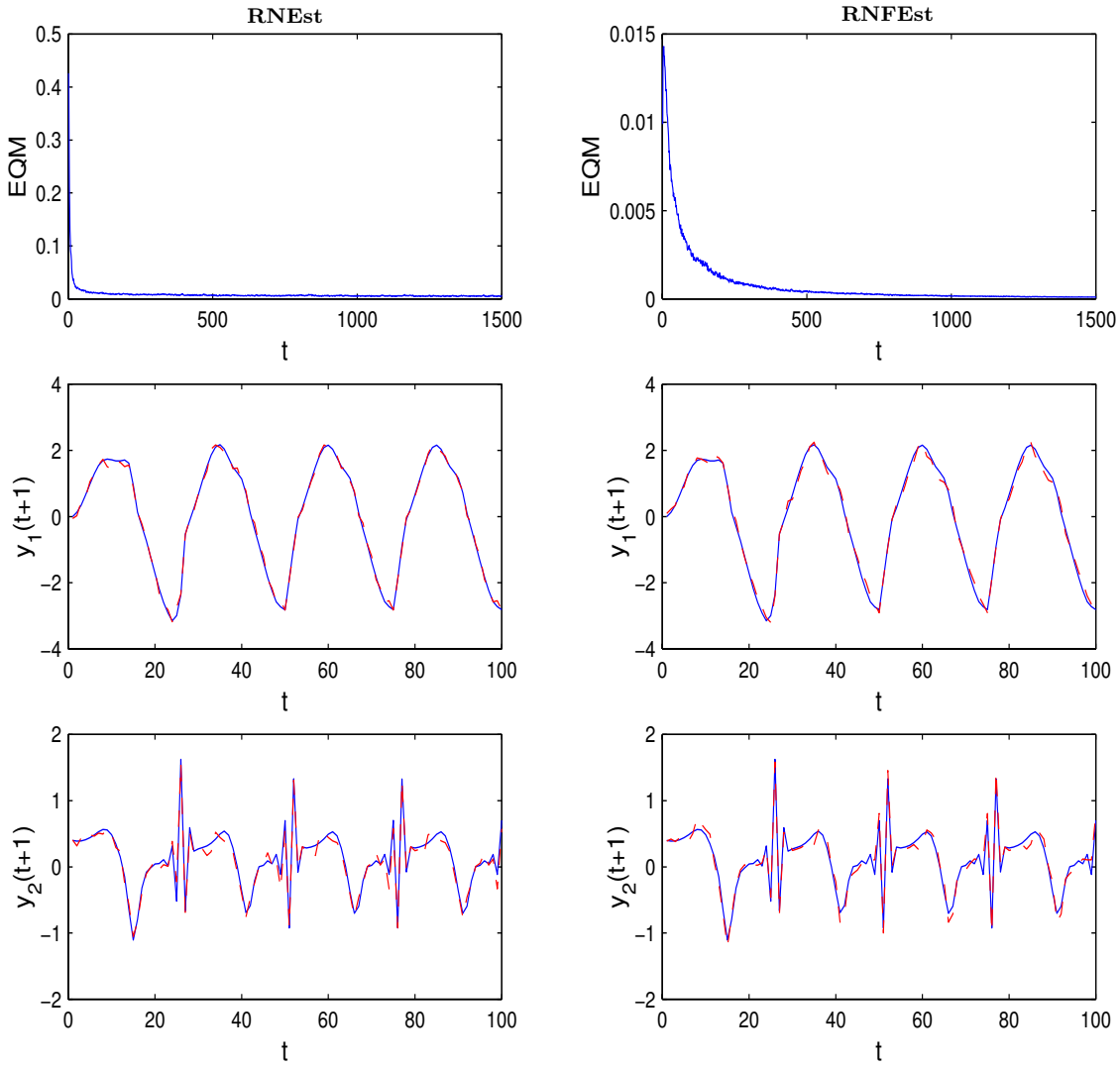


Figura 6.6: Identificação, Exemplo 6.3: (—) saída desejada, $(\cdots)$ saída da rede neural.

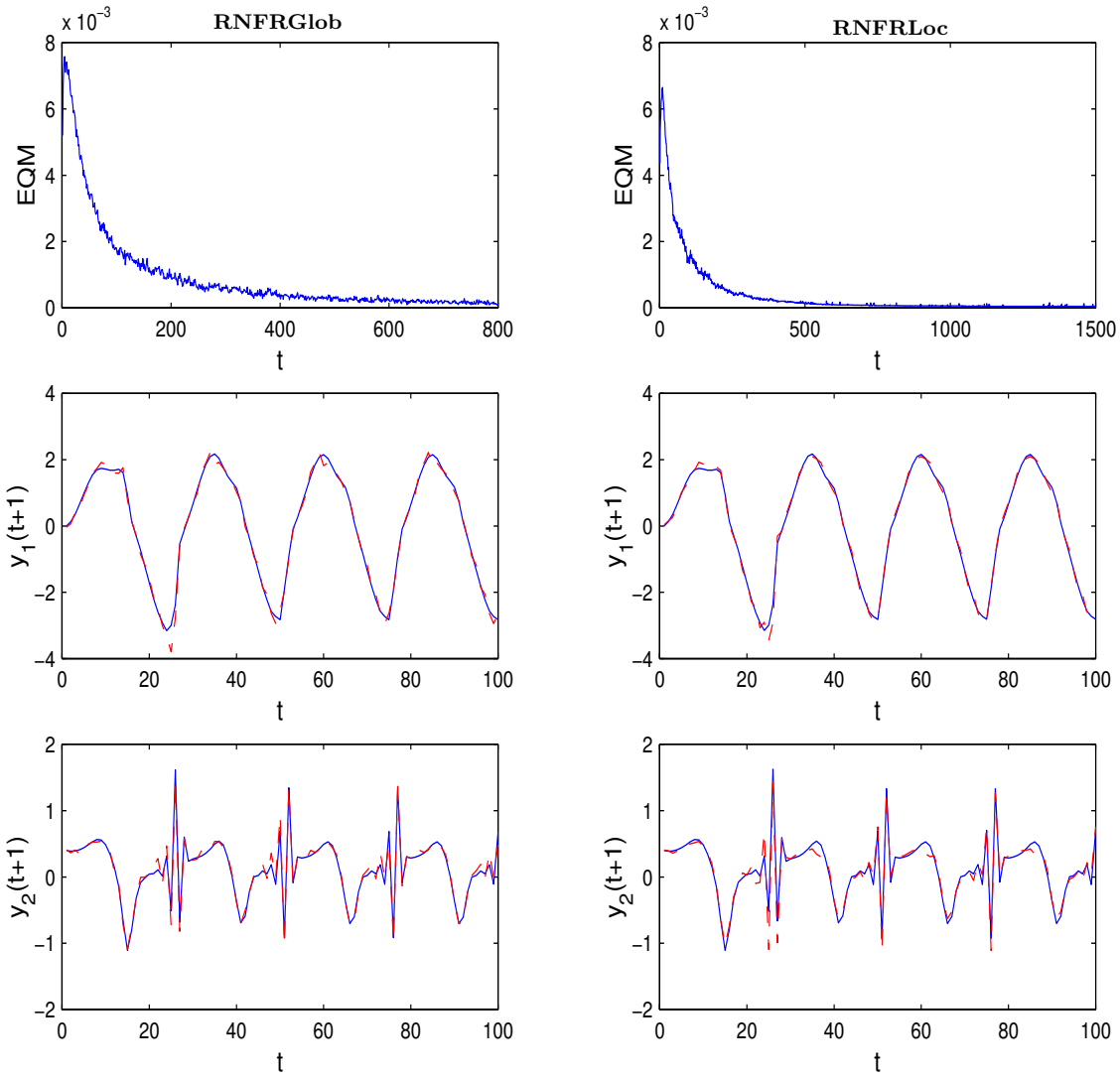


Figura 6.7: Identificação, Exemplo 6.3: (—) saída desejada, ($\cdots$) saída da rede neural.

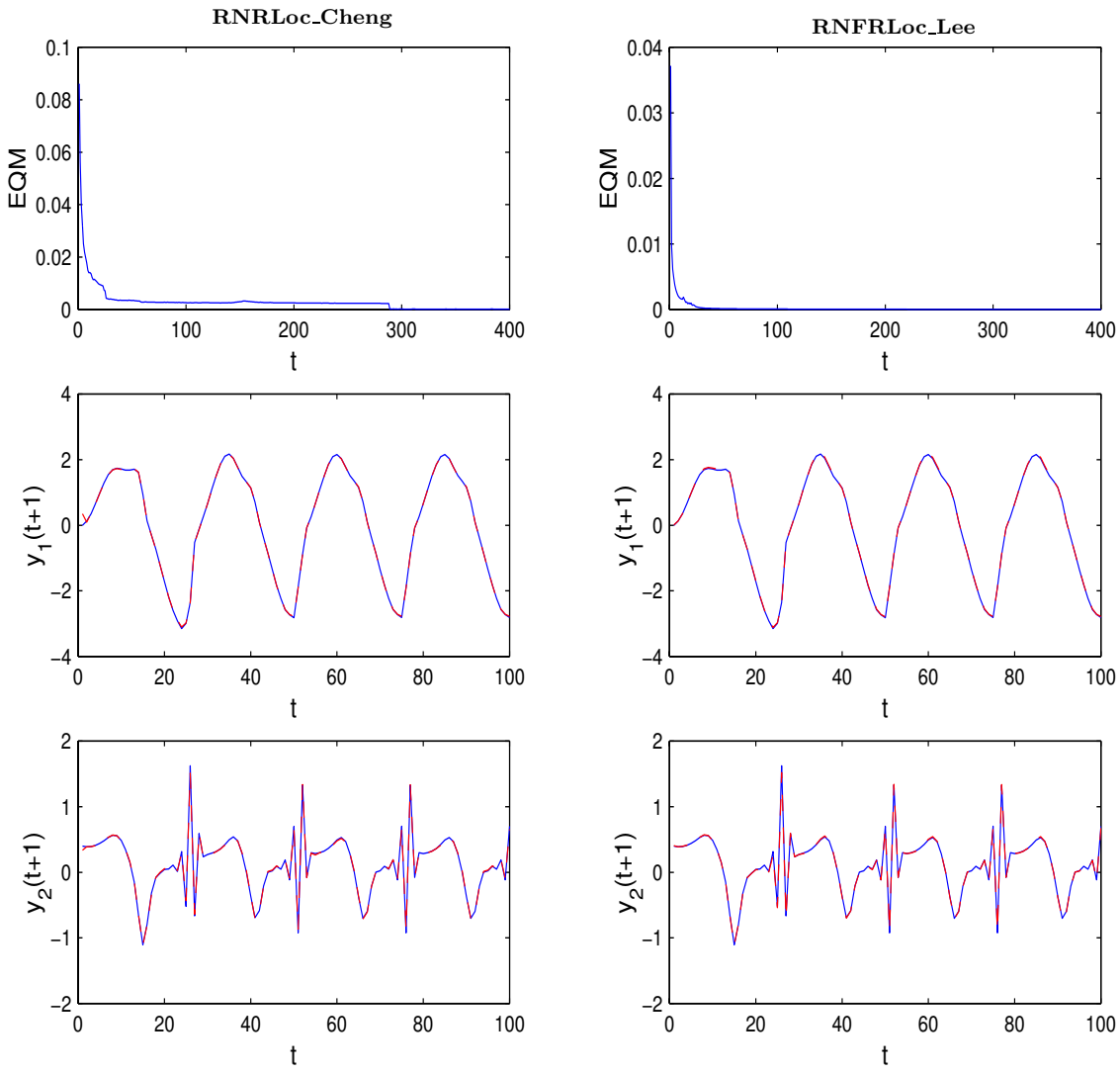


Figura 6.8: Identificação, Exemplo 6.3: (—) saída desejada, $(\cdots)$ saída da rede neural.

EXEMPLO 6.4

Neste caso o modelo é dado por uma equação à diferenças de terceira ordem, correspondendo ao modelo IV da caracterização dos sistemas. O sistema a modelar é definido pela equação:

$$y(t+1) = f[y(t),\ y(t-1),\ y(t-2),\ u(t),\ u(t-1)] \tag{6.9}$$

onde a função $f(\cdot)$ suposta desconhecida é definida como:

$$f[x_1,\ x_2,\ x_3,\ x_4,\ x_5] = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_3^2 + x_2^2} \tag{6.10}$$

Vários testes foram realizados utilizando um conjunto inicial de treinamento formado por 2000 pontos gerados aleatoriamente no intervalo $[-2, 2]$ e com distribuição uniforme. Os resultados não foram os esperados. Como um tentativa de excitar todos os modos do sistema, um novo conjunto de treinamento foi gerado para todas as estruturas. Este segundo conjunto de teste foi composto por 2400 pontos, sendo os 800 primeiros gerados aleatoriamente com distribuição uniforme, no intervalo $[-1.0, 1.0]$. Os próximos pontos foram gerados através da função $u(t) = \sin(2\pi t/250)$. As condições iniciais tanto para o conjunto de treinamento como para os conjuntos de teste foram $y(0) = y(-1) = y(-2) = u(-1) = 0.0$.

Embora se acrescente mais padrões no conjunto de treinamento para **SFA**, não é possível obter melhoras no seu desempenho.

As entradas utilizadas para cada rede são descritas na Tabela 6.8. A Tabela 6.9 caracteriza os parâmetros que definem a complexidade das estruturas avaliadas, assim como o número de parâmetros ativos por iteração.

Tabela 6.8: Entradas das redes neurais - Exemplo 6.4.

Estrutura	n	Entradas Utilizadas
RNEst	5	$y(t)$, $y(t-1)$, $y(t-2)$, $u(t)$, $u(t-1)$
RNRLoc	1	$u(t)$
RNFest	5	$y(t)$, $y(t-1)$, $y(t-2)$, $u(t)$, $u(t-1)$
RNFRGlob	1	$u(t)$
RNFRLoc	1	$u(t)$
RNRLoc_Cheng	1	$u(t)$
SFA	5	$y(t)$, $y(t-1)$, $y(t-2)$, $u(t)$, $u(t-1)$
RNFRLoc_Lee_5	5	$y(t)$, $y(t-1)$, $y(t-2)$, $u(t)$, $u(t-1)$
RNFRLoc_Lee_2	2	$y(t)$, $u(t)$
ANFIS	5	$y(t)$, $y(t-1)$, $y(t-2)$, $u(t)$, $u(t-1)$
MLP	5	$y(t)$, $y(t-1)$, $y(t-2)$, $u(t)$, $u(t-1)$

Tabela 6.9: Características das redes neurais - Exemplo 6.4.

Estrutura	n	p	Tipo de Agrupamento	N_i	Na_{max}	Parâmetros / iteração
RNEst	5	1	UNIFORME	[3,3,3,3,3]	32	192
RNRLoc	1	1	UNIFORME	[20]	2	6
RNFEst	5	1	UNIFORME	[7,7,7,6,6]	32	192
RNFRGlob	1	1	UNIFORME	[30]	2	8
RNFRLoc	1	1	UNIFORME	[30]	2	6
RNRLoc_Cheng	1	1	—	—	50	201
SFA	5	1	KNN	[1964]	—	—
RNFRLoc_Lee_5	5	1	—	[30,30,30,30,30]	30	480
RNFRLoc_Lee_2	2	1	—	[30,30]	30	210
ANFIS	5	1	—	[7,7,7,7,7]	7	112
MLP	5	1	—	[30,20]	[30,20]	821

As taxas de aprendizado, para os modelos propostos foi de $\eta = 0.5$ para os pesos da camada de saída e para os pesos da camada intermediária e de recorrência, taxas com valores $\alpha_1 = \alpha_3 = 0.001$ e $\alpha_2 = \alpha_4 = 0.0001$.

Uma vez treinados os modelos computacionais, estes foram testados com dois conjuntos de teste diferentes, ambos compostos por 1000 pontos, mas gerados utilizando sinais diferentes de entrada $u(t)$. O primeiro conjunto de teste foi gerado considerando um sinal de entrada

$u(t)$ definido pela equação:

$$u(t) = u_1(t) = 0.8\text{sin}(2\pi t/250) + 0.2\text{sin}(2\pi t/25); \tag{6.11}$$

O segundo conjunto de teste foi gerado utilizando um sinal de entrada definido como:

$$u(t) = u_2(t) = \begin{cases} \text{sin}(\pi t/25) & t <= 250, \\ 0.2 & t > 250 \text{ e } t <= 500 \\ -0.2 & t > 500 \text{ e } t <= 700 \\ 0.3\text{sin}(\pi t/25) + 0.1\text{sin}(\pi t/32) + 0.6\text{sin}(\pi t/10) & t > 700. \end{cases} \tag{6.12}$$

A Tabela 6.10 mostra os erros quadráticos médios (EQM) de treinamento, os erros de teste e os tempos de processamento (T_{proc}) para o treinamento de cada uma das redes consideradas neste trabalho.

Tabela 6.10: Erros quadráticos médios (EQM) - Exemplo 6.4.

Estrutura	Épocas	T_{proc} (s)	EQM Treinamento $(\times 10^{-3})$	EQM Teste1 $(\times 10^{-2})$	EQM Teste2 $(\times 10^{-2})$
RNEst	180	285	0.002	0.027	0.055
RNRLoc	120	56	0.001	0.004	0.069
RNFEst	31	24	0.000	0.048	0.562
RNFRGlob	800	222	0.000	0.036	0.070
RNFRLoc	800	193	0.000	0.036	0.067
RNRLoc_Cheng	600	256	0.002	0.059	0.104
SFA	2000	9	2.722	0.078	0.127
RNFRLoc_Lee_5	500	600	0.397	0.405	0.168
RNFRLoc_Lee_2	500	983	0.008	0.130	0.170
ANFIS	500	—	16.096	0.610	0.670
MLP	500	1021	0.002	0.021	0.027

A análise das Tabelas 6.9 e 6.10 sugere que, em termos de um compromisso entre complexidade e desempenho, as redes neurofuzzy recorrentes híbridas propostas são tão ou mais eficientes que as outras estruturas avaliadas (Luna, Ballini & Gomide 2003a).

Todos os modelos estáticos mostraram um desempenho satisfatório, pois ordem dos erros de teste de todos estes modelos foi em geral da ordem de 10^{-4}. Todos estes modelos utilizaram as cinco entradas necessárias e utilizadas na equação 6.9. Já os modelos recorrentes, com exceção do modelo **RNFRLoc_Lee_5** e **RNFRLoc_Lee_2**, utilizaram uma única entrada, como indica a Tabela 6.8.

O modelo neurofuzzy proposto apresentou erros de teste baixos, precisando para isto do menor número de épocas entre todas as estruturas. Embora a rede **MLP** tenha conseguido menores erros de teste, esta precisou de um maior período de treinamento e de um número maior de neurônios e parâmetros a serem atualizados por iteração (821 parâmetros) quando comparado com o modelo **RNFEst** (192 parâmetros), como indica a Tabela 6.9. Claramente, o número de parâmetros no caso das redes neurofuzzy propostas é substancialmente menor, com um custo computacional baixo e estruturas mais simples para o modelo.

O modelo neurofuzzy proposto em (Lee & Teng 2000) apresentou erros de teste na ordem de 10^{-3}, utilizando duas e cinco entradas, com um total de 30 partições para cada uma destas entradas utilizadas. Os resultados obtidos come esta rede utilizando uma única entrada não foram os esperados. Além disso, a quantidade de parâmetros que esta rede utiliza por iteração, assim como a quantidade de neurônios ativos, é superior à necessidade de recursos computacionais de quaisquer modelo recorrente proposto e testado.

Devido a que os modelos neurofuzzy recorrentes propostos utilizaram una única entrada, estes forneceram modelos simples e eficientes para a modelagem do sistema definido pela Equação 6.9. Outros testes considerando um número maior de entradas para as redes neurofuzzy propostas foram feitos. Contudo, o ganho em precisão não justificou o crescimento exponencial das estruturas resultantes.

Os erros de treinamento, assim como os erros obtidos durante o primeiro teste, são apresentados na Figura 6.9 para os modelos **RNFEst** e **RNFRGlob**. A Figura 6.10 ilustra os resultados de treinamento e do primeiro teste para a rede **RNFRLoc** e **RNEst**. Os resultados de simulação utilizando o segundo conjunto de teste são ilustrados na Figura 6.11.

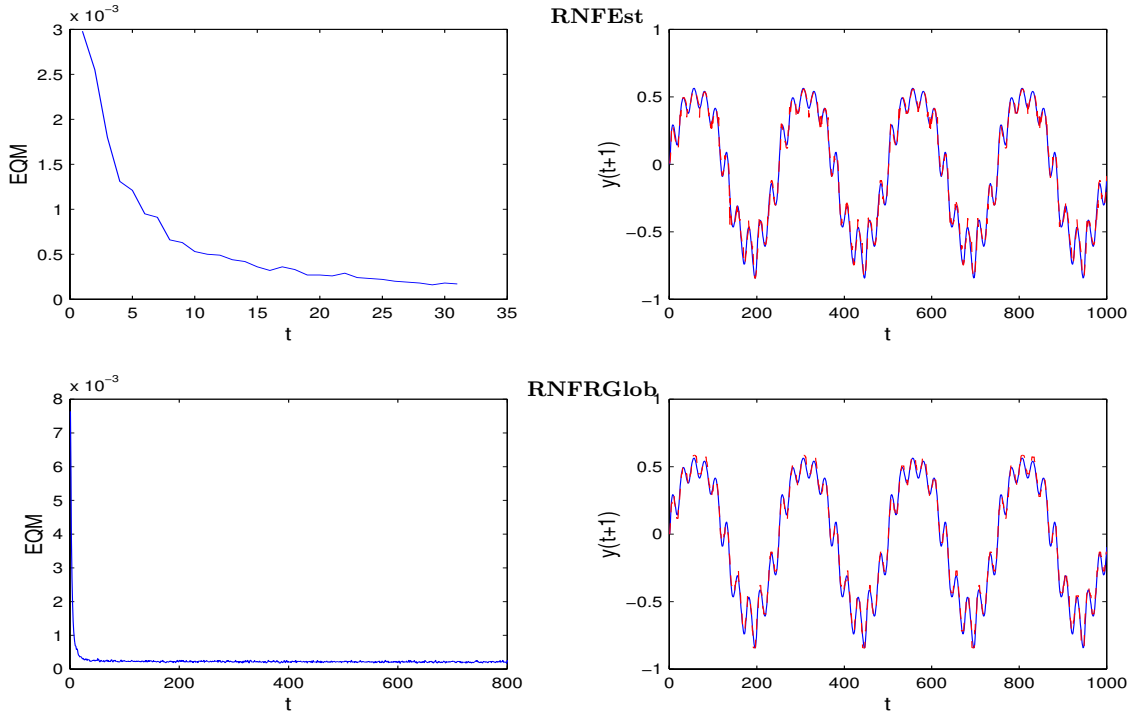


Figura 6.9: Primeiro teste de identificação, Exemplo 6.4: (—) saída desejada, (· · ·) saída da rede neural.

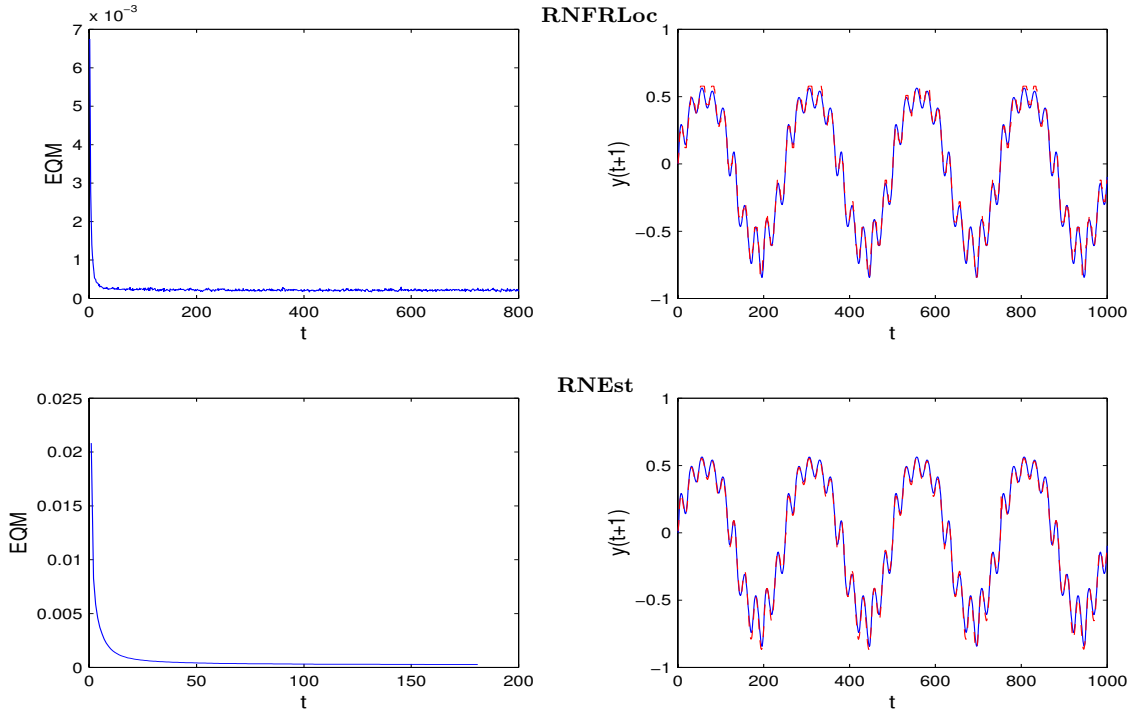


Figura 6.10: Primeiro teste de identificação, Exemplo 6.4: (—) saída desejada, (· · ·) saída da rede neural.

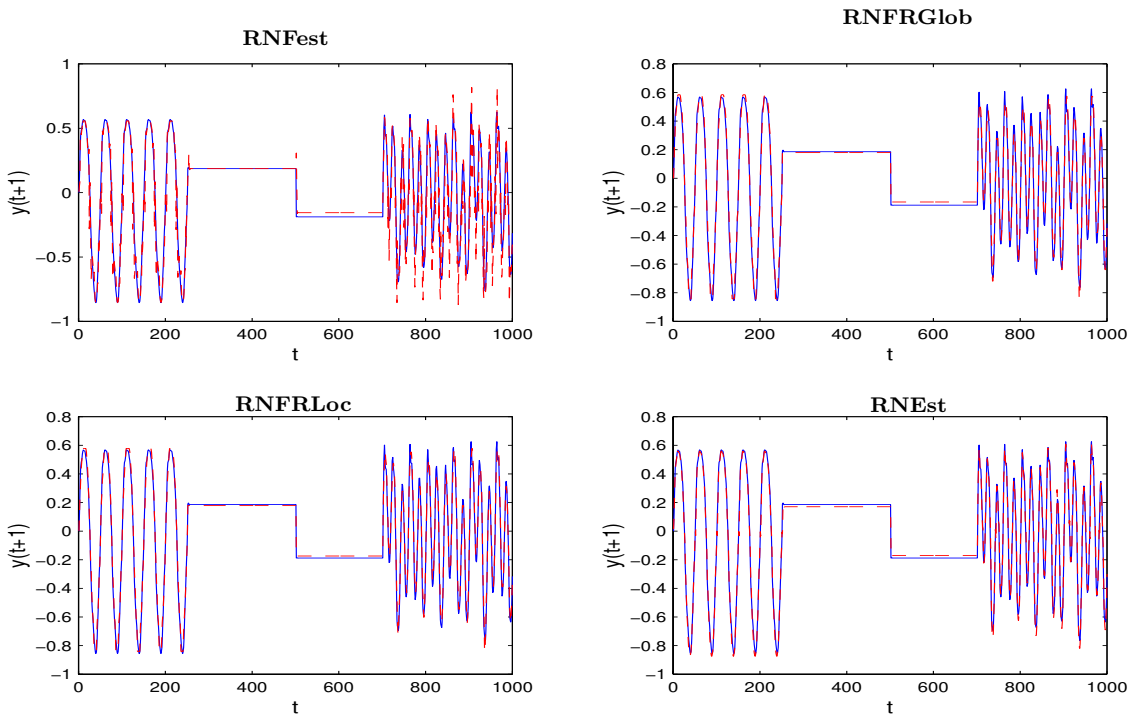


Figura 6.11: Segundo teste de identificação, Exemplo 6.4: (—) saída desejada, (· · ·) saída da rede neural.

6.2 Controle de Sistemas

Como foi mencionado na Seção 5.2.3, o método de controle a ser utilizado nos estudos de simulação é modelo de referência. Este modelo também foi utilizado em (Narendra & Parthasarathy 1990) e (Wang 1994) e consta de três etapas: identificação das não linearidades do sistema, controle *off-line* e controle *on-line*, sendo estas etapas resumidas a seguir:

1. ***Identificação***: o processo de identificação segue a metodologia descrita na Seção 5.2.2. Os modelos de redes neurais, sistemas fuzzy e sistemas neurofuzzy foram treinados utilizando um sinal de entrada escolhido, de tal forma a excitar todos os modos do sistema.

2. ***Controle off-line***: nesta etapa, o sistema em malha fechada é construido utilizando a estrutura obtida durante o treinamento na primeira etapa. Neste caso, os pesos das redes neurais são mantidos fixos.

3. ***Controle on-line***: na terceira etapa, é realizado um ajuste fino dos pesos das redes neurais, via um processo de identificação e controle simultâneo, utilizando taxas de aprendizagem baixas durante o ajuste e tomando como como valores iniciais, os pesos obtidos no treinamento *off-line*.

O modelo de controle em malha fechada para os sistemas é ilustrado na Figura 6.12.

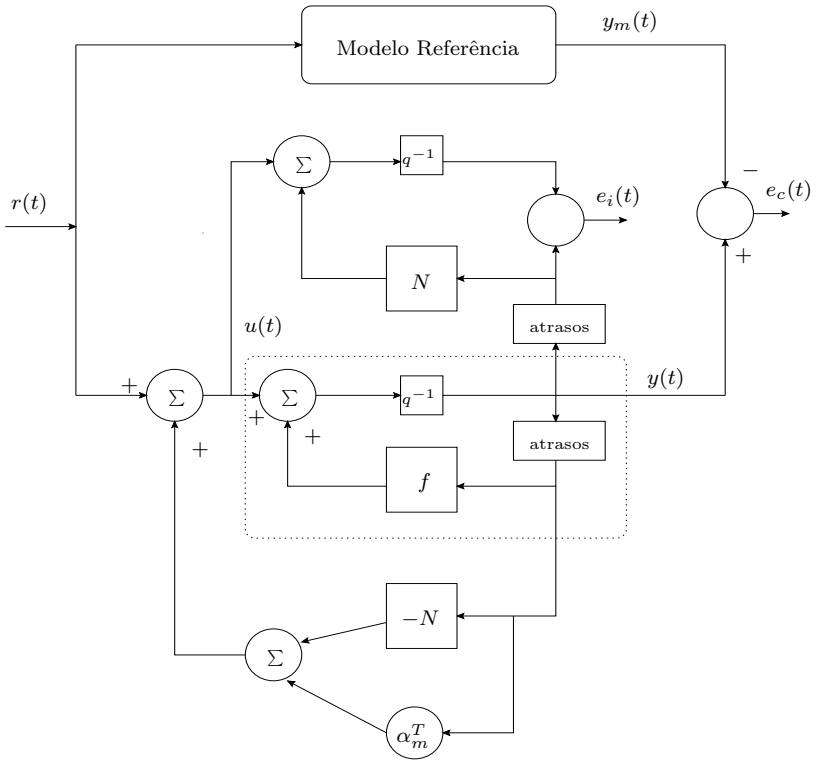


Figura 6.12: Sistema neural de controle.

A forma escolhida para identificar cada sistema não linear permite a aplicação de técnicas simples de controle de sistemas dinâmicos, como será mostrado a seguir. Para isto, foram escolhidos dois exemplos de sistemas não lineares identificados na Seção 6.1, ou seja, foram considerados os exemplos 6.2 e 6.3, respectivamente.

EXEMPLO 6.5

Este sistema é definido pelas Equações (6.3) e (6.4) repetidas abaixo por conveniência:

$$y(t+1) = f[y(t),\ y(t-1)] + u(t) \tag{6.13}$$

onde,

$$f[y(t),\ y(t-1)] = \frac{y(t)y(t-1)[y(t)+2.5]}{1+y^2(t)+y^2(t-1)} \tag{6.14}$$

Durante o processo de identificação, assume-se $f(\cdot)$ desconhecida. O modelo de referência utilizado para o controle é definido pela equação à diferenças de segunda ordem:

$$y_m(t+1) = 0.6y_m(t) + 0.2y_m(t-1) + r(t) \tag{6.15}$$

sendo $r(t)$ o sinal de entrada do sistema. O erro de controle $e_c(t+1)$, utilizado para avaliar o desempenho *off-line*, assim como para o treinamento *on-line*, é definido como a diferença entre a saída real do sistema $y(t+1)$ e o sinal de referência $y_m(t+1)$, isto é:

$$e_c(t+1) = y(t+1) - y_m(t+1) \tag{6.16}$$

O objetivo do controlador é o de determinar um sinal de controle $u_c(t)$ de tal forma a minimizar $e_c(t)$, isto é, fazer $\lim_{t\to\infty} e_c(t) = 0$. Se $f(\cdot)$ (Equação (6.14)) fosse conhecida, o sinal de controle poderia ser determinado como:

$$u(t) = -f[y(t),\ y(t-1)] + 0.6y(t) + 0.2y(t-1) + r(t) \tag{6.17}$$

Substituindo a Equação (6.17) na Equação (6.13) e utilizando a equação resultante na Equação (6.15), para o cálculo do erro de controle, tem-se:

$$e_c(t+1) = 0.6e_c(t) + 0.2e_c(t-1) \tag{6.18}$$

Como o modelo de referência é, por hipótese, assintoticamente estável, tem-se que $\lim_{t\to\infty} e_c(t) = 0$ para condições iniciais arbitrárias (Narendra & Parthasarathy 1990).

Quando $f(\cdot)$ não é conhecida, deve-se utilizar o modelo obtido pelo método de identificação, ou seja, utilizar a rede neurofuzzy determinada na primeira etapa. Assim, se $N[\cdot]$ é a estimação de $f(\cdot)$ proporcionada pela rede neural, a Equação (6.17) pode ser re-escrita como:

$$u(t) = -N[\cdot] + 0.6y(t) + 0.2y(t-1) + r(t) \tag{6.19}$$

Substituindo a Equação (6.19) na Equação (6.3), obtém-se a equação à diferenças, que governa o comportamento do sistema em malha fechada, isto é:

$$y(t+1) = f[y(t),\ y(t-1)] - N[\cdot] + 0.6y(t) + 0.2y(t-1) + r(t) \tag{6.20}$$

A Figura 6.13 mostra a saída do sistema em malha aberta e malha fechada para:

$$r(t) = r_1(t) = 0.5\text{sen}(2\pi t/25)$$

A Figura 6.14 mostra a saída do sistema em malha aberta e malha fechada para

$$r(t) = r_2(t) = 0.4\cos(2\pi t/30)$$

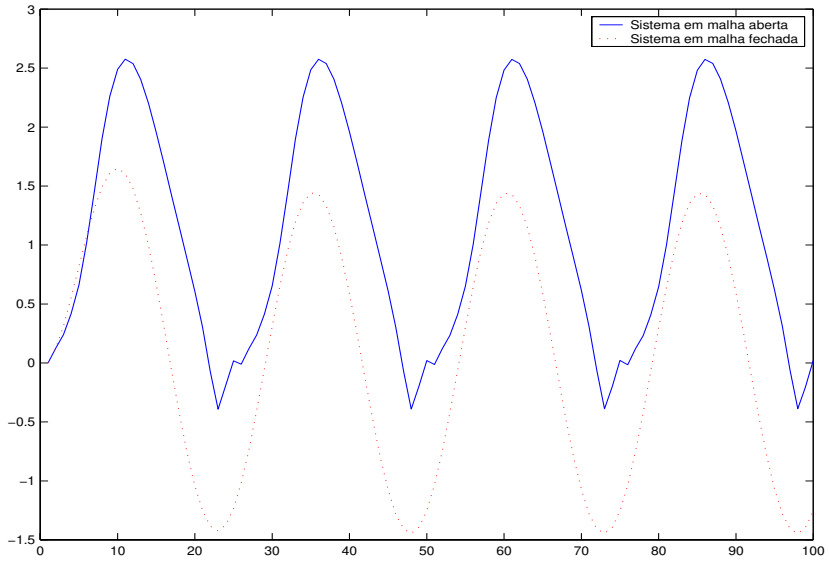


Figura 6.13: Exemplo 6.5: saída do sistema em malha fechada para $r_1(t)$.

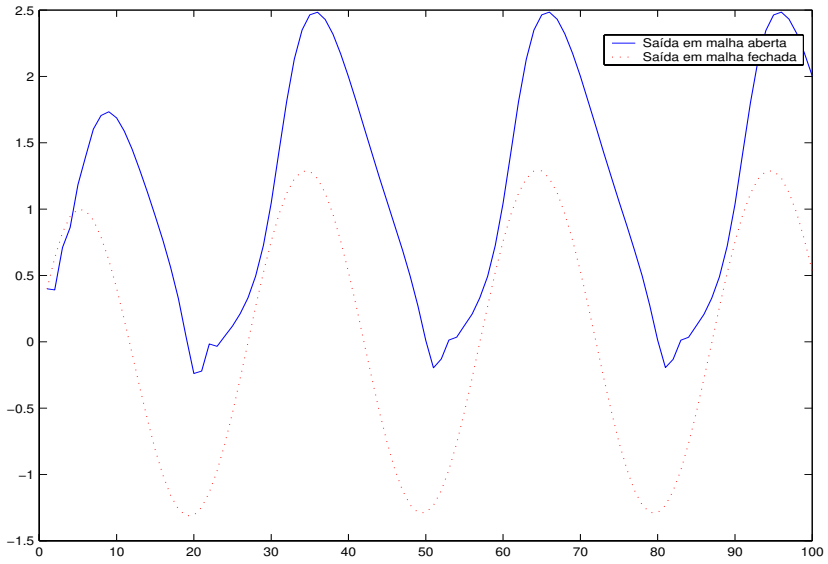


Figura 6.14: Exemplo 6.5: saída do sistema em malha fechada para $r_2(t)$.

As entradas utilizadas para a identificação foram aquelas apresentadas na Tabela 6.3, Seção 5.4.1; estas entradas foram as mesmas utilizadas durante o processo de controle. O número de épocas para o ajuste *off-line* dos modelos de redes neurais são aqueles mostrados na Tabela 6.5, Seção 5.4.1. A Tabela 6.11 apresenta o número de iterações durante o treinamento *on-line* das arquiteturas avaliadas, assim como o tempo de processamento e os erros quadráticos médios de teste (EQM), tanto para o controle *off-line* como para o controle *on-line*. No caso do **ANFIS**, não foi possível efetuar o refinamento *on-line* dos modelos obtidos *off-line*.

Analisando a Tabela 6.11, verifica-se que as redes neurofuzzy propostas e as suas variações apresentaram um bom desempenho tanto no controle *off-line* como no controle *on-line*, isto considerando tanto os erros de teste, assim como tempos de processamento e uso de recursos computacionais. Todas as redes neurais se mostraram melhores uma vez feito o refinamento dos seus pesos.

As redes neurofuzzy propostas atingiram erros de teste na ordem de 10^{-3} tanto no controle *off-line* como no controle *on-line*. Os tempos de processamento também mostraram-se mínimos (Tabela 6.11).

A rede neurofuzzy proposta em (Lee & Teng 2000) também obteve o desempenho esperado. Porém, utilizando uma única entrada, esta apresentou erros de teste maiores quando comparado com a rede neurofuzzy de recorrência local e com uma única entrada (Tabela 6.11). Já o desempenho da rede **RNFRLoc_Lee_1** é melhorado após o treinamento *on-line*.

Tabela 6.11: Erros quadráticos médios (EQM) - Exemplo 6.5.

Estrutura	Iterações on-line	T_{proc} (s)	EQM Treinamento on-line $(\times 10^{-4})$	EQM Teste1 off-line	EQM Teste1 on-line	EQM Teste2 off-line	EQM Teste2 on-line
				$(\times 10^{-2})$			
RNEst	500	0.008	0.089	0.403	0.002	0.001	0.001
RNRLoc_2	12000	2	0.073	0.550	0.002	0.003	0.002
RNFEst	20000	0.054	0.000	0.536	0.004	0.005	0.002
RNFRGlob_2	20000	4	0.001	0.454	0.002	0.005	0.003
RNFRLoc_2	4000	0.062	0.001	0.639	0.004	0.008	0.003
RNFRLoc_1	10000	1	0.020	7.462	0.009	0.087	0.009
RNRLoc_Cheng	40000	11	6.208	0.042	0.001	0.000	0.000
SFA	300	2	31.000	0.426	0.003	0.006	0.003
RNFRLoc_Lee_2	40000	19	0.000	0.016	0.001	0.000	0.001
RNFRLoc_Lee_1	40000	17	0.389	16.016	0.004	0.167	0.003
ANFIS	—	—	—	0.098	—	0.052	—
MLP	10000	4	0.000	0.001	0.000	0.001	0.000

O sistema fuzzy adaptativo **SFA**, para o ajuste *on-line*, precisou de gerar novos centros, considerando a saída do sistema em malha fechada; foram geradas 197 funções de pertinência, com centros diferentes dos 70 obtidos no treinamento *off-line*.

Cabe ressaltar que o sistema de controle aqui considerado, diferentemente das outras técnicas, requer uma única rede neurofuzzy para efetuar o controle, fornecendo um sistema em malha fechada com um desempenho aceitável e exigências computacionais menores, como foi mostrado durante a obtenção das estruturas neurofuzzy no processo de identificação.

Os resultados dos teste obtidos para as estruturas neurofuzzy propostas são ilustrados nas Figuras 6.15 para a rede **RNFEst**, 6.16 para a rede **RNFRGlob_2** e 6.17 para a rede **RNFRLoc_2**.

O primeiro teste para o exemplo atual, corresponde a um sinal de entrada $u(t) = r_1(t) = 0.5\text{sen}(2\pi t/25)$. Já o segundo teste corresponde a um sinal de entrada $u(t) = r_2(t) = 0.4\cos(2\pi t/30)$, com $t = 0, \ldots, 99$.

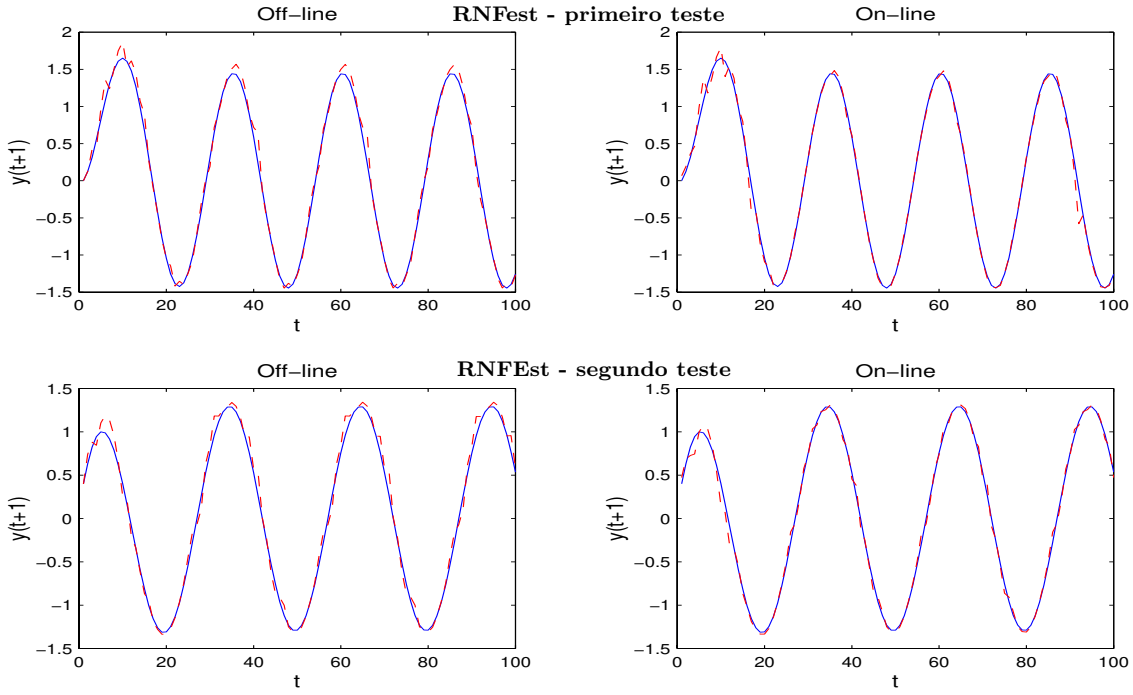


Figura 6.15: Controle, Exemplo 6.5: (—) saída desejada, $(\cdots)$ saída do sistema.

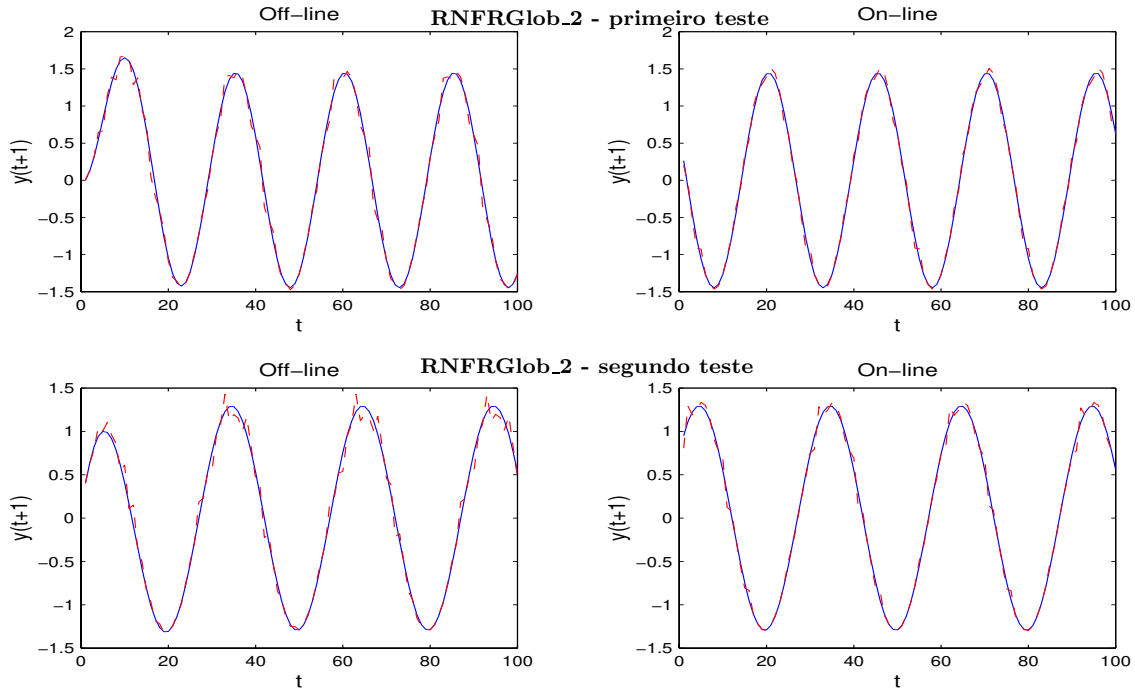


Figura 6.16: Controle, Exemplo 6.5: (—) saída desejada, ($\cdots$) saída do sistema.

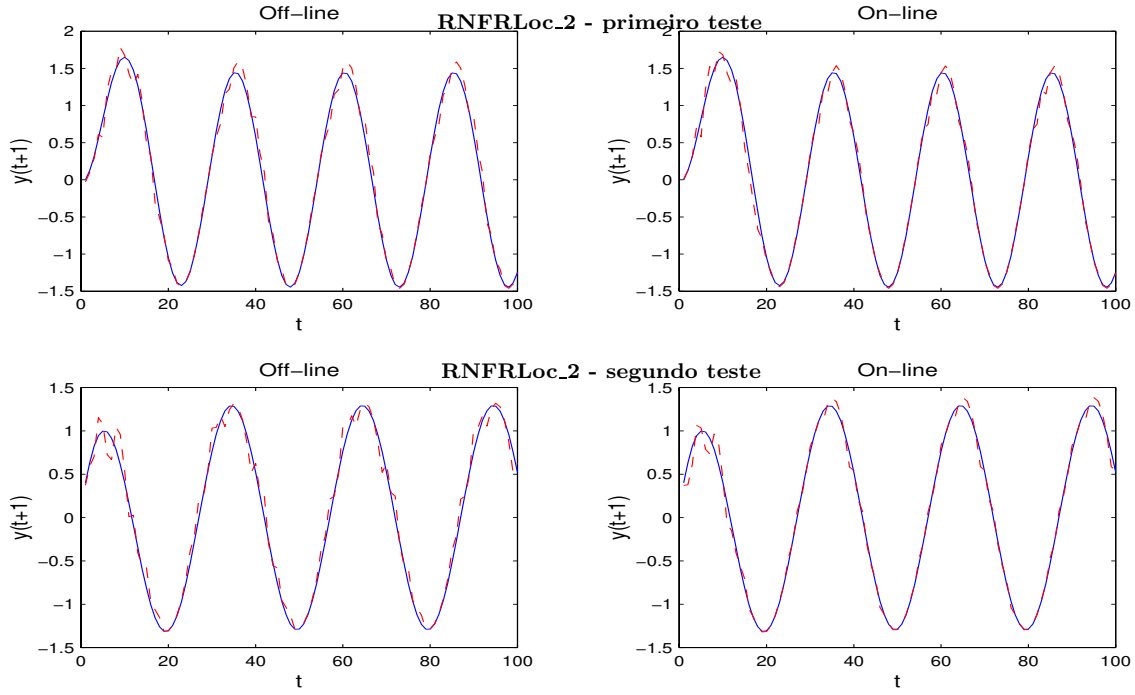


Figura 6.17: Controle, Exemplo 6.5: (—) saída desejada, ($\cdots$) saída do sistema.

EXEMPLO 6.6

A dinâmica deste sistema é definido pela Equação (6.7) repetida abaixo:

$$\begin{bmatrix} y_1(t+1) \\ y_2(t+1) \end{bmatrix} = \begin{bmatrix} \dfrac{y_1(t)}{1+y_2^2(t)} \\ \\ \dfrac{y_1(t)y_2(t)}{1+y_2^2(t)} \end{bmatrix} + \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \tag{6.21}$$

O modelo de referência linear é descrito pelas equações a diferenças:

$$\begin{bmatrix} y_{m1}(t+1) \\ y_{m2}(t+1) \end{bmatrix} = \begin{bmatrix} 0.6 & 0.2 \\ 0.1 & -0.8 \end{bmatrix} \begin{bmatrix} y_{m1}(t) \\ y_{m2}(t) \end{bmatrix} + \begin{bmatrix} r_1(t) \\ r_2(t) \end{bmatrix} \tag{6.22}$$

sendo $r_1(t)$ e $r_2(t)$ os sinais de entrada. No processo de identificação, considera-se uma única rede neural com duas entradas e duas saídas. O ajuste *on-line* dos pesos é iniciado uma vez terminado o treinamento *off-line*. A Figura 6.18, ilustra as saídas do sistema em malha aberta e fechada.

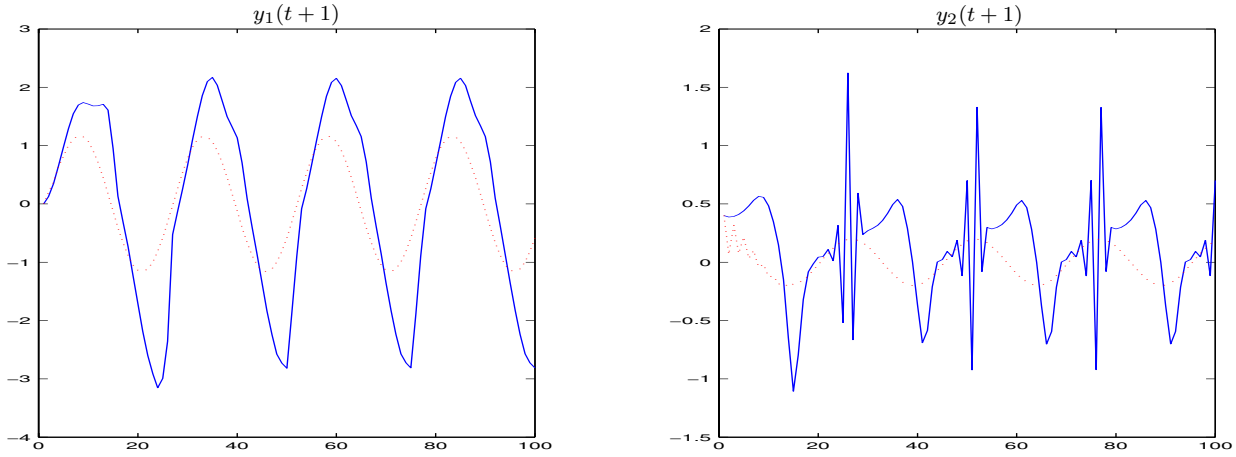


Figura 6.18: Exemplo 6.6, saídas y_1 e y_2: (—) malha aberta, $(\cdots)$ malha fechada.

Após o treinamento *off-line*, o modelo identificado para o sistema é colocado em malha fechada e testado, obtendo assim, um erro quadrático médio de teste para o controle *off-line*. Quando o ajuste *on-line* do modelo obtido pelo processo de identificação do sistema é feito, o

sistema em malha fechada é testado novamente, conseguindo desta forma, um erro quadrático médio de teste *on-line*, para cada uma das saídas do sistema.

A Tabela 6.12, apresenta o número de iterações que foram necessários para o ajuste *on-line* de cada uma das redes neurais, assim como tempos de processamento e os EQMs de teste. Os testes foram feitos considerando um sinal de entrada $r_1(t) = 0.5\ \text{sen}(2\pi t/25)$ e $r_2(t) = 0.4\text{cos}(2\pi t/25)$.

Tabela 6.12: Erros quadráticos médios (EQM) - Exemplo 6.6.

Estrutura	Iterações on-line	T_{proc}	EQM Treinamento on-line	EQM Teste off-line		EQM. Teste on-line	
				y_1	y_2	y_1	y_2
		(s)	$(\times 10^{-4})$	$(\times 10^{-2})$			
RNEst	20000	3	0.085	0.467	0.303	0.067	0.033
RNRLoc	10000	1	0.010	0.470	0.059	0.134	0.024
RNFEst	40000	7	0.005	0.969	0.179	0.747	0.039
RNFRGlob	40000	24	0.000	0.449	0.085	0.489	0.075
RNFRLoc	5000	0.9	0.000	0.358	0.079	0.180	0.042
RNRLoc_Cheng	40000	12	0.000	0.004	0.003	0.000	0.000
SFA	8000	99	20.000	0.152	0.006	0.123	0.009
RNFRLoc_Lee	40000	17	0.000	0.004	0.002	0.000	0.000
ANFIS	—	—	—	0.048	0.135	—	—
MLP	100000	3	0.000	0.001	0.000	0.004	0.000

As estruturas híbridas recorrentes propostas alcançaram um EQM de teste na ordem de 10^{-3}, tanto no treinamento *off-line* como no treinamento *on-line*. Entre estes modelos, a rede **RNEst** e a rede recorrente **RNFRLoc** conseguiram o melhor desempenho em controle, embora todos os outros modelos tenham conseguido um desempenho similar na identificação. Alguns picos obtidos no controle *off-line* foram suavizados no controle *on-line*.

O modelo de controlador fornecido pela rede neurofuzzy **RNFRLoc_Lee** mostrou um ótimo desempenho, precisando para isto, ajustar 240 e 400 parâmetros por iteração. As redes neurofuzzy propostas ajustam 16, 20 e 32 parâmetros por iteração (**RNFEst**, **RNFRLoc** e **RNFRGlob**), como mostra a Tabela 6.6. Isto demonstra que a relação entre custo computacional e desempenho é forte e se faz necessária a procura de um equilíbrio entre ambos.

O sistema fuzzy adaptativo, da mesma forma que nos exemplos anteriores, precisou gerar novos centros para ser aplicado como controlador no sistema em malha fechada. Para isto, durante o ajuste *on-line* dos 8000 padrões utilizados, o sistema necessitou definir um total de 2317 novos centros, isto é, necessitou de 2317 partições do espaço de entrada, sendo cada partição associada a uma função de pertinência gaussiana.

A Figura 6.19 ilustra os resultados de controle *off-line* e *on-line* obtidos para a rede **RNEst**. Os resultados obtidos para a rede neurofuzzy estática **RNFEst** são apresentados na Figura 6.20. Os resultados de simulação obtidos para a rede neurofuzzy recorrente **RNFRLoc**, são apresentados na Figura 6.21.

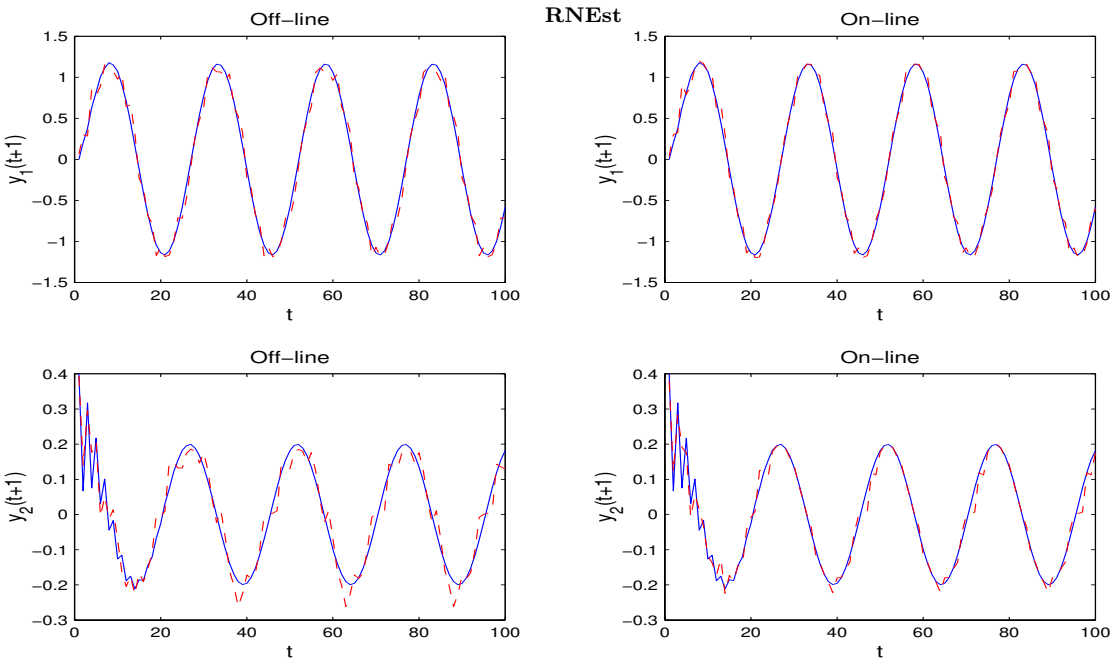


Figura 6.19: Exemplo 6.6, primeira coluna com $r_1(t)$ e segunda coluna com $r_2(t)$: (—) saída desejada, $(\cdots)$ saída do sistema.

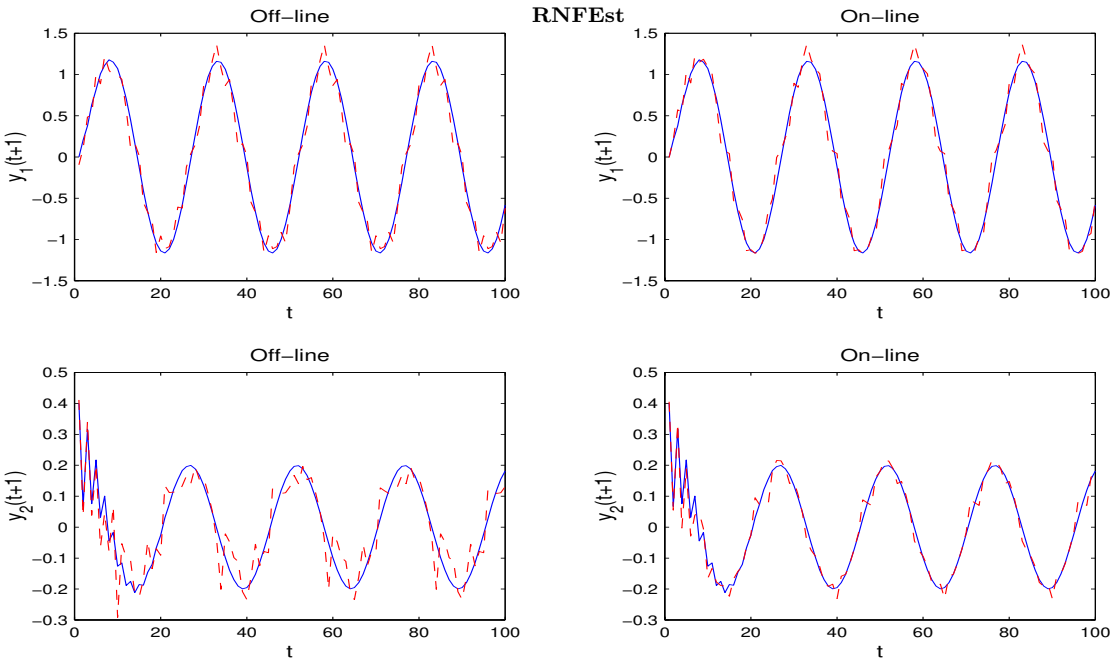


Figura 6.20: Exemplo 6.6, primeira coluna com $r_1(t)$ e segunda coluna com $r_2(t)$: (—) saída desejada, $(\cdots)$ saída do sistema.

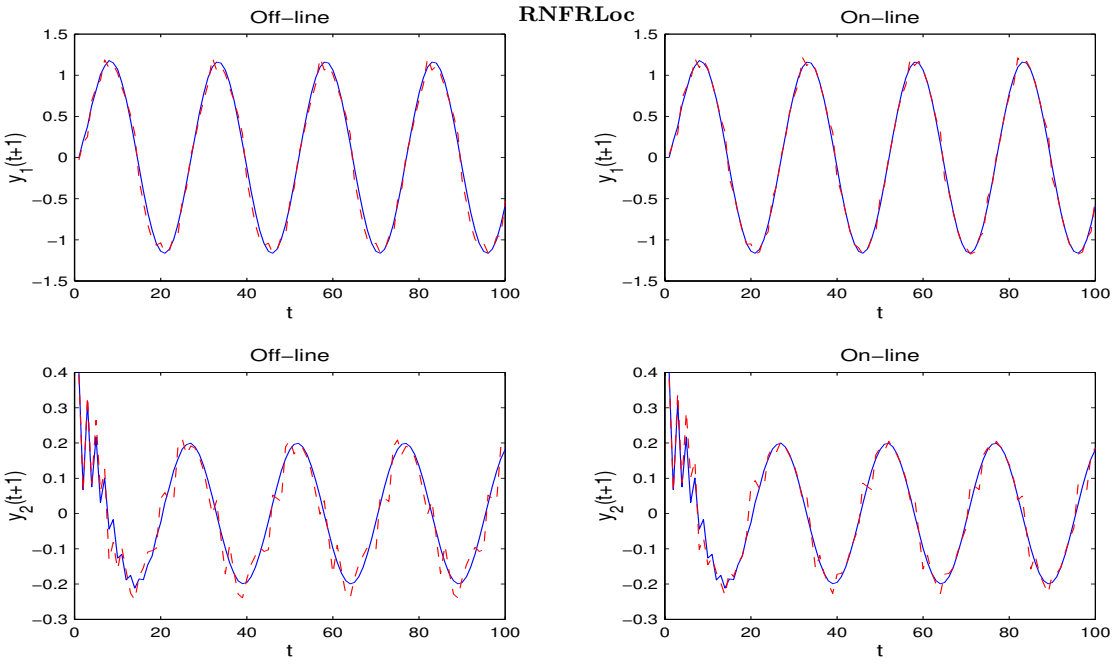


Figura 6.21: Exemplo 6.6, primeira coluna com $r_1(t)$ e segunda coluna com $r_2(t)$: (—) saída desejada, $(\cdots)$ saída do sistema.

Dos resultados de simulação apresentados, pode-se dizer que as redes neurofuzzy propostas mostraram-se eficientes no processo de identificação de sistemas. No caso do controle, estas tiveram maiores dificuldades. Contudo, as redes neurofuzzy conseguiram melhorar o seu desempenho após o ajuste *on-line* e tanto em identificação como em controle, atingiram bons resultados em termos de precisão, recursos computacionais e tempo de processamento.

A recorrência aplicada nas redes neurofuzzy apresentou-se como uma opção vantajosa pois a necessidade de informação para identificar os sistemas foi menor, isto em relação ao número de entradas para modelar os sistemas, além do período de treinamento utilizado.

Das simulações estudadas, observou-se que nem sempre é necessária a recorrência global, pois a maioria das vezes a rede neurofuzzy com recorrência local proposta (**RNFRLoc**) forneceu resultados tão bons ou melhores aos obtidos pela rede neurofuzzy com recorrência interna global (**RNFRGlob**). Isto devido à restrição na quantidade de neurônios lógicos ativos por iteração na camada intermediária. Portanto, nem sempre foi necessária uma rede neural mais complexa para obter os melhores resultados. Assim, é necessário o estabelecimento de um compromisso entre complexidade da solução e desempenho desejado.

Dos testes feitos, foi observada a grande diferença no tratamento dos problemas de identificação e controle, pois embora algumas estruturas de redes neurais tenham-se mostrado eficientes na resolução do problema de identificação de um sistema, estas não tiveram um bom desempenho quando aplicadas no controle do mesmo sistema, sendo necessário o uso de estruturas mais complexas com uma partição do espaço de entrada mais refinada e/ou períodos de treinamento mais longos. Contudo, o desempenho dos modelos neurofuzzy propostos foi tão bom ou melhor aos outros modelos testados.

Os resultados apresentados foram obtidos considerando um modelo em modo serie-paralelo (Narendra & Parthasarathy 1990). Contudo, os modelos neurais/neurofuzzy propostos também podem ser utilizados configurando o modelo em modo paralelo.

A Figura a seguir ilustra a saída da rede neurofuzzy com recorrência local (**RNFRLoc**), definida pelas Tabelas 6.1, 6.2 e trabalhando em modo paralelo na identificação do sistema não linear descrito no exemplo 6.1.

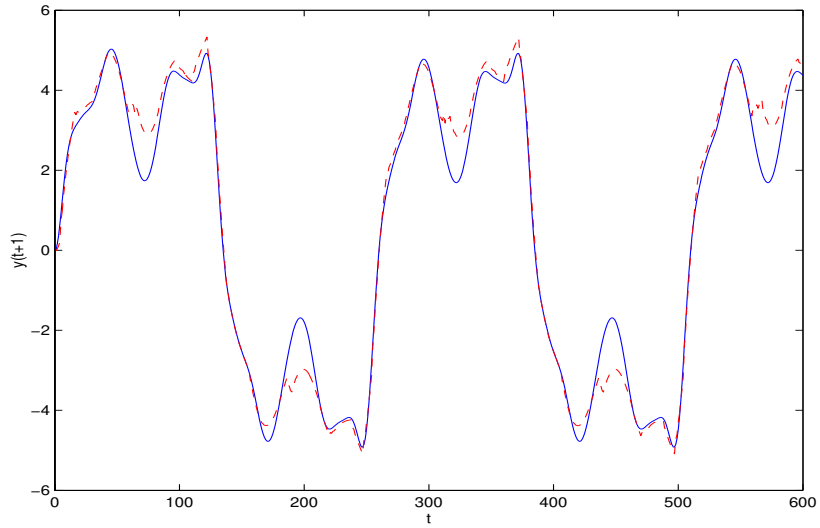


Figura 6.22: Desempenho da rede **RNFRLoc** em modo paralelo na identificação do sistema do exemplo 6.1: (—) saída desejada, $(\cdots)$ saída da rede neural.

Como pode ser observado, a rede neurofuzzy com recorrência interna local trabalhando em modo paralelo, consegue um bom desempenho, obtendo neste caso, um EQM de teste de 0.1352, para um sinal de entrada $u(t) = \text{sen}(2\pi t/250)$, com $t = 0, \ldots, 599$. Esta característica também foi observada para as outras redes neurofuzzy propostas (**RNFEst** e **RNFRGlob**).

A Figura 6.23 ilustra a saída da rede neural para o mesmo exemplo de identificação, sendo que desta vez, as saídas de todos os neurônios lógicos AND da camada intermediária da rede **RNFRLoc** foram consideradas para o processamento da saída da rede. Além disso, todos os pesos da camada de saída (v) foram atualizados. O resultado apresentado foi obtido através da configuração do sistema em modo serie-paralelo. O EQM obtido neste caso foi de 0.0780. Já a mesma rede neural numa configuração paralela não obteve resultados satisfatórios, como mostra a Figura 6.24.

Dos últimos testes apresentados, pode-se concluir que as redes neurofuzzy propostas conseguem um desempenho satisfatório nos problemas abordados, tanto em modo serie-paralelo como em modo paralelo. No entanto, quando todas as saídas dos neurônios lógicos são consideradas no cálculo da saída da rede, o desempenho da rede neural em modo paralelo é prejudicado.

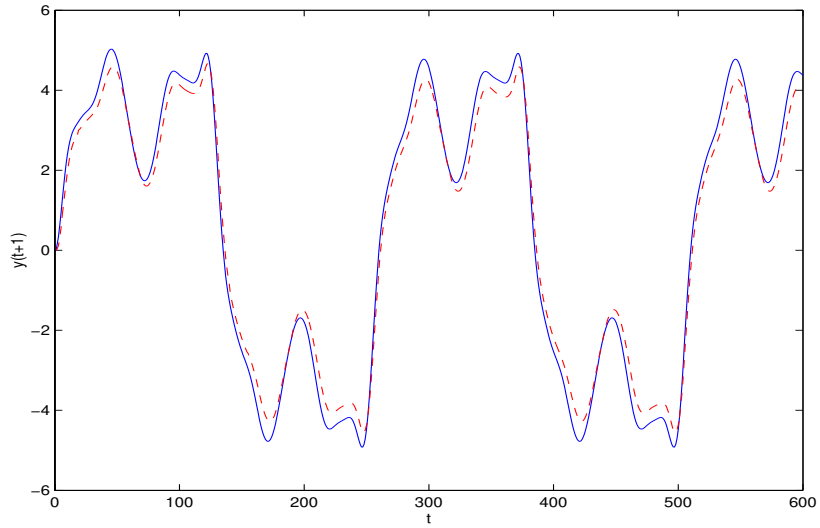


Figura 6.23: Desempenho da rede **RNFRLoc** em modo serie-paralelo na identificação do sistema do exemplo 6.1, considerando todos os neurônios AND: (—) saída desejada, ($\cdots$) saída da rede neural.

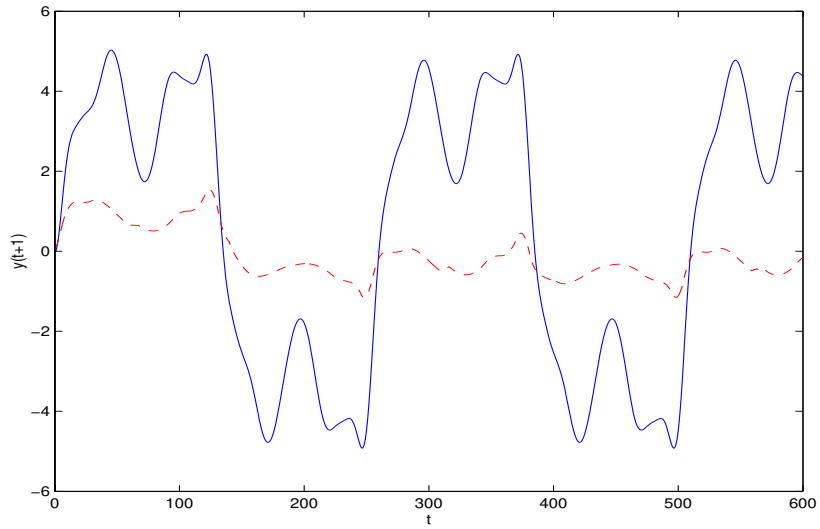


Figura 6.24: Desempenho da rede **RNFRLoc** em modo paralelo na identificação do sistema do exemplo 6.1, considerando todos os neurônios AND: (—) saída desejada, ($\cdots$) saída da rede neural.

A rede neurofuzzy com recorrência local (**RNFRLoc**) foi também aplicada a previsão de séries temporais. Em (Luna, Magalhães, Ballini, Gomide & Soares 2003), é apresentada a previsão de vazões afluentes mensais do posto de Furnas utilizando um modelo de rede **RNFRLoc** para cada mês. Os resultados obtidos foram comparados com os resultados obtidos

do modelo periódico autoregressivo médias móveis (PARMA), sendo este último o modelo atual utilizado pelo setor elétrico brasileiro. A rede neurofuzzy também mostrou-se eficiente nesta aplicação.

6.3 Resumo

Este capítulo apresentou os resultados de simulação obtidos utilizando as metodologias de identificação e controle descritas no capítulo anterior. As redes neurais foram testadas utilizando modelos série-paralelos para uma série de exemplos. A abordagem utilizada para o controle dos sistemas foi a técnica de controle por modelo de referência. As não linearidades dos sistemas foram identificadas para, a seguir, testar as estruturas em um sistema em malha aberta. Os pesos dos sistemas, inicialmente obtidos via o treinamento *off-line* foram ajustados através de um treinamento *on-line*. Os resultados de simulação e comparações entre redes neurais e redes neurofuzzy propostas na literatura demonstram a capacidade das redes neurofuzzy propostas neste trabalho de identificar e controlar os sistemas dinâmicos não lineares, geralmente com um número reduzido de parâmetros e iterações, acarretando em menor erro de controle e tempo de processamento.

Capítulo 7

Conclusão e Perspectivas Futuras

7.1 Conclusão

A combinação de diferentes abordagens da área da inteligência computacional é motivo de grande interesse nestes últimos anos, pois a possibilidade de conseguir modelos tão ou mais robustos e eficientes para a resolução de diversos problemas, mostra-se como uma opção atraente. É assim que a unificação das redes neurais com os sistemas fuzzy em um modelo com capacidade de aprendizagem, de representação e inferência, denominado de modelo neurofuzzy, foi motivo de estudo neste trabalho.

Foram apresentadas duas propostas de redes neurofuzzy recorrentes, as quais foram geradas a partir de uma rede neurofuzzy estática. Dois tipos de recorrência nos neurônios lógicos que compõem as redes foram considerados: recorrência interna local e recorrência interna global.

Os modelos neurofuzzy aqui propostos utilizam neurônios lógicos na camada intermediária e podem ser implementados utilizando diferentes normas triangulares, com uma clara facilidade para extração do conhecimento a partir de dados entrada-saída. Isto é, os modelos neurofuzzy apresentam equivalência com um sistema de inferência fuzzy, deixando de ser uma total "caixa preta", como no caso das redes neurais clássicas.

O desempenho dos modelos propostos foi avaliado considerando problemas de identificação e controle de sistemas dinâmicos discretos não lineares, explorando exemplos utilizados na literatura, de complexidade diversa.

Modelos neurais estáticos e recorrentes, assim como sistemas fuzzy e modelos neurofuzzy

estáticos e recorrentes da literatura foram ajustados para cada um dos exemplos, tanto de identificação como de controle, utilizando para isso, os métodos de treinamento associados a cada uma das redes neurais.

Os resultados de simulação demonstraram que os modelos neurofuzzy apresentados neste trabalho são capazes de resolver problemas que envolvam dinâmicas não lineares e relações temporais, diminuindo assim, a quantidade de informação *a priori* necessária para a resolução do problema, como foi no caso dos problemas de identificação de sistemas apresentados no capítulo anterior.

Foi observado que, nem sempre é necessária a recorrência interna global na rede neurofuzzy para a resolução de um problema de identificação e controle, sendo suficiente a recorrência interna local aplicado a um sistema neurofuzzy estático. Isto é, nem sempre um modelo mais complexo garante uma melhor solução ao problema.

No caso do problema de identificação e controle, é também de grande influência a natureza do sistema a ser identificado, levando à necessidade de mais ou menos recursos para a obtenção de um bom desempenho. Contudo, é válido para todos os modelos computacionais testados.

Cabe observar que, embora, os modelos neurofuzzy tenham apresentado um tempo de processamento muito pequeno (complexidade linear com relação ao tempo) e uma necessidade menor de recursos por iteração, elas possuem uma complexidade exponencial com relação ao tamanho da sua estrutura, devido à técnica de partição do espaço de entrada. Ou seja, uma partição do espaço entrada mais refinada, provocará um crescimento exponencial no tamanho da estrutura da rede.

O algoritmo de aprendizado proposto para os modelos neurofuzzy apresentados neste trabalho mostrou-se simples e rápido, não precisando de cálculos de derivadas para o ajuste dos pesos da camada intermediária e da recorrência.

Pode-se concluir que as redes neurofuzzy recorrentes mostram-se como uma alternativa atrativa em aplicações envolvendo dinâmicas não lineares.

7.2 Perspectivas Futuras

Como trabalho futuro pode-se considerar a formalização do processo de extração de regras da arquitetura dos modelos neurofuzzy propostos. Isto é, determinar regras do tipo $SE - ENTÃO$ que descrevam a dinâmica da rede como um todo. Trabalhos similares são encontrados na literatura, mas aplicados unicamente a redes neurais puras. Este processo pode ser estendido a modelos neurofuzzy, como no caso dos modelos propostos.

Considerando o problema de crescimento exponencial da estrutura dos modelos neurofuzzy propostos, com relação às partições do espaço de entrada, é possível a utilização de algum método construtivo para a incorporação de regras que fossem sendo necessárias na melhora do desempenho da rede. Isto é, aumentar o número de neurônios lógicos na camada intermediária enquanto a estrutura é ajustada para uma aplicação determinada, ou a modificação *on-line* da partição do espaço de entrada durante o ajuste da rede, de tal forma que o desempenho da rede melhore e o aumento da capacidade da estrutura seja gradual e suave.

Uma outra consideração necessária ao estudo dos modelos neurofuzzy apresentados neste trabalho é o análise da influência de ruido nos dados de treinamento no seu desempenho.

Finalmente, seria interessante definir algum critério de seleção das normas triangulares a serem utilizadas, para cada neurônio lógico necessário na arquitetura híbrida.

Referências Bibliográficas

Abraham, A. (n.d.). Beyond integrated neuro-fuzzy systems: Reviews, prospects, perspectives and directions. *citeseer.nj.nec.com/508142.html

Alexander, J. & Mozer, M. (1999). Template-based procedures for neural network interpretation, *Neural Networks* **12**: 479–498.

Atiya, A. & Parlos, A. (2000). New results on recurrent network training: unifying the algorithms and accelerating convergence, *IEEE Transactions on Neural Networks* **11**(3): 697–709.

Ballini, R. (2000). *Análise e Previsão de Vazões Utilizando Modelos de Série Temporais, Redes Neurais e Redes Neurais Nebulosas*, Tese de Doutorado, FEEC - Unicamp, Brasil.

Barto, A. & Jordan, M. (1987). Gradient Following without Backpropagation in Layered Networks, *in Proceedings of the IEEE First International Conference on Neural Networks* (2): 629–636.

Baxt, W. (1990). Use of an Artificial Neural Network for Data Analysis in Clinical Decision Making: The Diagnosis of Acute Coronary Occlusion, *Neural Computation* **2**: 480–489.

Baxt, W. (1991). Use of an Artificial Neural Network for the Diagnosis of Myocardial Infarction, *Annals of Internal Medicine* **115**: 843–848.

Benítez, J., Castro, J. & Requena, I. (1997). Are Artificial Neural Networks Black Boxes?, *IEEE Transactions on Neural Networks* **8**(5): 1156–1164.

Bersini, H. & Gorrini, V. (1992). A Simple Direct Adaptive Fuzzy Controller derived from its Neural Equivalent, *Technical Report TR/IRIDIA/92-20*, Institut de Reserches Interdisciplinaires et de Développements en Intelligence Artificielle.

Bersini, H. & Gorrini, V. (1993). FUNNY (FUzzY or Neural Net) Methods for Adaptive Process Control, *Technical Report TR/IRIDIA/93-12*, Institut de Reserches Interdisciplinaires et de Développements en Intelligence Artificielle.

Bersini, H. & Gorrini, V. (1994). Recurrent Fuzzy Systems, *Technical Report TR/IRIDIA/94-11*, Institut de Reserches Interdisciplinaires et de Développements en Intelligence Artificielle.

Brouwer, R. K. (2000). A fuzzy recurrent artificial neural network (frann) for pattern classification, *International Journal of uncertainty, Fuzziness and Knowledge-Based Systems* **8**(5): 525–538.

Buckley, J. J. & Hayashi, Y. (1994). Fuzzy neural networks: A survey, *Fuzzy Sets and Systems* **66**: 1–13.

Buckley, J. J. & Hayashi, Y. (1995). Neural nets for fuzzy systems, *Fuzzy Sets and Systems* **71**(3): 265–276.

Cai, Y.-D., Li, Y.-X. & Chou, K.-C. (2000). Using neural networks for prediction of domain structural classes, *Biochimica et Biophysica Acta (BBA)/Protein Structure and Molecular Enzymology* **1476**(1): 1–2.

Cai, Y.-D. & Zhou, G. (2000). Prediction of protein structural classes by neural network, *Biochimie* **82**(8): 783–785.

Caminhas, W. M. (1997). *Estratégias de Deteção e Diagnóstico de Falhas em Sistemas Dinâmicos*, Tese de Doutorado, Universidade Estadual de Campinas - UNICAMP, Brasil.

Caminhas, W., Tavares, H., Gomide, F. & Pedrycz, W. (1999). Fuzzy sets based neural networks: Structure, learning and applications, *Journal of Advanced Computational Intelligence* **3**(3): 151–157.

Campolucci, P., Uncini, A., Piazza, F. & Rao, B. D. (1999). On-Line Learning Algorithms for Locally Recurrent Neural Networks, *IEEE Transactions on Neural Networks* **10**(2): 253–270.

Castro, J. L., Mantas, C. J. & Benítez, J. M. (2002). Interpretation of Artificial Neural Networks by Means of Fuzzy Rules, *IEEE Transactions on Neural Networks* **13**(1): 101–116.

Cheng, Y., Karjala, T. & Himmelblau, D. (1997). Closed Loop Nonlinear Process Identification Using Internally Recurrent Nets, *Neural Networks* **10**(3): 573–586.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function, *Mathematics fo Control, Signals and Systems* **2**: 303–314.

de Moraes Lima, C. A. (2000). *Emprego de Teoria de Agentes no Desenvolvimento de Dispositivos Neurocomputacionais Híbridos e Aplicação ao Controle e Identificação de Sistemas Dinâmicos*, Tese de Mestrado, FEEC - Unicamp.

Elman, J. L. (1990). Finding Structure in Time, *Cognitive Science* **14**: 179–211.

Figueiredo, M. (1997). *Redes Neurais Nebulosas Aplicadas em Problemas de Modelagem e Controle Autônomo*, Tese de Doutorado, FEEC - Unicamp, Brasil.

Figueiredo, M. & Gomide, F. (1997). A Neural Fuzzy Approach for Fuzzy System Design, *International Conference on Neural Networks - ICNN'97* pp. 420–425.

Figueiredo, M. & Gomide, F. (1999). Design of Fuzzy Systems Using Neurofuzzy Networks, *IEEE Transactions on Neural Networks* **10**(4): 815–827.

Féraud, R. & Clérot, F. (2002). A methodology to explain neural network classification, *Neural Networks* **15**: 237–246.

Fu, L. (1994). Rule generation from neural networks, *IEEE Transactions on System, Man and Cybernetics* **24**: 1114–1124.

Fullér, R. (1995). Neural fuzzy systems, 1995. *citeseer.nj.nec.com/64350.html

Fuzzy Logic Toolbox, For Use with MATLAB® (2000). 2 edn, The Math Works, Inc., Natick, MA.

Giles, C., Omlin, C. & Thornber, K. (1999). Equivalence in Knowledge Representation: Automata, Recurrent Neural Networks, and Dynamical Fuzzy Systems, *Proceedings of the IEEE* **87**(9): 1623–1640.

Grossberg, S. (1976). Adaptive Pattern Classification and Universal Recoding. I: Parallel Development and Coding of Neural Feature Detectors, *Biological Cybernetics* **23**: 121–134.

Gupta, M. & QI, J. (1992). *On fuzzy neuron models*, John Wiley and Sons, Inc., chapter 25, pp. 479–491.

Haykin, S. (1994). *Neural Networks, A Comprehensive Foundation*, Macmillan Publishing Company.

Hebb, D. (1949). *The Organization of Behavior*, N.Y. Wiley, New York.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the U.S.A.* **79**: 2554–2558.

Hunt, K., Sbarbaro, D., Zbikowski, R. & Gawthrop, P. (1992). Neural Networks for Control Systems - A Survey, *Automatica* **28**(6): 1083–1112.

Hush, D. & Horne, B. (1993). Progress in Supervised Neural Networks, *IEEE Signal Processing Magazine* pp. 8–39.

Hussain, M. A. (1999). Review of the applications of neural networks in chemical process control - simulation and online implementation, *Artificial Intelligence in Engineering* **13**: 55–68.

Iyoda, E. (2000). *Inteligência computacional no projeto automático de redes neurais híbridas e redes neurofuzzy heterogêneas*, Tese de Mestrado, FEEC - Unicamp, Brasil.

Jang, J.-S. R. (1993). ANFIS: Adaptive-Network-Based Fuzzy Inference System, *IEEE Transactions on Systems, Man and Cybernetics* **23**(3): 665–685.

Jang, J.-S. R., Sun, C.-T. & Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing*, Prentice Hall.

Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine, *In Proceedings of the Eighth Annual Conference of the Cognitive Science Society* .

Jouffe, L. (1998). Fuzzy Inference System Learning by Reinforcement Methods, *IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews* **28**(3): 338–355.

Kaelbing, L. P., Littman, M. L. & Moore, A. W. (1996). Reinforcement Learning: A Survey, *Journal of Artificial Intelligence Research* **4**: 237–285.

Kohonen, T. (1982). Self-organized Formation of Topologically Correct Feature Maps, *Biologica Cybernetics* **43**: 59–69.

Kohonen, T., Kaski, S., Lagus, K., Salojrvi, J., Honkela, J., Paatero, V. & Saarela, A. (2000). Self organization of a massive document collection.
*citeseer.nj.nec.com/378852.html

Ku, C. & Lee, K. (1995). Diagonal recurrent neural networks for dynamic systems control, *IEEE Transactions on Neural Networks* **6**(1): 144–156.

Lau, C. (1991). *Neural Networks, Theoretical Foundations and Analysis*, IEEE Press.

Lee, C. & Teng, C. (2000). Identification and control of dynamic systems using recurrent fuzzy neural networks, *IEEE Transactions on Fuzzy Systems* **4**(8): 349–366.

Lee, M., Lee, S.-Y. & Park, C. (1995). Neurofuzzy controller design using neurofuzzy identifier, *International Journal of Approximate Reasoning* **13**: 269–285.

Lin, C.-T. & Lee, C. G. (1996). *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*, Prentice Hall Inc.

Lin, F.-J. & Wai, R.-J. (2001). Hybrid Control using Recurrent Fuzzy Neural Network for Linear-Induction Motor Servo Drive, *IEEE Transactions on Fuzzy Systems* **9**(1): 102–115.

Lin, Y. & Cunningham III, G. (1995). A New Approach to Fuzzy-Neural System Modeling, *IEEE Transactions on Fuzzy Systems* **3**(2): 190–198.

Luna, I., Ballini, R. & Gomide, F. (2002). Rede Neurofuzzy Recorrente para Identificação e Controle de Sistemas Dinâmicos Discretos, *Anais do XIV Congresso Brasileiro de Automação, Natal - Brasil* pp. 353–358.

Luna, I., Ballini, R. & Gomide, F. (2003a). Incorporação de Recocrrência em Estruturas Neurofuzzy, *VI Congresso Brasileiro de Redes Neurais, São Paulo - Brasil* pp. 97 – 102.

Luna, I., Ballini, R. & Gomide, F. (2003b). Sistema Híbrido Recurrente para Modelaje y Control de Sistemas Dinámicos no Lineales, *I Congreso Internacional de Cientificos Peruanos, Lima - Perú* .

Luna, I., Magalhães, M. H., Ballini, R., Gomide, F. & Soares, S. (2003). Aplicação de Sistemas Neurofuzzy Recorrentes a Previsão de Vazões, *VI Simpósio Brasileiro de Automação Inteligente - Bauru* .

Mamdani, E. (1974). Applications of fuzzy algorithms for simple dynamic plant, *Proceedings of the IEEE* **121**(12): 1585–1588.

McCulloch, W. & Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics* **5**: 115–133.

Miller, I. T., Sutton, R. & Werbos, P. (1990). *Neural Networks for Control*, MIT Press, Cambridge, Massachusetts, London, England.

Minsky, M. (1969). *Perceptrons*, MIT Press, Cambridge MA.

Mitra, S. & Hayashi, Y. (2000). Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework, *IEEE Transactions on Neural Networks* **11**(3): 748–768. May.

Narendra, K. & Parthasarathy, K. (1990). Identification and Control of Dynamical Systems using Neural Networks, *IEEE Transactions on Neural Networks* **1**(1): 4–27.

Nerrand, O., Roussel-Ragot, P., Personnaz, L., Dreyfuz, G. & Marcos, S. (1993). Neural Networks and Nonlinear Adaptive Filtering: Unifying Concepts and New Algorithms, *Neural Computation* **5**: 165–199.

Nerrand, O., Roussel-Ragot, P., Urbani, D., Personnaz, L. & Dreyfus, G. (1994). Training Recurrent Neural Networks: Why and How? An Illustration in Dynamical Process Modeling, *IEEE Transactions on Neural Networks* **5**(2): 178–184.

Nguyen, D. H. & Widrow, B. (1990). Neural Networks for Self-Learning control Systems, *IEEE Control Systems Magazine* pp. 18–23.

Olurotimi, O. (1994). Recurrent Neural Network Training with Feedforward Complexity, *IEEE Transactions on Neural Networks* **5**(2): 185–197.

Onat, A., Kita, H. & Nishikawa, Y. (1998). Recurrent Neural Networks for Reinforcement Learning: Architecture, Learning and Internal Representation, *Proceedings of the IEEE - World Congress on Computational Intelligence* pp. 2010–2015.

Pearlmutter, B. A. (1995). Gradient Calculations for Dynamic Recurrent Neural Networks: a Survey, *IEEE Transactions on Neural Networks* **6**(5): 1212–1228.

Peat, M. (1996). Neural Networks in the Capital Markets, *Journal of Economic Behavior and Organization* **31**(2): 296–298.

Pedrycz, W. & Gomide, F. (1998). *An Introduction to Fuzzy Sets: Analysis and Design*, MIT Press, Cambridge, MA.

Rafiq, M., Bugmann, G. & Eastbrook, D. (2001). Neural network design for engineering applications, *Computers and Structures* **79**(17): 1541–1552.

Ronco, E. & Gawthrop, P. (1997). Neural Networks for Modelling and Control, *Technical Report csc97008*, Centre for System and Control, Department of Mechanical Engineering, University of Glasgow.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review* **65**: 386–408.

Rumelhart, D. & McClelland, J. (1986). *Parallel Distributed Processing*, Vol. 1, MIT Press, Cambridge.

Santos, E. & Von Zuben, F. (1999). Efficient second-order learning algorithms for discrete-time recurrent neural networks, *Recurrent Neural Networks: Design and Applications* pp. 47–75. L. R. Medsker and L. C. Jain, Eds., CRC Press.

Savkovic-Stevanovic, J. (1996). Neural net controller by inverse modeling for a distillation plant, *Computers Chemical Engineering* **20**: S925–S930.

Scarselli, F. & Tsoi, A. C. (1998). Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results, *Neural Networks* **11**(1): 15–37.

Setiono, R. (2000). Extracting M-of-N rules from trained neural networks, *IEEE Transactions on Neural Networks* **11**(2): 512–519.

Suykens, J., Vanderwalle, J. & Moore, D. B. (1995). *Artificial Neural Networks for Modelling and Control of Non-Linear Systems*, Kluwer Academic Publishers, Boston.

Takagi, T. & Sugeno, M. (1985). Fuzzy Identification of Systems and Its Applications to Modeling and Control, *IEEE Transactions on Systems, Man and Cybernetics* .

Tatibana, C. & Kaetsu, D. (2002). Uma Introdução às Redes Neurais, http://www.din.uem.br/ia/neurais.

Timmerman, A. (1997). Neural Networks in Finance and Investing. Using Artificial Intelligence to improve Realworld Performance, *International Journal of Forecasting* **13**(1): 144–146.

Von der Malsburg, C. (1973). Self-Organizing of Orientation Sensitive Cells in the Striate Cortex, *Kybernetik* **14**: 85–100.

Von Zuben, F. (1993). *Redes Neurais Aplicadas ao Controle de Máquina de Indução*, Tese de Mestrado, FEEC - Unicamp, Brasil.

Von Zuben, F. (1996). *Modelos Paramétricos e Não-Paramétricos de Redes Neurais Artificiais e Aplicações*, Tese de Doutorado, FEEC - Unicamp, Brasil.

Wang, L. (1994). *Adaptive Fuzzy Systems and Control*, Prentice Hall.

Warner, B. & Misra, M. (1996). Understanding Neural Networks as Statistical Tools, *The American Statistician* **50**(4): 284–293.

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*, Tese de Doutorado, Harvard University, Cambridge, MA.

Widrow, B. & Hoff, M. (1960). Adaptive Switching Circuits, *WESCON Conv. Rec.* pp. 96–140.

Yager, R. & Filev, D. (1994). *Essentials of Fuzzy Modeling and Control*, John Wiley and Sons. Inc.

Yamakawa, T., Uchino, E., Miki, T. & Kusanagi, H. (1992). A Neo Fuzzy Neuron and its Applications to System Identification and Predictions to System Behavior, *Proceedings of the 2nd IZUKA* pp. 477–483.

Zadeh, L. (1965). Fuzzy Sets, *Information and Control* **8**: 338–353.

Zhang, J. & Morris, A. J. (1999). Recurrent Neuro-Fuzzy Networks for Nonlinear Process Modeling, *IEEE Transactions on Neural Networks* **10**(2): 313–326.