UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

LUÍS GUSTAVO MOTA SOUZA

MODELOS LINEARES LOCAIS PARA IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS USANDO REDES NEURAIS COMPETITIVAS

FORTALEZA
2012

LUÍS GUSTAVO MOTA SOUZA

MODELOS LINEARES LOCAIS PARA IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS USANDO REDES NEURAIS COMPETITIVAS

Tese submetida à Coordenação do Programa de Pós-Graduação em Engenharia de Teleinformática, da Universidade Federal do Ceará, como parte dos requisitos exigidos para obtenção do grau de Doutor em Engenharia de Teleinformática.

Orientador: Prof. Dr. Guilherme de Alencar Barreto

FORTALEZA
2012

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Pós-Graduação em Engenharia - BPGE

S716m Souza, Luís Gustavo Mota.
Modelos lineares locais para identificação de sistemas dinâmicos usando redes neurais competitivas. / Luís Gustavo Mota Souza – 2012.
138 f. : il. color., enc. ; 30 cm.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Tecnologia, Departamento de Engenharia de Teleinformática, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2012
Área de Concentração: Sinais e Sistemas.
Orientação: Prof. Dr. Guilherme de Alencar Barreto.

1. Teleinformática. 2. Mapas auto- organizáveis. 3. Interpolação. 4. Inteligência artificial. I. Título.

CDD 621.38

LUÍS GUSTAVO MOTA SOUZA

MODELOS LINEARES LOCAIS PARA IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS USANDO REDES NEURAIS COMPETITIVAS

Tese submetida à Coordenação do Curso de Pós-Graduação em Engenharia de Teleinformática, da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Doutor em Engenharia de Teleinformática. Área de concentração Sinais e Sistemas.

Aprovada em 27/02/2012.

BANCA EXAMINADORA

Prof. Dr. Guilherme de Alencar Barreto (Orientador)
Universidade Federal do Ceará - UFC

Prof. Dr. George André Pereira Thé
Universidade Federal do Ceará - UFC

Prof. Dr. José Carlos Teles Campos
Universidade Federal do Ceará - UFC

Prof. Dr. Fernando Antonio Campos Gomide
Universidade de Campinas - Unicamp

Profa. Dra. Marley Maria Bernardes Rebuzzi Vellasco
Pontifícia Universidade Católica do Rio de Janeiro - PUC/RJ

Dedico este trabalho principalmente a minha amada esposa Carla, pelo seu amor e paciência ao longo destes árduos anos de luta e superação proporcionados pelo doutorado, a minha filha Giovana, que foi o maior presente que recebi de Deus até o momento, a minha amada mãe Terezinha, que me deu tantos incentivos, e por último ao meu pai Expedito (in memorium), pelo seu amor.

Agradecimentos

A Deus, o Senhor da minha vida.

Ao meu orientador, Prof. Guilherme de Alencar Barreto, a este sou bastante grato pela orientação e pelos incentivos dados ao meu trabalho.

Aos professores e funcionários do Departamento de Engenharia de Teleinformática que de forma direta ou indireta participaram do desenvolvimento deste trabalho.

A todos os colegas que compõem o grupo de pesquisa GRAMA, pelas críticas e sugestões ora apresentadas.

À minha família como todo, nas pessoas dos meus irmãos, cunhadas e sobrinhas, pelo apoio durante esta jornada.

Ao meu estimado sogro José Carlos e minha querida sogra Liduína, pelo carinho e atenção desprendidos durante estes anos.

À CAPES por ter custeado meus estudos de doutorado durante estes anos.

Resumo

Nesta tese aborda-se o problema de identificação de sistemas dinâmicos sobre a ótica dos modelos locais, em que o espaço de entrada é particionado em regiões de operação menores sobre as quais são construídos modelos de menor complexidade (em geral, lineares). Este tipo de modelo é uma alternativa aos chamados modelos globais em que a dinâmica do sistema é identificada usando-se uma única estrutura (em geral, não-linear) que cobre todo o espaço de entrada. Assim, o tema alvo desta tese é o projeto de modelos lineares locais cujo espaço de entrada é particionado por meio do uso de algoritmos de quantização vetorial, principalmente aqueles baseados em redes neurais competitivas. Para este fim, são propostos três novos modelos lineares locais baseados na rede SOM (*self-organizing map*), que são avaliados na tarefa de identificação do modelo inverso de quatro sistemas dinâmicos comumente usados na literatura em *benchmarks* de desempenhos. Os modelos propostos são também comparados com modelos globais baseados nas redes MLP (*multilayer perceptron*) e ELM (*extreme learning machines*), bem como com outros modelos lineares locais, tais como o modelo fuzzy Takagi-Sugeno e o modelo neural LLM (*local linear mapping*). Um amplo estudo é realizado visando comparar os desempenhos de todos os modelos supracitados segundo três critérios de avaliação, a saber: (*i*) erro médio quadrático normalizado, (*ii*) análise dos resíduos, e (*iii*) teste estatístico de Kolmogorov-Smirnov. De particular interesse para esta tese, é a avaliação da robustez dos modelos locais propostos com relação ao algoritmo de quantização vetorial usado no treinamento do modelo. Os resultados obtidos indicam que os desempenhos dos modelos locais propostos são superiores aos dos modelos globais baseados na rede MLP e equivalentes aos modelos globais baseados na rede ELM.

Palavras–Chave: Identificação de Sistemas. Sistemas dinâmicos. Redes neurais competitivas. Modelos Lineares Locais. Mapa de Kohonen. Quantização Vetorial.

Abstract

In this thesis the problem of nonlinear system identification is approached from the viewpoint of local models. The input space is partitioned into smaller operational regions with lower complexity models (usually linear) built for each one. This type of model is an alternative to global models, for which the system dynamics is identified using a single structure (usually nonlinear ones) that covers the whole input space. The aim of this thesis is to design of local linear models whose input space is partitioned by means of vector quantization algorithms, special those based on competitive learning neural networks. For this purpose, three novel local linear modeling methods based on the SOM (*self-organizing map*) are introduced and evaluated on the identification of the inverse model of four dynamical systems commonly used in the literature for performance benchmarking. The proposed models are also compared with global models based on the MLP (*multilayer perceptron*) and ELM (*extreme learning machines*), as well as with alternative local linear models, such as the Takagi-Sugeno fuzzy model and the LLM (*local linear mapping*) neural model. A comprehensive study is carried out to compare the performances of all the aforementioned models according to three evaluation criteria, namely: (*i*) normalized mean squared error, (*ii*) residual analysis, and (*iii*) Kolmogorov-Smirnov test. Of particular interest to this thesis is the evaluation of the robustness of the proposed local models with respect to the vector quantization algorithm used to train the model. The obtained results indicates that the performance of the proposed local models are superior to those achieved by the MLP-based global models and equivalent to those achieved by ELM-based global models.

Keywords: System Identification. Dynamical Systems. Competitive Neural Networks. Local Linear Models. Self-Organizing Map. Vector Quantization.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ANN	*Adaptive Neural Network*
D-MKSOM	KSOM Múltiplo Baseado em Clusters de Dados
EG	Erro de Generalização
KSOM	Modelo Local Baseado em K Neurônios Vencedores
KS	Teste estatístico de Kolmogorov-Smirnov
LMS	*Least-Mean-Squares*
LLM	*Linear Local Mapping*
LESSOM	*Local Least-Squares SOM*
MSE	*Mean-Squared Error*
MLP	*Multi-layer Perceptron*
ELM	*Extreme Learning Machine*
MIMO	*Multiple-Input Multiple-Output*
NMSE	*Normalized Mean-Squared Error*
P-MKSOM	KSOM Múltiplo Baseado em Protótipos
RBF	*Radial Basis Function*
RNA	Redes Neurais Artificiais
SOM	*Self-Organizing Map*
SISO	*Single-Input Single-Output*
TS	Modelo Fuzzy de Takagi-Sugeno
VQTAM	*Vector-Quantized Temporal Associative Memory*
WTA	*Winner Take All*

Lista de Símbolos

t	Tempo discreto
$\{u(t)\}$	Sequência de símbolos de dados de entrada da planta industrial
$\{y(t)\}$	Sequência de símbolos de dados de saída da planta industrial
q	Número de memória da sequência de entrada, $u(t)$
p	Número de memória da sequência de saída, $y(t)$
$f[\cdot]$	Mapeamento não-linear desconhecido
$f^{-1}[\cdot]$	Mapeamento inverso não-linear desconhecido
$\mathbf{v}(t)$	Vetor regressor no instante t
$\mathbf{x}(t)$	Vetor de entrada no instante t
N	Número total de amostras do conjunto de dados utilizado pela rede neural
N_1	Número de amostras do conjunto de dados utilizado na etapa de treinamento da rede neural
N_2	Número de amostras do conjunto de dados utilizado na etapa de teste da rede neural
h_1	Número de neurônios existentes na camada escondida das redes MLP e ELM
h_2	Número de neurônios existentes na segunda camada escondida da rede MLP
w_{ij}	Peso do i-ésimo neurônio oculto ligado a j-ésima componente de entrada
θ_i	Limiar associado ao neurônio i
u_i	Ativação do i-ésimo neurônio da camada oculta
m_{li}	Peso do l-ésimo neurônio de saída ligado ao i-ésimo neurônio oculto
θ_l	Limiar associado ao neurônio de saída l
$\mathbf{W}$	Matriz de pesos sinápticos da camada escondida
$\mathbf{y}(t)$	Vetor de saída dos neurônios da camada escondida do modelo ELM no instante t
$\mathbf{Y}$	Matriz de saída dos neurônios da camada escondida
$\bar{\mathbf{y}}(t)$	Vetor de saída dos neurônios da camada escondida do modelo ELM no instante t (com bias)
$\bar{\mathbf{Y}}$	Matriz de saída dos neurônios da camada escondida (com bias)
$\mathbf{m}$	Vetor de pesos sinápticos do neurônio na camada de saída
$J(\cdot)$	Função custo do erro quadrático instantâneo

Z	Conjunto de todos os parâmetros (pesos e limiares) da rede
V_i	Célula de Voronoi do i-ésimo neurônio
$\mathbf{w}_i(t)$	Vetor de pesos associado ao neurônio i no instante t
N_i	Número de vetores de dados pertencentes à célula de Voronoi do i-ésimo protótipo
$\alpha(t)$	Taxa de aprendizagem usada pelos algoritmos
N_{i^*}	Número de vetores de dados associados ao protótipo $\mathbf{w}_{i^*}$
$i^*(t)$	Índice do neurônio vencedor na rede no instante t
C_i	Número de vezes que o neurônio i foi escolhido vencedor
$f_i(t)$	Fator de ponderamento da distância euclidiana para o neurônio i no algoritmo FSCL no instante t
α_0, α_T	Valores inicial e final de $\alpha(t)$, respectivamente
$\mathcal{X}$	Espaço contínuo dos dados de entrada
$p(\mathbf{x})$	Densidade de probabilidade do vetor de entrada $\mathbf{x}$
$\mathcal{A}$	Espaço de saída discreto
Φ	Transformação não-linear
g	Número de neurônios existentes na rede SOM
$\mathbf{r}_i$	Coordenadas do neurônio i em um arranjo uni- ou bi-dimensional
$\mathbf{r}_{i^*}$	Coordenadas do neurônio vencedor i^* em um arranjo uni- ou bi-dimensional
$h(i^*, i; t)$	Função vizinhança da rede SOM no instante t
K_{i^*}	Conjunto vizinhança topológico que contém os neurônios vizinhos de i^*
$\sigma(t)$	Abertura da vizinhança topológica
σ_0, σ_f	Valores inicial e final de $\sigma(t)$, respectivamente
$\mu_i(\mathbf{x})$	Grau de pertinência de um vetor $\mathbf{x}$ associado ao neurônio i
G	Número de regras fuzzy
$\mathbf{w}_i^f$	Vetor protótipo de um agrupamento fuzzy associado ao ponto focal i
z	Grau de nebulosidade da função
$\mathbf{a}_i$	Vetor de coeficientes do filtro transversal linear associado ao neurônio i
α'	Taxa de aprendizagem do algoritmo LMS na rede LLM
$\mathbf{x}^{in}(t)$	Vetor de entrada do mapeamento dinâmico usado pela rede VQTAM
$x^{out}(t)$	Escalar de saída do mapeamento dinâmico usado pela rede VQTAM
$\mathbf{w}_i^{in}(t)$	Vetor de pesos do neurônio i associado ao espaço de entrada do mapeamento dinâmico
$w_i^{out}(t)$	Escalar do neurônio i associado ao espaço de saída do mapeamento dinâmico
K	Número de neurônios vencedores

$\mathbf{R}$	Matriz de regressão formada pelos vetores de pesos do espaço de entrada na rede KSOM
$\mathbf{p}$	Vetor de predição formada pelos escalares de pesos do espaço de saída na rede KSOM
$j_k^{(i)}$	k-ésimo índice do vetor protótipo mais próximo ao vetor $\mathbf{w}_i^{in}$
$\mathcal{J}_i$	Conjunto contendo os índices dos K vetores protótipos mais próximos a $\mathbf{w}_i^{in}$, incluindo o neurônio i
$\mathbf{R}_i$	Matriz composta das partes das entradas dos K vetores protótipos cujos os índices pertencem ao conjunto $\mathcal{J}_i$ da rede P-MKSOM
$\mathbf{b}_i^{out}$	Vetor composto das partes de saída dos K vetores protótipos cujos os índices pertencem ao conjunto $\mathcal{J}_i$ da rede P-MKSOM
$\mathbf{X}_i^{in}$	Conjunto de vetores de dados de entrada pertencentes à célula de Voronoi do neurônio i na rede D-MKSOM
$\mathbf{x}_i^{out}$	Conjunto de escalares de saída (x^{out}) associados com cada vetor $\mathbf{x}^{in}$ na rede D-MKSOM
$\mathbf{D}_i$	Matriz formada pelos vetores $\mathbf{x}^{in}$ pertencentes às células de Voronoi do conjunto de índices $\mathcal{J}_i$
$\mathbf{d}_i^{out}$	Vetor formado com as correspondentes partes de saída x^{out} associados aos vetores $\mathbf{x}^{in}$
$u(t)$	Saída desejada no instante t para modelagem inversa
$\hat{u}(t)$	Saída estimada no instante t
$e(t)$	Erro de aproximação no instante t
$\hat{f}[\cdot]$	Aproximação da função desconhecida $f[\cdot]$
σ_e^2	Variância do erro
σ_u^2	Variância do sinal de entrada
R^i	i-ésima regra fuzzy
$\mathcal{X}_j^i$	Conjuntos fuzzy de premissas
γ	Largura da premissa fuzzy
τ_i	Nível de disparo da i-ésima regra fuzzy
λ_i	Nível de disparo normalizado da i-ésima regra fuzzy
π_i	Vetor de parâmetros do i-ésimo modelo linear
$\mathbf{x}_e$	Vetor de entrada expandido
θ	Vetor composto por todos os parâmetros dos modelos lineares
ψ	Vetor total de ponderamento fuzzy da entrada $\mathbf{x}_e$ apresentada
$\mathbf{x}_t$	Vetor de entrada no instante t

$\mathbf{x}_{et}$	Vetor de entrada expandido no instante t

Sumário

1 Introdução

1.1 Identificação de Sistemas

Identificação de sistemas é a área da ciência que tem como meta principal a construção de modelos matemáticos que reproduzam adequadamente o funcionamento de um sistema dinâmico a partir de medições das suas entradas e saídas (AGUIRRE, 2007). Conhecer um modelo que descreva o comportamento de um sistema dinâmico é essencial em diversas áreas do conhecimento técnico-científico, principalmente em Engenharia e Estatística.

Grosso modo, métodos de identificação de sistemas podem ser classificados em duas categorias: modelos globais ou modelos locais. Modelos globais são projetados para caracterizar todo o domínio do problema usando apenas uma única estrutura matemática. Estas estruturas podem ser polinomiais, racionais, e até neurais (*feedforward* ou recorrentes, uni- ou multicamadas, etc.) (LAWRENCE et al., 1996). Modelos locais, por outro lado, particionam o domínio do problema em regiões menores de modo que um modelo mais estruturalmente simples seja adotado para cada região.

O paradigma de modelagem linear local está intimamente relacionada com a técnica estatística de regressão linear por partes. A ideia básica consiste em aproximar uma relação funcional global, supostamente existente entre múltiplas variáveis de entrada e uma de saída, por meio de um conjunto de hiperplanos (isto é, modelos lineares), cada qual com seu vetor de coeficientes estimado com dados pertencentes apenas à partição local. Didaticamente, é como aproximar uma superfície não-linear qualquer por um conjunto de hiperplanos.

Desde que foi demonstrado que redes neurais artificiais (RNAs) são aproximadores universais de função (HORNIK et al., 1989), tais estruturas vêm sendo utilizadas para fins de identificação e controle de sistemas. Costuma-se afirmar que, historicamente, identificação de sistemas dinâmicos não-lineares usando RNAs ganhou destaque como uma promissora área de pesquisa teórica e aplicada com a publicação do trabalho de Narendra

& Parthasarathy (1990). Estes autores mostraram que, sob determinadas condições, modelos neurais NARX[1] são capazes de tratar com eficiência problemas de identificação de sistemas dinâmicos não-lineares complexos. Desde então, a área de identificação de sistemas dinâmicos baseados em RNAs tem sido dominada por arquiteturas supervisionadas, tais como as redes Perceptron Multi-Camadas (*Multi-Layer Perceptron*, MLP) e Função de Base Radial (*Radial Basis Function*, RBF).

Menos usada para identificação de sistemas dinâmicos que seus pares supervisionados, mas com contribuições que remontam ao início da década de 1990 (veja, por exemplo, o trabalho de Walter et al.,1990), a rede SOM (*Self-Organizing Map*) vem sendo usada para este fim como uma alternativa eficiente às redes MLP e RBF (SOUZA; BARRETO, 2010; DÍAZ-BLANCO et al., 2007; CHO et al., 2007, 2006; BARRETO; SOUZA, 2006; BARRETO; ARAÚJO, 2004; PRINCIPE et al., 1998; KOHONEN et al., 1996). Basicamente, abordagens baseadas na rede SOM encaixam-se no paradigma de modelagem local linear, sendo o modelo local geralmente do tipo ARX[2].

A utilização de modelos lineares locais em identificação de sistemas dinâmicos é o tópico principal a ser abordado nesta tese. Em particular, está-se interessado na identificação do modelo inverso do sistema. Mais detalhes sobre este problema serão dados a seguir.

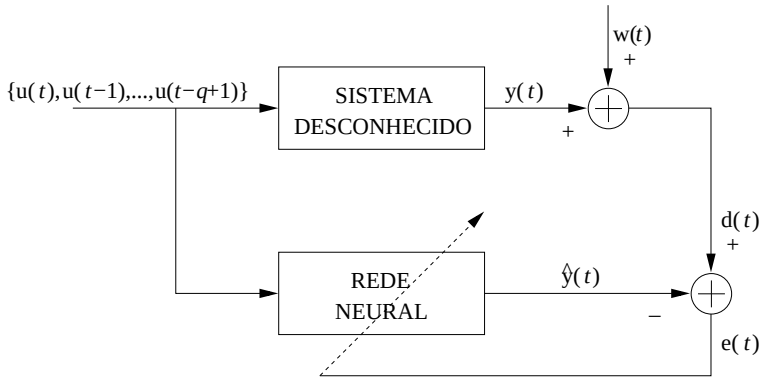
1.2 Motivação: Por que modelagem inversa?

RNAs podem ser usadas na identificação dos modelos direto e inverso de um sistema dinâmico. A Figura 1.1(a) ilustra o processo de aprendizado do modelo direto de um dado sistema, enquanto a Figura 1.1(b) ilustra a obtenção de um modelo inverso.

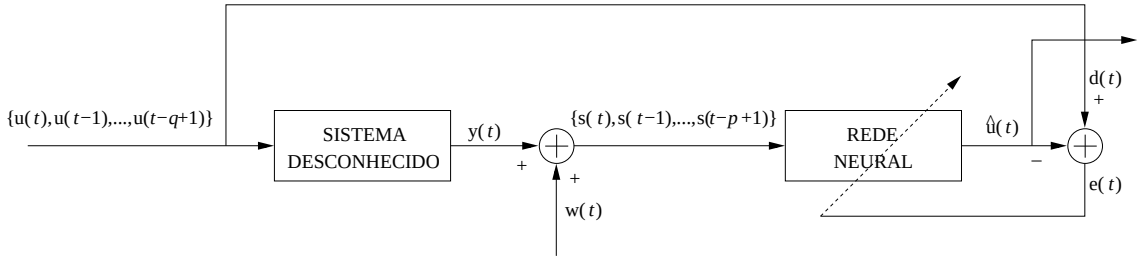
Na Figura 1.1(a), a amostra atual $u(t)$ e um conjunto de q amostras passadas de um sinal de tempo discreto qualquer $\{u(t), u(t-1), \ldots, u(t-q+1)\}$, representando o sinal de entrada do sistema, são aplicados simultaneamente ao sistema real e à rede neural no instante atual. Com o passar do tempo a saída da rede, $\hat{y}(t)$, vai se aproximando do valor da saída real do sistema, $d(t)$, passando a se comportar como o próprio sistema. A presença de ruído $w(t)$ na formação da saída desejada $d(t)$ serve para representar incertezas de medição.

Pode-se desprezar a influência do ruído, sem maiores consequências ao problema de

[1] Acrônimo de *Nonlinear AutoRegressive models with eXogenous inputs.*
[2] Acrônimo de *AutoRegressive models with eXogenous inputs.*



(a) Configuração de modelagem direta de sistemas.



(b) Configuração de modelagem inversa de sistemas.

Figura 1.1 Ilustração do processo de obtenção dos modelos direto e inverso de um dado sistema usando redes neurais artificiais.

modelagem direta. Neste caso, a relação matemática que se está tentando aproximar pode ser descrita como

$$y(t) = F(u(t), u(t-1), ..., u(t-q+1)), \tag{1.1}$$

em que $F(\cdot)$ é uma função desconhecida que (supostamente) representa a dinâmica do sistema real. A constante $q > 0$ é um número inteiro positivo, chamada de ordem da memória de entrada. A rede neural fornece então a seguinte saída do modelo direto:

$$\hat{y}(t) = \hat{F}(u(t), u(t-1), ..., u(t-q+1)). \tag{1.2}$$

Na Figura 1.1(b), a rede neural recebe agora como entrada as saídas passadas do sistema real (adicionada de ruído) e como saída a entrada do sistema real. De modo similar ao problema de modelagem direta, pode-se também desprezar a influência do ruído. Assim, a função que se está tentando aproximar pode ser descrita como

$$u(t) = F^{-1}(y(t-1), y(t-2), ..., y(t-p)), \tag{1.3}$$

em que $F^{-1}(\cdot)$ denota o mapeamento inverso de $F(\cdot)$. A constante $p > 1$ é um número

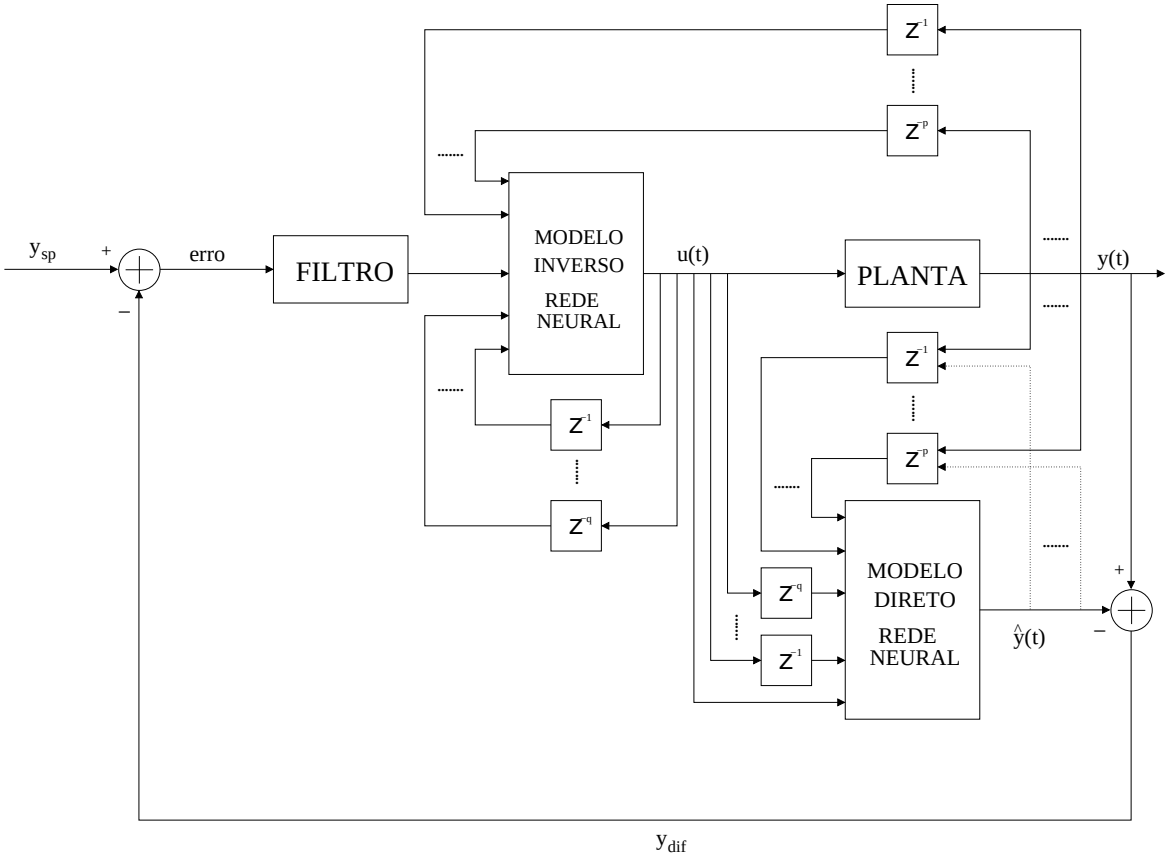


Figura 1.2 Estrutura da rede neural na configuração de controle por modelo interno.

inteiro positivo, chamada de ordem da memória de saída. A rede neural fornece então a seguinte saída do modelo inverso:

$$\hat{u}(t) = \hat{F}^{-1}(y(t-1), y(t-2), ..., y(t-p)). \tag{1.4}$$

A identificação do modelo inverso, ou simplesmente identificação inversa, é uma tarefa mais difícil de lidar que a direta, pois trata-se de um problema mal-posto (*ill-posed*), i.e. pode ter múltiplas soluções. Nesta tese, aborda-se tal problema sob a ótica dos modelos locais lineares. Um modelo inverso é útil em muitas estratégias de problemas de controle de sistemas, tal como a de controle por modelo interno (*Internal Model Control*, IMC) (HUSSAIN; KERSHENBAUM, 2000), ilustrado na Figura 1.2. Nesta figura, a rede neural que aprende o modelo inverso da planta também recebe como entrada o sinal de erro (filtrado, neste caso) entre a saída do sistema ($y(t)$) e a saída predita pelo modelo direto ($\hat{y}(t)$).

Uma extensão da abordagem para identificação inversa mostrada na Equação 1.4, comumente chamada de aprendizagem inversa generalizada (HUNT et al., 1992; HUSSAIN, 1996), é ilustrado na Figura 1.3. Neste caso, a rede neural é alimentada com o valores

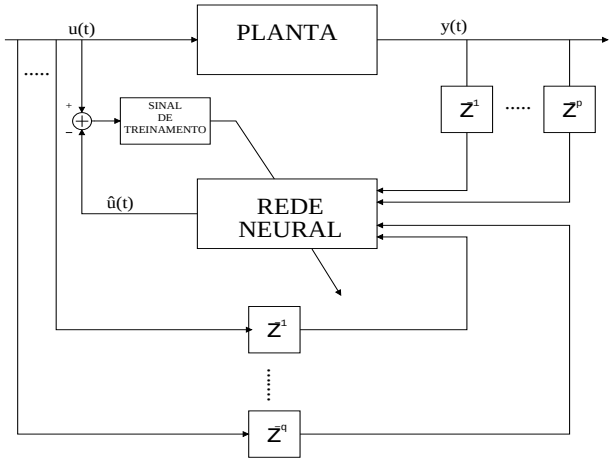


Figura 1.3 Aprendizagem inversa generalizada para treinamento de redes neurais em controle.

passados tanto do sinal de entrada quanto do sinal de saída da planta, de modo a estimar o valor atual do sinal de entrada da planta ($u(t)$). Matematicamente, tem-se

$$\hat{u}(t) = \hat{F}^{-1}(u(t-1), ..., u(t-q); y(t-1), ..., y(t-p)), \tag{1.5}$$

de modo que uma rede neural assim treinada passa então a representar o modelo inverso do sistema e pode ser utilizada, por exemplo, para fins de controle.

Nesta tese, apenas o problema de identificação do modelo inverso de um sistema dinâmico SISO[3] será tratado. Potenciais aplicações dos modelos a serem desenvolvidos nesta tese em sistemas de controle não serão abordados, ficando para trabalhos futuros.

1.3 Objetivos Geral e Específicos

O objetivo principal desta tese é propor e avaliar modelos lineares locais baseados em algoritmos de quantização vetorial, principalmente aqueles baseados em redes neurais competitivas, e aplicá-los na identificação de sistemas dinâmicos.

Quanto aos objetivos específicos desta tese, pode-se listar os seguintes itens:

- Estender a aplicação de modelos lineares locais que originalmente haviam sido aplicados em previsão de séries temporais ao problema correlato, porém mais geral, de identificação de sistemas dinâmicos SISO.
- Propor dois novos modelos lineares locais baseados na rede SOM para identificação inversa de sistemas dinâmicos.

[3]Acrônimo de *Single-input, Single-output*

- Avaliar o desempenho dos modelos locais propostos para uma gama de conjuntos de dados disponíveis na literatura especializada de identificação de sistemas.
- Comparar o desempenho dos modelos locais propostos com o desempenho de modelos globais baseados em redes neurais supervisionadas, tais como as redes MLP e ELM, e também com o desempenho obtido pelo modelo fuzzy Takagi-Sugeno.
- Avaliar o desempenho dos modelos propostos com base em dois critérios de avaliação, a saber: (*i*) erro médio quadrático normalizado e (*ii*) análise dos resíduos.
- Aplicar o teste estatístico de Kolmogorov-Smirnov para avaliar a sensibilidade dos modelos locais propostos em relação ao algoritmo de quantização vetorial usado no particionamento do domínio do problema.

1.4 Produção Científica

Ao longo do desenvolvimento desta tese, foram publicados ou submetidos a congressos ou periódicos os seguintes artigos:

1. BARRETO, G. A., **SOUZA, L. G. M.** "Multiple Local Linear Models for System Identification Using the Self-Organizing Map", submetido ao periódico *IEEE Transactions on Neural Networks and Learning Systems*, 2011.
2. BARRETO, G. A., **SOUZA, L. G. M.** "An Evaluation of Neural Vector Quantization Algorithms on the Design of Takagi-Sugeno Fuzzy Models", submetido ao periódico *Journal of Intelligent and Fuzzy Systems*, 2011.
3. BARRETO, G. A., **SOUZA, L. G. M.** "On Building Local Models for Inverse System Identification with Vector Quantization Algorithms", *Neurocomputing*, Vol. 73, Edição 10–12, Pags. 1993–2005, 2010.
4. **SOUZA, L. G. M.**, BARRETO, G. A. "Multiple Local Models for System Identification Using Vector Quantization". In: 18th European Symposium on Artificial Neural Networks (*ESANN*), Bruges, Belgium, Pags. 393–398, 2010.
5. **SOUZA, L. G. M.**, BARRETO, G. A. "Nonlinear System Identification Using Local ARX Models Based on the Self-Organizing Map". *Learning and Nonlinear Models*, Vol. 4, No. , Pags. 112–123, 2008.

6. **SOUZA, L. G. M.**, BARRETO, G. A. "Multiple Local ARX Modeling for System Identification Using the Self-Organizing Map", In: II European Symposium on Time Series Prediction (*ESTSP*), Porvoo, Finlândia, Pags. 215–224, 2008.

6. **SOUZA, L. G. M.**, BARRETO, G. A. "Local linear NARX models based on the self-organizing map", In: VIII Congresso Brasileiro de Redes Neurais (*CBRN*), Florianópolis, Vol. 1, Pags. 1–6, 2007.

1.5 Resumo dos Capítulos Restantes

O restante deste documento está organizado segundo a lista de capítulos apresentada abaixo.

Capítulo 2 - Neste capítulo serão apresentadas as duas arquiteturas de modelos globais baseados em redes neurais supervisionadas a serem utilizadas na tarefa de identificação inversa de sistemas dinâmicos. Particular atenção será dada à descrição da rede ELM (*Extreme Learning Machine*), por ser uma arquitetura de rede neural feedforward recente e que vem recebendo bastante atenção por parte da comunidade de Inteligência Computacional.

Capítulo 3 - Neste capítulo serão apresentados os algoritmos de quantização vetorial que serão usados para particionar o domínio do problema em regiões menores sobre as quais são construídos os modelos lineares locais. Em particular, será dada ênfase à descrição do funcionamento dos algoritmos baseados em redes neurais competitivas não-supervisionadas, em virtude de sua importância para os algoritmos a serem propostos neste trabalho.

Capítulo 4 - Neste capítulo serão apresentados três modelos lineares locais baseados na rede SOM já disponíveis na literatura há algum tempo, mas que serão empregados na tarefa de identificação inversa de sistemas dinâmicos a título de comparação de desempenho. Por último, será feita uma rápida apresentação do modelo local fuzzy Takagi-Sugeno para análise do seu desempenho na mesma tarefa.

Capítulo 5 - Neste capítulo são introduzidas as duas principais contribuições desta tese, a saber, dois modelos lineares locais baseados no algoritmo SOM. O primeiro modelo proposto constrói os modelos locais de cada partição do espaço de entrada usando um subconjunto dos vetores-protótipos do algoritmo de quan-

tização vetorial utilizado. Já o segundo modelo proposto constrói os modelos locais usando subconjuntos (*clusters*) dos dados de treinamento.

Capítulo 6 - Neste capítulo, os principais modelos locais e globais apresentados nos capítulos anteriores terão seus desempenhos avaliados na tarefa de identificação do modelo inverso de quatro sistemas dinâmicos bastante conhecidos na literatura especializada. As metodologias de simulação e comparação de desempenho são também apresentadas neste capítulo, bem como uma breve descrição de cada conjunto de dados utilizados.

Capítulo 7 - Este capítulo finaliza o presente documento ao fazer um resumo geral dos resultados obtidos e das principais contribuições da tese, além de apontar possíveis direções para trabalhos futuros na mesma linha de pesquisa.

Apêndice *A* - Neste apêndice são apresentados avaliações adicionais dos desempenhos dos modelos estudados nesta tese. Em particular, será avaliada a utilidade do diagrama de caixas (*boxplot*) na análise qualitativa dos resíduos produzidos por tais modelos.

2 *Modelos Globais em Identificação de Sistemas Dinâmicos*

Neste capítulo serão descritas as redes neurais MLP e ELM aplicadas à tarefa de identificação de sistemas dinâmicos. Estas redes são conhecidas na literatura como modelos globais, uma vez que uma só arquitetura é usada na tarefa de identificação. Será definido também o tipo de estrutura matemática a ser empregada na modelagem de sistemas dinâmicos. E por último, mas não menos importante, serão apresentadas as regras de ativação e aprendizagem desses modelos neurais globais, seguindo uma notação matricial que facilitará no entendimento geral dos algoritmos.

2.1 Introdução

Redes neurais artificiais (RNAs) são modelos bastante empregados na teoria de identificação de sistemas. Elas incluem mecanismos de aprendizagem não-linear junto com elementos processadores simples, chamados usualmente de neurônios artificiais. O modelo desses neurônios foi proposto inicialmente por McCulloch & Pitts (1943), seguindo a ideia que os neurônios têm capacidade de adaptação em função de estímulos (informação) oriundos do meio em que estão inseridos, lançando mão de processamento paralelo e distribuído para processar e codificar a informação nas conexões com outros neurônios.

As RNAs apresentam-se como uma ferramenta computacional eficiente no tratamento de problemas não-lineares em processamento de sinais (HWANG et al., 1997; ZAKNICH, 2003). Estes problemas requerem mapeamentos entrada-saída não-lineares, comumente encontrados em aproximação de funções e classificação de padrões. Este primeiro é tratado no decorrer desta tese, enquanto maiores esclarecimentos sobre a aplicação de redes neurais para classificação de padrões podem ser encontrados em Ripley (1996).

A principal dificuldade em identificação de sistemas é encontrar modelos computaci-

onais capazes de representar adequadamente a dinâmica de sistemas desconhecidos sem perda de generalidade e exatidão. Técnicas diferentes envolvendo modelos não-lineares foram propostas e aplicadas para modelar e controlar sistemas não-lineares (CHEN; XI, 1998; PRINCIPE et al., 1998), mas a obtenção de modelos adequados para este tipo de sistema é muito mais complexa e difícil quando comparada ao do tipo linear.

Em modelagem de sistemas não-lineares, é comum o uso de técnicas baseadas na partição do espaço de entrada e de saída utilizando ou um único modelo ou modelos múltiplos. A ideia principal da abordagem de modelo único é encontrar uma função global do sistema capaz de representar todo o seu comportamento dinâmico que define a relação existente entre a entrada e a saída. Este mapeamento ou função pode ser realizado usando abordagens como redes neurais (NARENDRA; PARTHASARATHY, 1990; NARENDRA, 1996; BARRETO; ARAÚJO, 2004) ou sistemas neuro-fuzzy (AZEEM et al., 2000; BABUSKA; VERBRUGGEN, 2003; RUBIO, 2009), para mencionar duas possibilidades. Modelos múltiplos ou locais serão comentados posteriormente no Capítulo 4.

A abordagem global a ser vista nesta tese está associada com o uso de modelos neurais, tais como as redes de perceptron de múltiplas camadas (MLP-*Multilayer Perceptron*)(RUMELHART et al., 1986) e máquina de aprendizado extremo (ELM-*Extreme Learning Machine*)(HUANG et al., 2006). Modelos globais representam o tema principal em aplicações de identificação de sistemas não-lineares e controle (NARENDRA; LEWIS, 2001; NORGAARD et al., 2000).

Para ilustrar o conceito de modelos globais aplicados à identificação de sistemas, será discutida a seguir, de uma forma conceitual, a tarefa de aproximação de uma função (regressão não-linear) usando as redes MLP e ELM. Diversas simulações são feitas com base na variação do número de neurônios da camada oculta dos modelos. Os resultados obtidos pelas redes neurais se aproximam ao comportamento global visto nestes gráficos definido pela função aproximada.

A função real a ser aproximada pelas redes neurais MLP e ELM é definida como:

$$y = sen[(2\pi/10)x] + 0.25x, \tag{2.1}$$

em que a variável x assume valores no intervalo [0,12]. Os dados utilizados para treinamento destas redes são gerados a partir da função da Equação (2.1), sendo contaminados por um ruído branco gaussiano.

Para treinamento da rede MLP foram utilizadas 100 épocas. Para cada simulação,

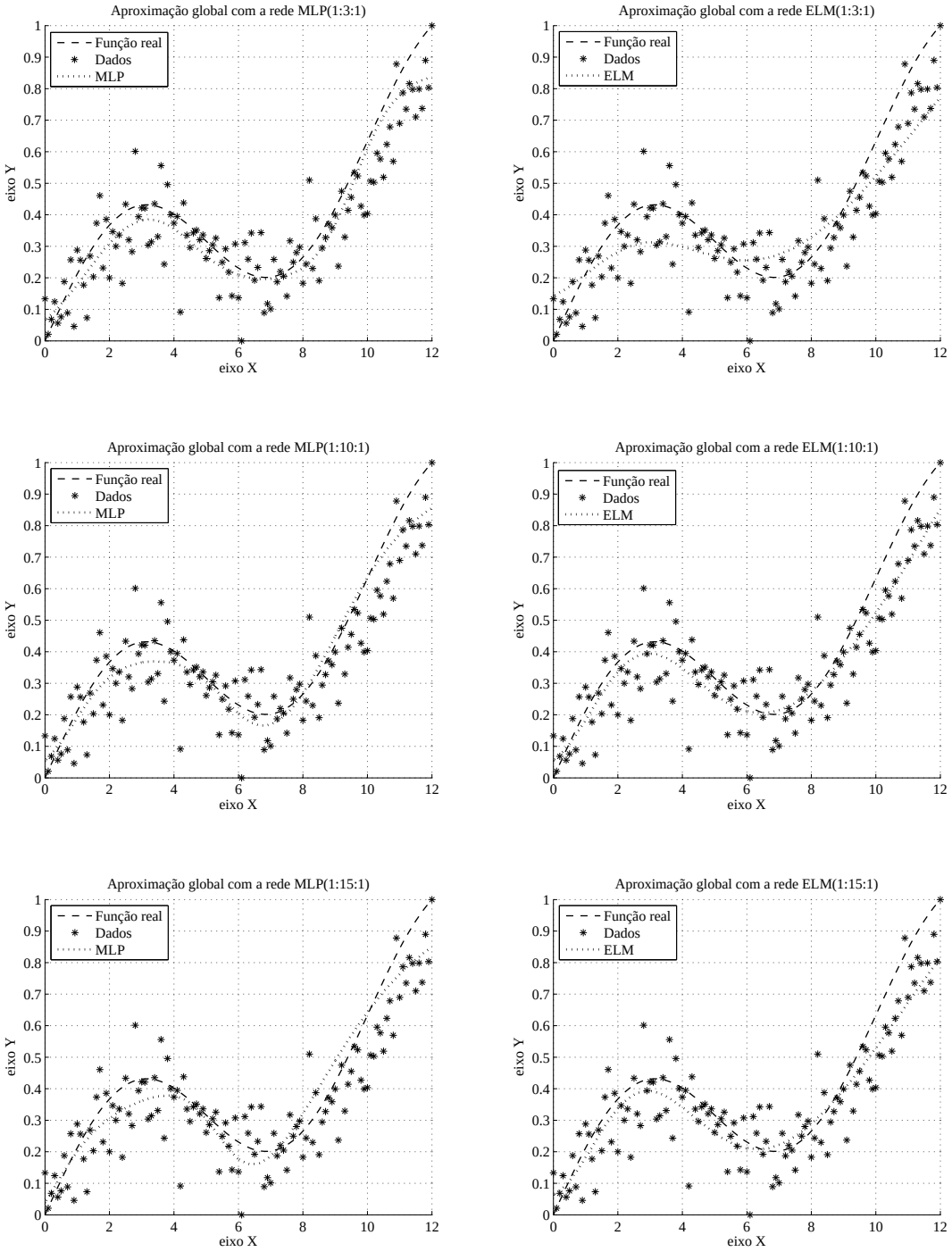


Figura 2.1 Aproximação global usando as redes MLP e ELM com 3, 10 e 15 neurônios ocultos cada.

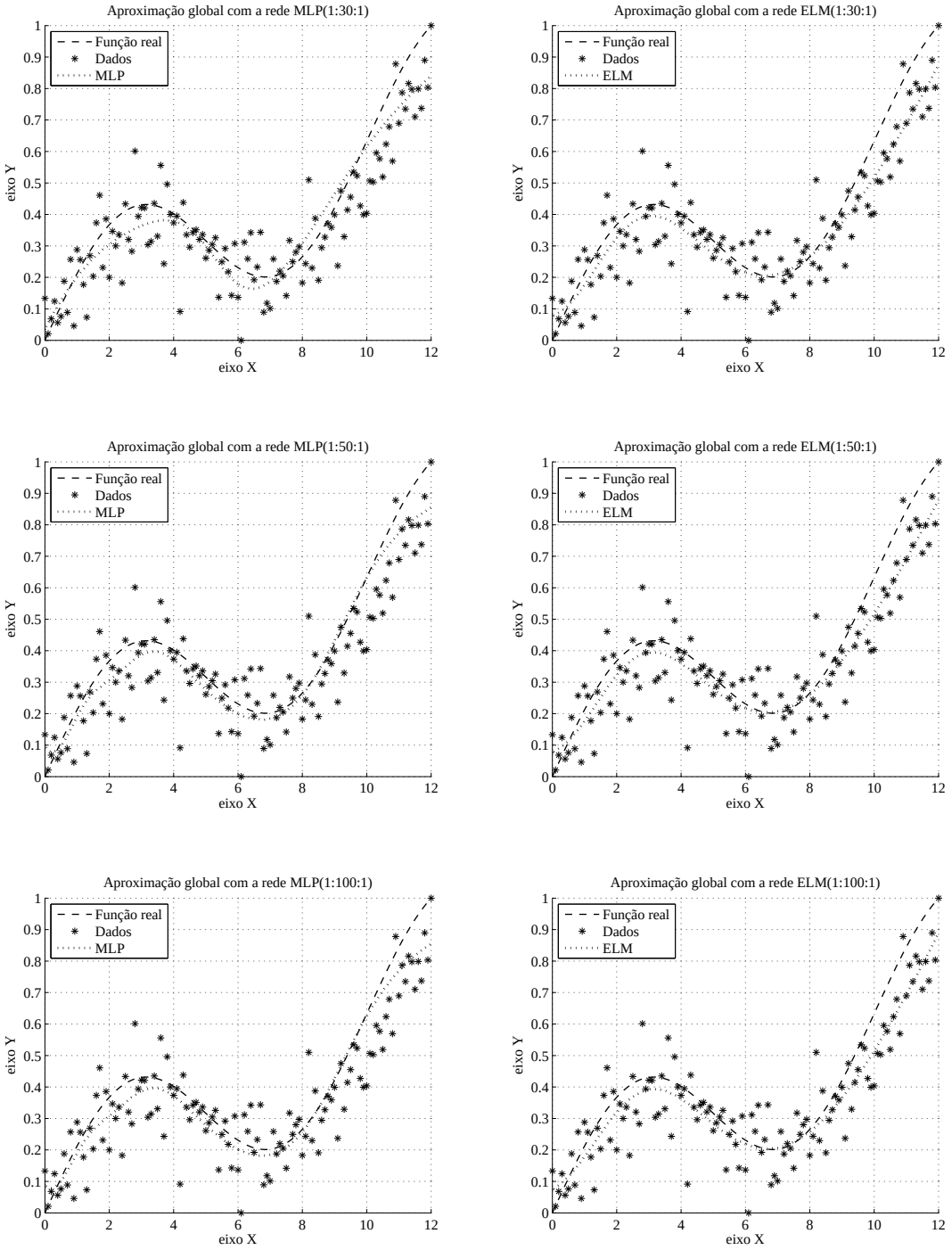


Figura 2.2 Aproximação global usando as redes MLP e ELM com 30, 50 e 100 neurônios ocultos cada.

a quantidade de neurônios da camada escondida variou com base nos seguintes valores: 3, 10, 15, 30, 50, 100. A rede ELM não utiliza etapa de treinamento, e maiores detalhes sobre esta rede serão comentados posteriormente no texto.

Os gráficos dos resultados para as redes MLP e ELM são mostrados nas Figuras 2.1 e 2.2. Ao analisar os resultados, constata-se que a rede ELM apresenta uma melhor aproximação da função para quantidades acima de 30 neurônios, ao contrário da rede MLP em que se vê resultados cada vez piores conforme o aumento desta quantidade, visto a partir de 15 neurônios, demonstrando um aprendizado com *overfitting*, prejudicando o desempenho de generalização do modelo para novos dados (pós-treinamento).

Com base nos resultados obtidos, pode-se constatar a limitação por parte da rede MLP com o aumento no número de neurônios na camada escondida. Já para o caso da rede ELM, o que se vê é uma adequada aproximação da função real. Com esta observação, percebe-se que a rede ELM apresenta melhores resultados com o aumento no número de neurônios na camada escondida, conforme citado no trabalho de Huang et al. (2006). Isso é devido principalmente a falta de uma etapa de treinamento na rede ELM, cujos detalhes serão esclarecidos na Seção 2.3.

A aplicação do conceito de modelo global pode ser estendido para o espaço tridimensional através da aproximação de uma superfície de dados definida a partir de uma função, tal como:

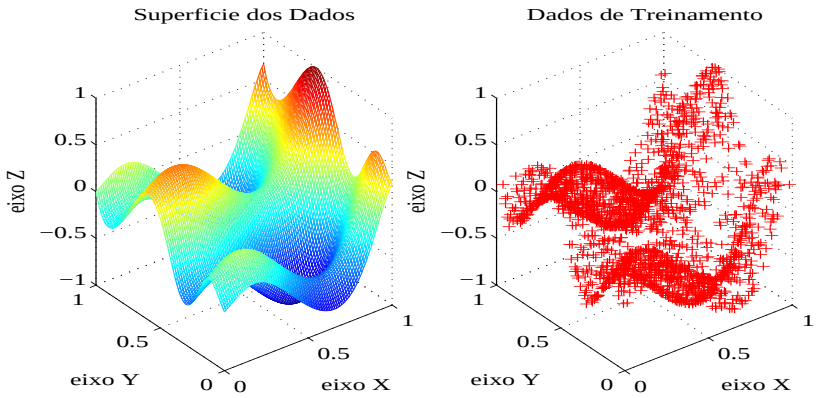
$$z = f(x, y) = 1.3356[1.5(1 - x) + \exp(2x - 1)sen(3\pi(x - 0.6)^2) + \tag{2.2}$$

$$\ldots + \exp(3(y - 0.5))sen(4\pi(y - 0.9)^2)]. \tag{2.3}$$

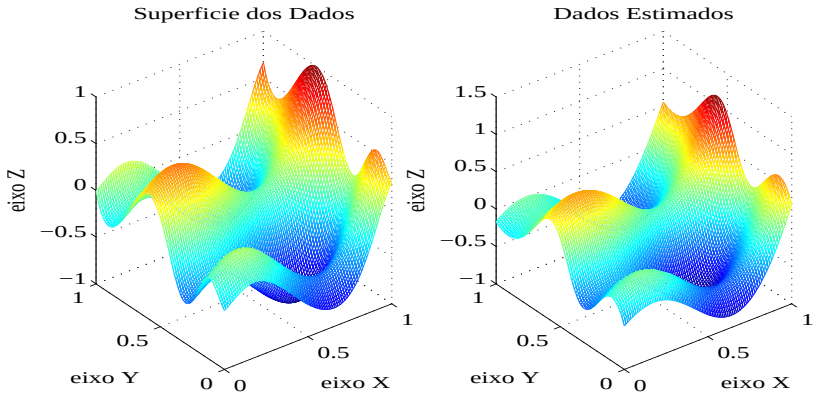
Os valores assumidos por x, y pertencem ao intervalo [0,1]. Para o treinamento da rede MLP foram utilizadas 1000 épocas, sendo o número de neurônios da camada intermediária igual a 30.

Na Figura 2.1(a), apresenta-se a superfície real dos dados e ao lado desta figura tem-se o conjunto de dados de treinamento utilizado pela rede MLP. Estes dados foram gerados a partir da função apresentada na Equação 2.2.

Após a etapa de treinamento, a rede MLP é testada com valores de (x, y) no intervalo de [0,1], e o resultado obtido pode ser visto na Figura 2.1(b). A superfície real dos dados é comparada juntamente com a superfície obtida a partir dos resultados de teste. O que se percebe é que a função aqui representada pela superfície dos dados no espaço tridimensional é aproximada como um todo por meio de um único modelo matemático.



(a) Gráficos da superfície original dos dados e do conjunto de treinamento da rede MLP.



(b) Comparação entre a superfície original dos dados e os resultados obtidos pela rede MLP.

Figura 2.3 Rede MLP. Aproximação de uma superfície tridimensional.

Portanto, a curva real tem sua aproximação feita a partir do uso de um modelo global, tal como a rede MLP.

O resultado obtido pela rede MLP demonstra que a superfície gerada pela mesma acompanha a suavidade característica da função, e a diferença existente entre os resultados está principalmente na amplitude dos dados.

2.1.1 Identificação de Sistemas Dinâmicos Não-lineares

Uma larga classe de sistemas dinâmicos não-lineares do tipo SISO (*Single-Input Single-Output*) com entrada u e de saída y pode ser descrito no tempo discreto pelo modelo

entrada-saída NARX[1] (LJUNG, 1999; NORGAARD et al., 2000):

$$\hat{y}(t) = f(\mathbf{v}(t-1)), \tag{2.4}$$

onde $y(t)$ denota a saída no instante de tempo t, $f(\cdot)$ é um mapeamento não-linear, e $\mathbf{v}(t-1) \in \mathbb{R}^{p+q}$ é o vetor de regressores, consistindo de um número finito de $p+q$ entradas e saídas passadas:

$$\mathbf{v}(t-1) = \begin{pmatrix} y(t-1) \\ y(t-2) \\ \vdots \\ y(t-p) \\ u(t-l) \\ u(t-l-1) \\ \vdots \\ u(t-l-q+1) \end{pmatrix}. \tag{2.5}$$

A ordem dinâmica do sistema é representada pelos números de memória de entrada q ($q \geq 1$), e memória de saída p ($p \geq 1$), sendo $q \leq p$. O parâmetro l ($l \geq 0$) é um termo de atraso de propagação do sinal, chamado de tempo morto (*dead time*). Sem perda de generalidade, assume-se $l = 0$ nesta tese, então obtém-se o seguinte modelo NARX:

$$\mathbf{v}(t-1) = \begin{pmatrix} y(t-1) \\ y(t-2) \\ \vdots \\ y(t-p) \\ u(t) \\ u(t-1) \\ \vdots \\ u(t-q+1) \end{pmatrix}. \tag{2.6}$$

Para a tarefa de modelagem inversa, na qual este trabalho se interessa, os modelos de redes neurais tem o propósito de aproximar o mapeamento inverso $f^{-1}(\cdot)$, dado por

$$\hat{u}(t) = f^{-1}(\mathbf{v}(t-1)), \tag{2.7}$$

[1]NARX, sigla para Nonlinear Auto-Regressive model with eXogenous inputs.

onde o vetor de regressores é agora definido como

$$\mathbf{v}(t-1) = \begin{pmatrix} y(t-1) \\ y(t-2) \\ \vdots \\ y(t-p) \\ u(t-1) \\ \vdots \\ u(t-q) \end{pmatrix}. \tag{2.8}$$

O objetivo é estimar a entrada $u(t)$ a partir dos valores antecessores e atuais das variáveis de entrada e saída, respectivamente. Este tipo de modelo inverso não-linear[2] e a identificação *online* correspondente de seus parâmetros são úteis, por exemplo, para controle em tempo real (NORGAARD et al., 2000).

2.2 Arquitetura da Rede MLP

Basicamente uma rede MLP é constituída de um conjunto de unidades de entrada (que recebem os sinais), uma ou mais camadas intermediárias (ou ocultas) compostas por neurônios não-lineares, e uma camada de saída composta por um ou mais neurônios lineares ou não-lineares. Os neurônios das camadas intermediárias são chamados de neurônios intermediários ou ocultos pelo fato de não terem acesso direto à saída da rede. Embora possa ter qualquer número de camadas ocultas, o enfoque nesta tese é a rede MLP com apenas uma camada de neurônios ocultos.

O principal motivo para escolha preferencial de redes de uma camada escondida está devidamente associado ao fato de que para tarefa de identificação de sistemas é a forma mais indicada por alguns trabalhos (NARENDRA; PARTHASARATHY, 1990).

A Figura 2.4 mostra a arquitetura de uma rede MLP *feedforward* com uma camada oculta e totalmente conectada, aplicada à modelagem direta. Isto significa que um neurônio em qualquer camada da rede é conectado a todas as unidades/neurônios da camada seguinte, e que um sinal de entrada propaga-se na rede da entrada para saída, isto é, da esquerda para direita, de camada em camada (HAYKIN, 2008). Nesta arquitetura $x_1, x_2, \ldots, x_{p+q}$ são as componentes do vetor de entrada $\mathbf{x}$ o qual deve ser acrescido da componente fixa $x_0 = -1$, correspondente ao limiar (*bias*). Portanto, o vetor de entrada

[2]Quando usado para controle inverso-direto, o termo $y(t)$ em (2.8) é substituído por um sinal de referência, $r(t)$, que é usualmente assumido estar disponível no tempo $t-1$.

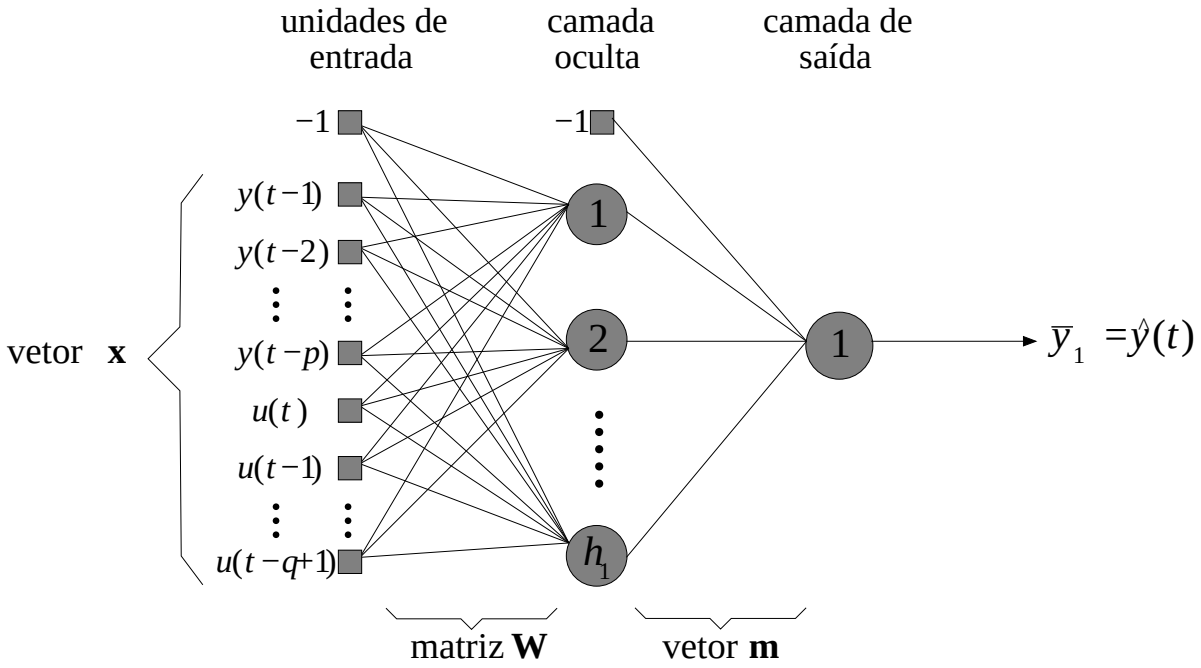


Figura 2.4 Rede MLP com uma camada oculta para modelagem direta.

da rede MLP, de dimensionalidade $p + q + 1$ (com a inclusão do limiar) em t torna-se

$$\mathbf{x}(t) = \begin{pmatrix} -1 \\ x_1(t) \\ x_2(t) \\ \vdots \\ x_p(t) \\ x_{p+1}(t) \\ x_{p+2}(t) \\ \vdots \\ x_{p+q}(t) \end{pmatrix} = \begin{pmatrix} -1 \\ y(t-1) \\ y(t-2) \\ \vdots \\ y(t-p) \\ u(t) \\ u(t-1) \\ \vdots \\ u(t-q+1) \end{pmatrix}. \tag{2.9}$$

Pode-se definir a arquitetura da rede MLP aplicada à modelagem inversa de um sistema dinâmico, vista na Figura 2.5, em que o vetor de entrada $\mathbf{x}(t)$ é representado na

seguinte forma, acrescentando a componente fixa $x_0 = -1$:

$$\mathbf{x}(t) = \begin{pmatrix} -1 \\ x_1(t) \\ x_2(t) \\ \vdots \\ x_p(t) \\ x_{p+1}(t) \\ x_{p+2}(t) \\ \vdots \\ x_{p+q}(t) \end{pmatrix} = \begin{pmatrix} -1 \\ y(t-1) \\ y(t-2) \\ \vdots \\ y(t-p) \\ u(t-1) \\ u(t-2) \\ \vdots \\ u(t-q) \end{pmatrix}. \tag{2.10}$$

É possível definir a matriz $\mathbf{X} \in \mathbb{R}^{(p+q+1)\times N_1}$ reunindo todos os vetores-coluna acima, para as N_1 amostras do conjunto de dados utilizado na etapa de treinamento da rede. A matriz $\mathbf{X}$ pode então ser escrita como

$$\mathbf{X} = [\mathbf{x}(1) \mid \mathbf{x}(2) \mid \ \ldots \ \mid \mathbf{x}(N_1)]. \tag{2.11}$$

Na Figura 2.5, h_1 $(2 \leq h_1 < \infty)$ representa o número de neurônios da camada oculta. Estes neurônios desempenham um papel crucial na rede MLP porque agem como detectores de características (HAYKIN, 2008). Conforme o processo de aprendizagem da rede avança, estes neurônios começam gradualmente a "descobrir" as características salientes presentes nos dados de treinamento. Daí então realizam uma transformação não-linear nos dados de entrada para um novo espaço, chamado espaço oculto ou espaço de características.

Na camada de saída da rede MLP, a quantidade de neurônios é igual a unidade $(g = 1)$, pois a tarefa de identificação de sistemas dinâmicos a qual se aplica esta rede neural requer somente um único elemento do tipo linear, para estimar o valor desejado que será a entrada atual $u(t)$ (ver Eq. (2.7)) definido na modelagem inversa de um sistema do tipo SISO.

A matriz $\mathbf{W} \in \mathbb{R}^{h_1\times(p+q+1)}$ é formada por todas as conexões (ou pesos) sinápticas entre as unidades de entrada e os neurônios da camada oculta. Portanto, cada elemento w_{ij} de $\mathbf{W}$ na Figura 2.5 representa a conexão sináptica entre a j-ésima entrada e o i-ésimo

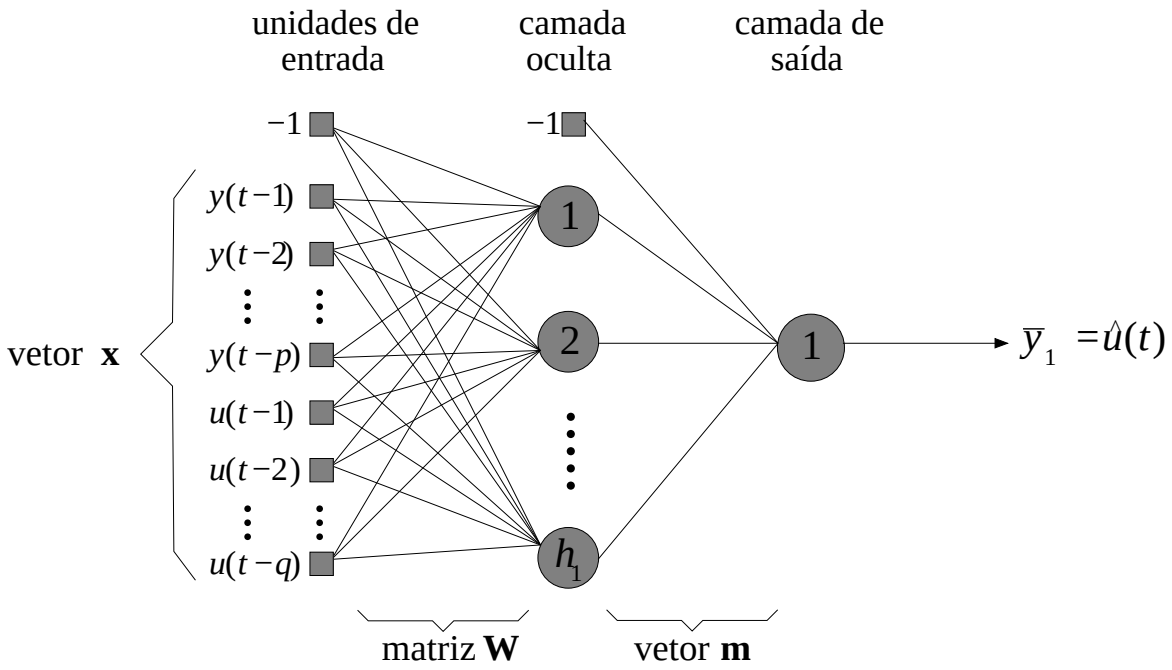


Figura 2.5 Rede MLP com uma camada oculta para modelagem inversa.

neurônio da camada oculta. A matriz $\mathbf{W}$ pode então ser representada por

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_{h_1}^T \end{bmatrix}, \tag{2.12}$$

em que cada linha $\mathbf{w}_i^T = [w_{i0}\ w_{i1}\ \ldots\ w_{i(p+q)}]$ corresponde aos pesos sinápticos entre o i-ésimo neurônio oculto e as $p+q+1$ unidades de entrada, incluindo o limiar.

Seguindo o mesmo princípio, define-se $\mathbf{m}^T \in \mathbb{R}^{1\times(h_1+1)}$ como o vetor formado pelas conexões sinápticas entre os neurônios da camada oculta e o neurônio da camada de saída. Cada componente m_i de $\mathbf{m}$ representa a conexão sináptica entre o i-ésimo neurônio oculto e o neurônio de saída. O vetor $\mathbf{m}$ pode então ser escrito como

$$\mathbf{m} = \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{h_1} \end{bmatrix}, \tag{2.13}$$

já incluindo o limiar no vetor de pesos.

Uma vez mostrada a arquitetura da rede MLP com uma camada oculta e apresentados os principais parâmetros, faz-se necessária uma discussão sobre o algoritmo a ser utilizado no treinamento da rede. Para esta finalidade foi escolhido o algoritmo de retropropagação do erro (*error back-propagation*), o qual é o tipo mais adotado entre os artigos vistos na literatura, deixando assim a apresentação do mesmo para alguns livros-textos conhecidos (PRINCIPE et al., 2000; HAYKIN, 2008), omitindo a necessidade de defini-lo neste trabalho.

Para tarefa de modelagem inversa de um sistema dinâmico é necessário somente um neurônio na camada de saída, e a sua função de ativação pode ser linear ou do tipo tangente hiperbólica. No caso da camada oculta, a função de ativação dos neurônios pode ser do tipo sigmóide logística ou tangente hiperbólica. Quando for feito algum comentário no texto desta tese referente à arquitetura da rede MLP, adotar-se-á a seguinte representação: MLP $(p+q+1 : h_1 : 1)$, se referindo à dimensão do vetor de entrada $(p+q+1)$ baseado no modelo NARX e acrescido do limiar, número de neurônios ocultos (h_1) e um único neurônio linear na camada de saída. Esta representação será adotada também pela rede ELM.

Na próxima seção será apresentada a rede neural global ELM, que é uma rede neural *feedforward* com apenas uma camada oculta de neurônios sigmoidais. A principal diferença da rede ELM em relação a rede MLP está na ausência de treinamento dos pesos sinápticos da camadas oculta. Os pesos de saída podem ser obtidos pelo método dos mínimos quadrados ou por um equivalente recursivo. Mais detalhes são fornecidos a seguir.

2.3 Máquina de Aprendizado Extremo

Para compreender melhor o que será exposto nesta seção deve-se levar em consideração algumas informações preliminares sobre esta rede neural. Inicialmente, considera-se esta rede sendo do tipo *feedforward*, ou seja sem realimentação, com apenas uma camada de neurônios ocultos e um neurônio de saída. Esta arquitetura de rede neural conhecida como Máquina de Aprendizado Extremo (*Extreme Machine Learning*, ELM) (HUANG et al., 2006) tem a mesma arquitetura de uma rede MLP de uma camada oculta, porém apresenta uma fase de aprendizado mais rápida que a da rede MLP. Uma breve descrição da arquitetura da rede ELM e do seu funcionamento é feita a seguir.

Os neurônios da camada oculta (primeira camada de pesos sinápticos) são representa-

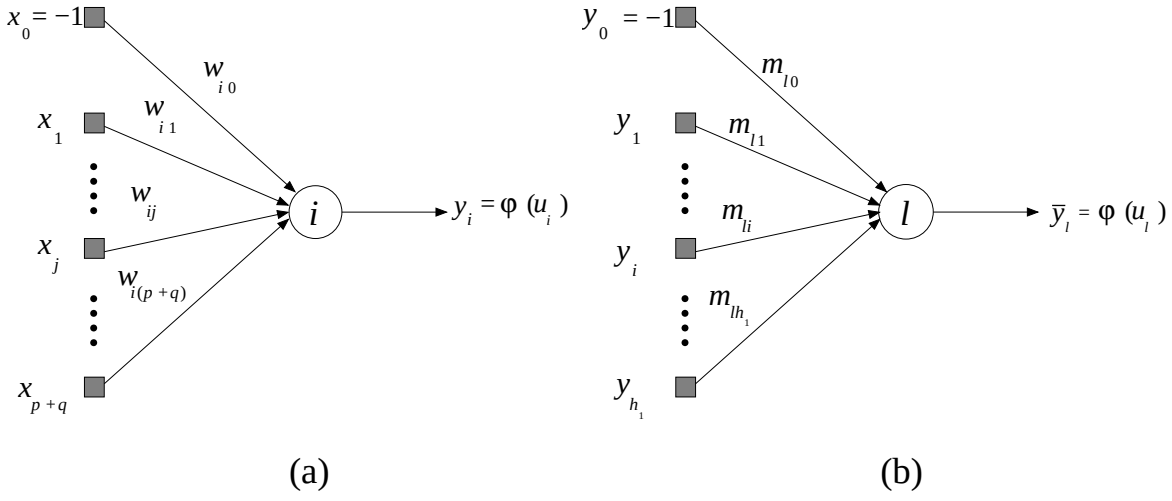


Figura 2.6 Rede ELM: (a) Neurônio da camada oculta. (b) Neurônio da camada de saída.

dos conforme mostrado na Figura 2.5(a), enquanto os neurônios da camada de saída (segunda camada de pesos sinápticos) são representados conforme mostrado na Figura 2.5(b).

O vetor de pesos associado a cada neurônio i da camada oculta, é

$$\mathbf{w}_i = \begin{bmatrix} w_{i0} \\ w_{i1} \\ \vdots \\ w_{i(p+q)} \end{bmatrix} = \begin{bmatrix} \theta_i \\ w_{i1} \\ \vdots \\ w_{i(p+q)} \end{bmatrix}, \tag{2.14}$$

em que θ_i é o limiar associado ao neurônio i. Os neurônios desta camada são chamados de neurônios ocultos por não terem acesso direto à saída da rede. De modo semelhante, o vetor de pesos associado a cada neurônio l da camada de saída é representado como

$$\mathbf{m}_l = \begin{bmatrix} m_{l0} \\ m_{l1} \\ \vdots \\ m_{lh_1} \end{bmatrix} = \begin{bmatrix} \theta \\ m_1 \\ \vdots \\ m_{h_1} \end{bmatrix}, \tag{2.15}$$

em que θ_l é o limiar associado ao neurônio de saída l. Como visto anteriormente para a rede MLP, para aplicação de identificação de sistemas dinâmicos usa-se somente um único neurônio na camada de saída, transformando o vetor do l-ésimo neurônio da Equa-

ção (2.15) no vetor semelhante a este:

$$\mathbf{m} = \begin{bmatrix} m_{10} \\ m_{11} \\ \vdots \\ m_{1h_1} \end{bmatrix} = \begin{bmatrix} \theta_1 \\ m_{11} \\ \vdots \\ m_{1h_1} \end{bmatrix}. \tag{2.16}$$

Como a rede ELM não possui uma etapa de treinamento semelhante ao da rede MLP, cuja atualização dos pesos sinápticos envolve cálculos considerando os sentidos direto e reverso como é comentado na literatura, quando se utiliza o algoritmo *backpropagation*. No caso da rede ELM o aprendizado é executado em três passos distintos que contribuem conjuntamente para obtenção do vetor de pesos sinápticos da camada de saída. Estes três passos são os seguintes.

2.3.1 Passo 1: Inicialização Aleatória dos Pesos da Camada Oculta

Esta etapa envolve a escolha dos pesos $w_{ij}, i = 0, \ldots, h_1$ e $j = 0, \ldots, p+q$, aleatoriamente, tipicamente é escolhida uma distribuição uniforme ou normal, isto é:

$$w_{ij} \sim U(a,b) \;\text{ ou }\; w_{ij} \sim N(0, \sigma^2) \tag{2.17}$$

em que $U(a,b)$ é uma distribuição no intervalo (a,b), e $N(0,\sigma^2)$ é uma distribuição normal com média zero e variância σ^2.

Em ambientes de programação tais como Matlab© e Octave, esta fase é facilmente implementada em uma linha apenas de código. Para isso, precisamos definir uma matriz de pesos $\mathbf{W}$, com h_1 linhas e $p+q+1$ colunas:

$$\mathbf{W} = \begin{bmatrix} w_{10} & w_{11} & \ldots & w_{1(p+q)} \\ w_{20} & w_{21} & \ldots & w_{2(p+q)} \\ \vdots & \vdots & \vdots & \vdots \\ w_{h_1 0} & w_{h_1 1} & \ldots & w_{h_1(p+q)} \end{bmatrix}_{h_1 \times (p+q+1)} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_{h_1}^T \end{bmatrix}, \tag{2.18}$$

onde nota-se que a i-ésima linha da matrix $\mathbf{W}$ é composta pelo vetor de pesos do i-ésimo neurônio oculto.

2.3.2 Passo 2: Acúmulo das Saídas dos Neurônios Ocultos

Este passo corresponde a etapa de cálculo das ativações dos neurônios ocultos e suas respectivas saídas. Este passo destina-se para obtenção de uma matriz formada a partir das saídas dos neurônios da camada oculta calculada conforme a apresentação de cada vetor de entrada à rede neural.

O fluxo de sinais se dá dos neurônios de entrada para os neurônios de saída, passando obviamente pelos neurônios da camada oculta. Por isso, diz-se que a informação está fluindo no sentido direto, ou seja:

Entrada $\rightarrow$ Camada Oculta $\rightarrow$ Camada de saída

Assim, após a apresentação de um vetor de entrada $\mathbf{x}$, em t, o primeiro é calcular a ativação de cada neurônio da camada oculta:

$$u_i(t) = \sum_{j=0}^{p+q} w_{ij} x_j(t) = \mathbf{w}_i^T \mathbf{x}(t), \quad i = 1, \ldots, h_1, \tag{2.19}$$

onde h_1 é o número de neurônios da camada oculta.

A operação sequencial da Eq. (2.19) pode ser feita de uma única vez se for utilizada uma notação vetorial. Neste caso, tem-se que o vetor de ativações $\mathbf{u}(t) \in \mathbb{R}^{h_1}$ na iteração t é calculado como

$$\mathbf{u}(t) = \mathbf{W}\mathbf{x}(t). \tag{2.20}$$

Em seguida, as saídas correspondentes são calculadas como

$$y_i(t) = \varphi\left[u_i(t)\right] = \varphi\left[\sum_{j=0}^{p+q} w_{ij} x_j(t)\right] = \varphi\left[\mathbf{w}_i^T \mathbf{x}(t)\right], \tag{2.21}$$

em que $\varphi(\cdot)$ é uma não-linearidade do tipo sigmoidal, $i = 1, \ldots, h_1$. Normalmente são utilizados dois tipos de função de ativação, a sigmóide dada por

$$y(t) = \varphi(v) = \frac{1}{1 + \exp\{-v\}}, \tag{2.22}$$

e a tangente hiperbólica, cuja expressão é dada por

$$y(t) = \varphi(v) = \frac{1 - \exp\{-v\}}{1 + \exp\{-v\}}. \tag{2.23}$$

O domínio destas funções é o conjunto dos números reais. A imagem da função sigmóide logística está restrita ao intervalo $[0, 1]$, enquanto a imagem da tangente hiperbó-

lica está restrita a $[-1, +1]$. De um extremo a outro ambas são sempre monotonicamente crescentes. Os gráficos das funções sigmóide e tangente hiperbólica estão mostrados na Figura 2.6.

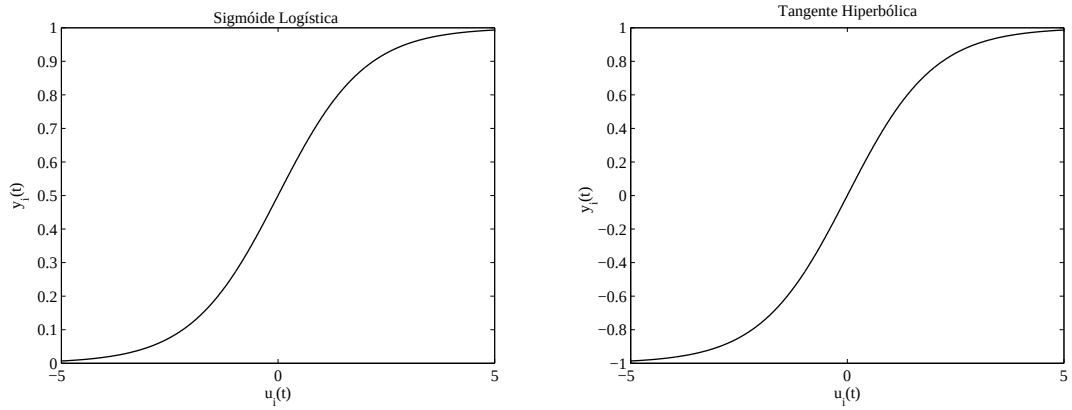


Figura 2.7 Funções de ativação sigmóide logística e tangente hiperbólica.

Em notação matriz-vetor, a Eq. (2.21) pode ser escrita como

$$\mathbf{y}(t) = \varphi(\mathbf{u}(t)) = \varphi(\mathbf{W}\mathbf{x}(t)), \tag{2.24}$$

em que a função de ativação $\varphi(\cdot)$ é aplicada a cada um dos h_1 componentes do vetor $\mathbf{u}(t)$.

Para cada vetor de entrada $\mathbf{x}(t), t = 1, \ldots, N_1$, tem-se um vetor $\mathbf{y}(t)$ correspondente, que deve ser organizado como uma coluna de uma matriz $\mathbf{Y}$. Esta matriz terá h_1 linhas por N_1 colunas:

$$\begin{aligned} \mathbf{Y} &= [\mathbf{y}(1)\ |\mathbf{y}(2)|\ \ldots\ |\mathbf{y}(N_1)], \\ &= \begin{bmatrix} y_1(1) & y_1(2) & \cdots & y_1(N_1) \\ y_2(1) & y_2(2) & \cdots & y_2(N_1) \\ \vdots & \vdots & \vdots & \vdots \\ y_{h_1}(1) & y_{h_1}(2) & \cdots & y_{h_1}(N_1) \end{bmatrix}_{h_1 \times N_1} . \end{aligned} \tag{2.25}$$

Uma linha de "-1" deve ser adicionada à matriz $\mathbf{Y}$, correspondendo à entrada cons-

tante associada ao limiar do neurônio de saída, passando a ser escrita como

$$\begin{aligned}\bar{\mathbf{Y}} &= [\bar{\mathbf{y}}(1)\,|\bar{\mathbf{y}}(2)|\,\ldots\,|\bar{\mathbf{y}}(N_1)], \\ &= \begin{bmatrix} -1 & -1 & \cdots & -1 \\ y_1(1) & y_1(2) & \cdots & y_1(N_1) \\ y_2(1) & y_2(2) & \cdots & y_2(N_1) \\ \vdots & \vdots & \vdots & \vdots \\ y_{h_1}(1) & y_{h_1}(2) & \cdots & y_{h_1}(N_1) \end{bmatrix}_{(h_1+1)\times N_1} .\end{aligned} \tag{2.26}$$

A matriz $\bar{\mathbf{Y}}$ é usada no Passo 3 para determinar os valores dos pesos do neurônio de saída da rede ELM.

2.3.3 Passo 3: Determinação dos Pesos do Neurônio de Saída

Para o problema de identificação do modelo inverso, cada vetor de entrada $\mathbf{x}(t), t = 1, \ldots, N_1$, tem-se um escalar como a saída desejada $u(t)$ correspondente. Ao organizar estes N_1 escalares como um vetor-coluna $\mathbf{d}$, tem-se então um vetor com N_1 componentes:

$$\mathbf{d} = \begin{bmatrix} u(1) \\ u(2) \\ \vdots \\ u(N_1) \end{bmatrix}_{N_1\times 1} . \tag{2.27}$$

Pode-se entender o cálculo dos pesos da camada de saída como o cálculo dos parâmetros de um mapeamento linear entre a camada oculta e a camada de saída. O papel de "vetor de entrada" para a camada de saída em t é desempenhado pelo vetor $\bar{\mathbf{y}}(t)$, enquanto a "saída" é representado pelo escalar $u(t)$. Assim, busca-se determinar o vetor $\mathbf{m}$ que melhor represente a transformação

$$u(t) = \mathbf{m}^T \bar{\mathbf{y}}(t). \tag{2.28}$$

Para isso, pode-se usar o método dos mínimos quadrados, também conhecido como método da pseudoinversa (PRINCIPE et al., 2000). Assim, de posse da matriz $\bar{\mathbf{Y}}$ e do vetor $\mathbf{d}$, o vetor de pesos $\mathbf{m}$ é determinado por meio da seguinte expressão:

$$\mathbf{m} = \left(\bar{\mathbf{Y}}\bar{\mathbf{Y}}^T\right)^{-1}\bar{\mathbf{Y}}\mathbf{d}. \tag{2.29}$$

É importante destacar que a matriz inversa $\left(\bar{\mathbf{Y}}\bar{\mathbf{Y}}^T\right)^{-1}$ na Equação (2.29) não depende do tamanho do conjunto de treinamento (N_1), mas sim do número de neurônios ocultos (h_1).

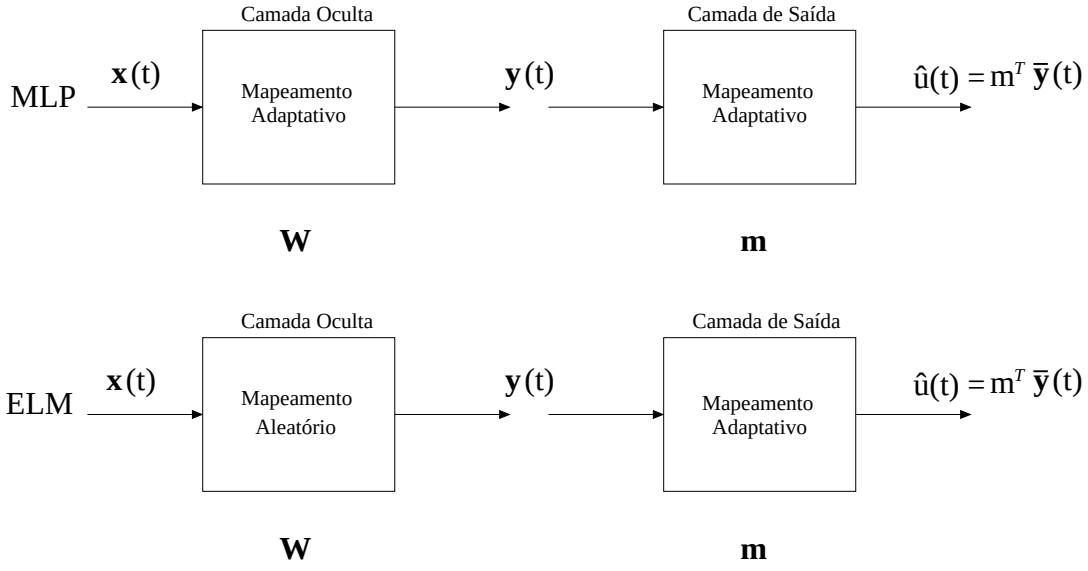


Figura 2.8 Mapeamento realizado pelas arquiteturas MLP e ELM.

2.3.4 Uso da Rede ELM Após o Treinamento

Durante o uso da rede após a etapa de treinamento, espera-se que a mesma seja capaz de generalizar o conhecimento adquirido para novos dados de entrada. Por generalização entende-se a habilidade da rede em utilizar o conhecimento armazenado nos seus pesos e limiares para gerar saídas coerentes para novos vetores de entrada. A generalização é considerada boa quando a rede, durante o treinamento, foi capaz de capturar (aprender) adequadamente a relação entrada-saída do mapeamento de interesse.

Uma vez determinada a matriz $\mathbf{W}$ e o vetor $\mathbf{m}$ de pesos, tem-se a rede ELM pronta para uso. Como o neurônio de saída é linear, sua saída é calculada como

$$\hat{u}(t) = \mathbf{m}^T \bar{\mathbf{y}}(t), \tag{2.30}$$

em que o vetor de saídas dos neurônios ocultos, $\bar{\mathbf{y}}(t)$, é calculado como

$$\bar{\mathbf{y}}(t) = \begin{bmatrix} -1 \\ \mathbf{y}(t) \end{bmatrix} = \begin{bmatrix} -1 \\ \varphi(\mathbf{W}\mathbf{x}(t)) \end{bmatrix}, \tag{2.31}$$

Para finalizar, pode-se destacar algumas diferenças que há entre esse modelo e a ar-

quitetura MLP. Com base na Figura 2.7, vê-se que a rede MLP realiza mapeamento não-linear entre o vetor de entrada $\mathbf{x}(t)$ e a saída correspondente $u(t)$, utilizando uma camada oculta e uma camada de saída, cujos pesos são atualizados adaptativamente por meio dos algoritmos de treinamento adotados comumente na literatura, tais como retro-propagação do erro ou Levenberg-Marquardt. No caso da rede ELM, a grande vantagem a ser destacada é a redução do tempo de projeto da rede, pois para este modelo não há treinamento dos pesos dos neurônios ocultos, uma vez que estes são definidos aleatórios (ver Eq. (2.17)), enquanto o vetor de pesos da camada de saída ($\mathbf{m}$) é obtido através do método da pseudoinversa (ver Eq. (2.29)), tornando o projeto da rede ELM bem mais veloz. Um sumário do algoritmo de treinamento ELM é apresentado na Tabela 2.1.

2.4 Resumo

Neste capítulo foi apresentada inicialmente a arquitetura da rede MLP *feedforward* com uma camada oculta e totalmente conectada, destacando os principais parâmetros envolvidos. Além disso, foi introduzida uma notação matricial para representar os dados de entrada, as matrizes de pesos e as respostas dos neurônios ocultos e de saída. Esta notação torna mais simples o entendimento da rede neurais MLP e também da ELM.

Por último, foi apresentada uma recente arquitetura neural de grande interesse e utilizada em diversas aplicações no meio acadêmico que se destaca em comparação à rede MLP em termos da agilidade e eficiência de treinamento. Esta rede é a ELM. Como dito anteriormente, a grande versatilidade desta rede está exatamente em não possuir no sentido real da palavra uma etapa responsável pelo treinamento direto dos parâmetros ajustáveis, tais como os pesos sinápticos das camadas oculta e de saída. O que se tem nesta arquitetura inicialmente é uma atribuição aleatória dos valores dos pesos ocultos, seguido pelo cálculo de suas saídas, conforme se apresente novos vetores de entrada, e posteriormente, a obtenção dos pesos da camada de saída a partir das saídas acumuladas na camada oculta e os valores referentes às saídas desejadas da rede neural.

Estas duas redes neurais são vistas na identificação de sistemas dinâmicos como modelos globais capazes de aproximar uma função que define o mapeamento entrada-saída existente nos dados provenientes de uma planta industrial. São ditas globais pois representam este mapeamento como sendo uma única função em todo domínio de interesse.

No próximo capítulo serão apresentadas os principais algoritmos de quantização vetorial que serão utilizados nesta tese. O estudo destes algoritmos terá uma grande relevância

Tabela 2.1 Algoritmo ELM para Identificação Inversa do Modelo.

Algoritmo ELM

Entradas

$\mathbf{x}(t)$: vetor de entrada, dimensão $(p+q+1)\times 1$ $u(t)$: variável observada

Algoritmo

1. Inicialização aleatória dos pesos ocultos $(t=0)$

$\mathbf{w}_i \sim U(0,1)$ ou $\mathbf{w}_i \sim N(0,\sigma^2),\ i=1,2,\ldots,h_1$

2. Obtenção do vetor de pesos de saída $(t=1,2,\ldots,N_1)$

2.1 Cálculo das ativações dos neurônios ocultos:

$u_i(t) = \mathbf{w}_i^T\mathbf{x}(t)$,

$y_i(t) = \varphi\left[u_i(t)\right] = \varphi\left[\mathbf{w}_i^T\mathbf{x}(t)\right]$,

onde $\varphi(\cdot)$ é uma função sigmoidal:

$\varphi(v) = \frac{1}{1+\exp\{-v\}}$, (função logística)

$\varphi(v) = \frac{1-\exp\{-v\}}{1+\exp\{-v\}}$, (tangente hiperbólica)

2.2 Acúmulo das ativações dos neurônios ocultos:

$$\bar{\mathbf{Y}} = \begin{bmatrix} -1 & -1 & \cdots & -1 \\ y_1(1) & y_1(2) & \cdots & y_1(N_1) \\ \vdots & \vdots & \vdots & \vdots \\ y_{h_1}(1) & y_{h_1}(2) & \cdots & y_{h_1}(N_1) \end{bmatrix}_{(h_1+1)\times N_1}.$$

2.3 Cálculo do vetor de pesos $\mathbf{m}\in\mathbb{R}^{h_1+1}$:

$\mathbf{d} = [u(1)\ |u(2)|\ \ldots\ |u(N_1)]^T$,

$\mathbf{m} = \left(\bar{\mathbf{Y}}\bar{\mathbf{Y}}^T\right)^{-1}\bar{\mathbf{Y}}\mathbf{d}$.

3. Etapa de Teste $(t=1,2,\ldots,N_2)$

3.1 Cálculo das ativações dos neurônios ocultos:

$$\bar{\mathbf{y}}(t) = \begin{bmatrix} -1 \\ \mathbf{y}(t) \end{bmatrix} = \begin{bmatrix} -1 \\ \varphi(\mathbf{W}\mathbf{x}(t)) \end{bmatrix}.$$

3.2 Cálculo da saída da rede: $\hat{u}(t) = \mathbf{m}^T\bar{\mathbf{y}}(t)$.

3.3 Cálculo do erro: $e(t) = u(t) - \hat{u}(t)$.

Saídas

$\hat{u}(t)$: saída estimada $e(t)$: erro

para a compreensão das arquiteturas que serão abordadas neste trabalho, e dos testes estatísticos que serão utilizados pelos mesmos para comparação dos seus resultados.

3 *Aprendizagem Competitiva para Quantização Vetorial*

Neste capítulo serão descritos os algoritmos de aprendizagem competitiva voltados para quantização vetorial que serão utilizados neste trabalho. Estes algoritmos também são chamados de quantizadores vetoriais neurais a fim de diferenciá-los dos quantizadores vetoriais de origem estatística, tal como o conhecido algoritmo K-médias. Os algoritmos a serem apresentados a seguir servirão de bloco construtivo para as arquiteturas propostas nesta tese no Capítulo 5.

3.1 Introdução

Este capítulo tem por objetivo apresentar sucintamente as arquiteturas de redes neurais competitivas utilizadas neste trabalho. O material a ser apresentado neste capítulo é, em grande parte, baseado nas seguintes referências: Kosko (1992), Haykin (2008), Principe et al. (2000), Kohonen (2001) e Frota (2005). Referências adicionais serão citadas quando necessárias.

É importante destacar que esta tese limita o escopo dos algoritmos de quantização vetorial estudados àqueles baseados em vetores-protótipos, sejam eles de origem neural ou não, e cujos protótipos podem ser atualizados após cada apresentação do vetor de entrada (modo sequencial ou padrão-a-padrão) ou após a apresentação de todos os vetores de entrada (modo *batch*). As arquiteturas a serem descritas neste capítulo são listadas a seguir:

- **Métodos estatísticos clássicos:** Algoritmo K-médias, nas versões *batch* e sequencial;
- **Redes neurais competitivas:** *Winner-Take-All* (WTA), *Frequency-Sensitive*

Competitive Learning (FSCL) e *Self-Organizing Map* (SOM);

- **Algoritmos fuzzy**: Fuzzy K-médias (*batch*) e *Fuzzy Competitive Learning* (FCL).

Conforme mencionado anteriormente, o foco principal desta tese está no projeto de modelos lineares locais para identificação inversa de sistemas usando algoritmos de quantização vetorial, sejam estes neurais, fuzzy ou estatísticos. Assume-se inicialmente que o espaço de entrada $\mathcal{X}$ é contínuo e de dimensão $p+q$. Assume-se também a posse de um conjunto de N_1 vetores de entrada $\mathbf{x}(t) \in \mathcal{X} \subset \mathbb{R}^{p+q}$, $t = 1, 2, \cdots, N_1$, selecionados de forma aleatória de $\mathcal{X}$. Na tarefa de identificação inversa usando o modelo NARX, cada vetor $\mathbf{x}(t)$ corresponde a um vetor de regressores de entrada, sendo representado como

$$\mathbf{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{p+q}(t) \end{pmatrix} = \begin{pmatrix} y(t-1) \\ y(t-2) \\ \vdots \\ y(t-p) \\ u(t-1) \\ \vdots \\ u(t-q) \end{pmatrix} \tag{3.1}$$

O vetor de pesos (ou vetor-protótipo) associado a cada neurônio das redes competitivas a serem descritas neste capítulo tem a mesma dimensão do vetor de entrada, sendo representado da seguinte forma:

$$\mathbf{w}_i = \begin{pmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{i(p+q)} \end{pmatrix}, \quad i = 1, 2, \ldots, g, \tag{3.2}$$

onde g é o número total de neurônios da rede.

3.2 Algoritmo *K*-Médias (Versão *Batch*)

O primeiro algoritmo baseado em protótipos a ser descrito é o algoritmo K-médias (MACQUEEN, 1967), também conhecido no campo de quantização vetorial como algoritmo de Linde-Buzo-Gray (LBG) ou algoritmo de Lloyd generalizado (VASUKI; VANATHI, 2006). A aplicação do algoritmo K-médias a um conjunto de N_1 vetores visa encontrar um conjunto

de g protótipos, $\{\mathbf{w}_i\}_{i=1}^{g}$, $g \ll N_1$, que particione os dados de entrada em exatamente g grupos distintos.

A região de influência de determinado protótipo é chamada de partição de Voronoi (ou Dirichlet) daquele protótipo, sendo definida como

$$V_i = \{\mathbf{x} \in \mathbb{R}^{p+q} \mid \|\mathbf{x} - \mathbf{w}_i\| < \|\mathbf{x} - \mathbf{w}_j\|, \forall j \neq i\}, \tag{3.3}$$

em que $\|\cdot\|$ denota a norma euclidiana. Assim, com g protótipos o espaço de entrada é particionado em g regiões de Voronoi.

O algoritmo K-médias provê um método simples para a obtenção de g protótipos a partir da solução do seguinte problema de otimização

$$\min_{\mathbf{w}_i} D = \sum_{i=1}^{g} \sum_{\mathbf{x} \in V_i} \|\mathbf{x} - \mathbf{w}_i\|^2, \tag{3.4}$$

D é conhecida por erro de quantização, erro de reconstrução ou ainda distorção. A minimização é realizada através dos seguintes de passos:

Passo 1 - Seleção aleatória de g vetores do conjunto de dados como protótipos iniciais;

Passo 2 - Separação do conjunto de dados em g regiões de Voronoi V_i, $i = 1, \ldots, g$, de acordo com a Equação (3.3);

Passo 3 - Os novos protótipos são recalculados como as médias aritméticas (centróides) dos dados alocados a cada região de Voronoi V_i, ou seja

$$\mathbf{w}_i = \frac{1}{N_i} \sum_{\mathbf{x} \in V_i} \mathbf{x}, \tag{3.5}$$

em que N_i é o número de vetores pertencentes à célula de Voronoi do i-ésimo protótipo; ou equivalentemente, é o número de vetores de dados para os quais o protótipo $\mathbf{w}_i$ é o mais próximo, segundo a métrica euclidiana.

Os passos 2 e 3 devem ser repetidos até que não haja mudanças substanciais no valor de D (Eq. (3.4)) ou um determinado número máximo de iterações tenha sido alcançado. Um critério de parada comumente usado verifica se a taxa de variação da distorção D está abaixo de um limiar de distorção $0 < \varepsilon \ll 1$ preestabelecido, ou seja

$$\left| \frac{D(t+1) - D(t)}{D(t+1)} \right| < \varepsilon, \tag{3.6}$$

em que o operador $|u|$ denota o valor absoluto de u, e $D(t)$ corresponde ao valor do erro de quantização na t-ésima época de ajuste dos protótipos.

O interesse do algoritmo K-médias *batch* para este trabalho é apenas didático. O interesse maior está em algoritmos de aprendizado padrão-a-padrão, também chamado de aprendizado *online* ou sequencial, devido a sua maior utilização em aplicações de quantização vetorial.

3.3 Algoritmo *K*-Médias (Versão Sequencial)

No algoritmo K-médias *batch*, os protótipos são atualizados somente após todo o conjunto de dados ter sido apresentado. Daí a razão do termo *batch*. Cada apresentação completa do conjunto de dados constitui uma *época* de treinamento. Assim, o treinamento do algoritmo K-médias *batch* se dá época-a-época, até a sua completa convergência.

Uma versão do algoritmo K-médias em que um dos protótipos é atualizado logo após a apresentação de um vetor de entrada é mais apropriada para problemas de quantização vetorial. Tal versão é comumente conhecida como K-médias sequencial ou adaptativo (DARKEN; MOODY, 1990).

Os protótipos do algoritmo K-médias são iniciados como na versão *batch*. Em seguida, a cada iteração t, determina-se o índice $i^*(t)$ do protótipo mais próximo do vetor de entrada atual $\mathbf{x}(t)$. Por fim, atualiza-se o protótipo selecionado por meio da seguinte regra:

$$\mathbf{w}_{i^*}(t+1) = \mathbf{w}_{i^*}(t) + \frac{1}{N_{i^*}(t)}[\mathbf{x}(t) - \mathbf{w}_{i^*}(t)], \tag{3.7}$$

em que $N_{i^*}(t)$ denota o número de vetores de dados para os quais o protótipo $\mathbf{w}_{i^*}$ foi selecionado até a iteração atual. Pode-se mostrar facilmente que os protótipos do algoritmo K-médias sequencial convergem para a mesma solução que os protótipos da versão *batch*. Uma maior sensibilidade à inicialização dos protótipos é uma limitação importante da versão sequencial em relação à versão *batch* do algoritmo K-médias.

3.4 Rede WTA

Os modelos neurais competitivos avaliados neste trabalho são baseados na distância euclidiana como métrica utilizada para a determinação do neurônio vencedor. Neste caso, tem-se que um certo neurônio i é escolhido como o neurônio vencedor, simbolizado como

$i^*(t)$, para o vetor de entrada atual, se a seguinte relação for satisfeita:

$$i^*(t) = \arg\min_i \|\mathbf{x}(t) - \mathbf{w}_i(t)\|, \tag{3.8}$$

na qual $i^*(t)$ é o índice do neurônio vencedor na rede, $\mathbf{x}(t) \in \mathbb{R}^{p+q}$ é um vetor de entrada da rede na iteração t e $\mathbf{w}_i(t) \in \mathbb{R}^{p+q}$, é o vetor de pesos associado ao neurônio i. Os algoritmos descritos a seguir têm sua operação baseada na Equação (3.8).

No algoritmo competitivo mais simples, conhecido como *Winner-take-all* (WTA), durante a fase de treinamento apenas o neurônio vencedor tem seu vetor de pesos $\mathbf{w}_{i^*}(t)$ atualizado em resposta a um dado vetor de entrada $\mathbf{x}(t)$. O treinamento da rede WTA é resumido a seguir:

Passo 1 - Atribuição de valores iniciais aos g protótipos;

Passo 2 - Apresentação de um vetor de treinamento atual $\mathbf{x}(t)$ à rede;

Passo 3 - Determinação do neurônio vencedor, $i^*(t)$, para o vetor de entrada atual usando a Equação (3.8);

Passo 4 - Atualização do vetor de pesos do neurônio vencedor através da seguinte regra de aprendizagem:

$$\mathbf{w}_{i^*}(t+1) = \mathbf{w}_{i^*}(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{w}_{i^*}(t)], \tag{3.9}$$

em que $0 < \alpha \ll 1$ denota o passo de aprendizagem.

Algumas observações importantes sobre a rede WTA são necessárias.

1. Para uma rede com g neurônios, a inicialização de seus protótipos é comumente feita de duas maneiras: (i) através da seleção aleatória de g vetores de dados; ou (ii) através da atribuição de valores aleatórios uniformemente distribuídos no intervalo $[0, 1]$. Recomenda-se a escolha do primeiro procedimento sempre que possível.

2. A rede WTA é equivalente ao algoritmo K-médias sequencial. De fato, os dois algoritmos coincidem se o passo de aprendizagem for definido como $\alpha = 1/C_{i^*}(t)$.

3. A taxa de aprendizagem pode ser fixa ou variável no tempo. Se for fixa, seu valor deve ser escolhido bem pequeno (e.g. $\alpha = 0,01$ ou $\alpha = 0,001$), a fim de garantir convergência do algoritmo. Se for variável, pode-se escolher uma taxa inicial relativamente alta (e.g. $\alpha_0 = 0,5$) para acelerar o aprendizado e uma taxa final pequena

(e.g. $\alpha_T = 0,01$ ou $\alpha_T = 0,001$) para garantir a convergência do algoritmo. Entre os valores inicial e final pode-se adotar um decaimento linear ou exponencial. Em todas as redes competitivas simuladas nesta tese, utiliza-se um decaimento exponencial:

$$\alpha(t) = \alpha_0 \left(\frac{\alpha_T}{\alpha_0} \right)^{(t/T)}, \tag{3.10}$$

tal que α_0 e α_T ($\alpha_0 \gg \alpha_T$) são os valores inicial e final de α. A velocidade de decaimento é controlada pelo parâmetro T, que simboliza o número máximo de iterações de treinamento.

4. Pode-se mostrar que o vetor de pesos de um determinado neurônio i converge para o centróide (centro de gravidade) do conjunto de vetores de treinamento para o qual o neurônio i foi selecionado vencedor. Para isto basta perceber que os valores esperados de $w_i(t+1)$ e $w_i(t)$ são iguais para $t \to \infty$. Daí, simbolizado por w_i^o o valor final do vetor de pesos w_i, a seguinte expressão pode ser obtida:

$$E\left\{\alpha[\mathbf{x} - \mathbf{w}_i^o]\right\} = 0 \qquad \Rightarrow \qquad \mathbf{w}_i^o = \frac{\int_{V_i} xp(x)dx}{\int_{V_i} p(x)dx}, \tag{3.11}$$

em que V_i é o conjunto de vetores de treinamento para o qual o neurônio i foi selecionado vencedor.

A despeito de sua simplicidade, a rede WTA é afetada por algumas questões que comprometem seriamente seu desempenho:

- **Escolha dos valores iniciais dos pesos da rede:** dependendo dos valores iniciais atribuídos aos pesos, alguns neurônios podem dominar o treinamento, sendo sempre selecionados como vencedores, enquanto outros nunca o são. As unidades não selecionadas são chamadas de unidades mortas (*dead units*).

- **Valorização excessiva da informação contida na entrada $\mathbf{x}(t)$ mais recente:** pela própria natureza do algoritmo, as entradas apresentadas à rede no início do treinamento têm menos influência no valor final dos pesos dos neurônios que aquelas apresentadas por último.

Para minimizar esses problemas, é comum modificar o algoritmo da rede WTA, criando variantes mais eficientes. As principais maneiras de se fazer isso são através da alteração da Equação (3.8) ou da alteração da Equação (3.9). Algumas destas modificações dão origem aos três algoritmos seguintes.

3.5 Rede FSCL

Este algoritmo, chamado *Frequency-Sensitive Competitive Learning* (FSCL), foi proposto por Ahalt et al. (1990). A rede FSCL altera a Equação (3.8) a fim de penalizar neurônios que são escolhidos vencedores com muita freqüência, de modo a permitir vitórias de outros neurônios:

$$i^*(t) = \arg\min_i \{f_i(t) \cdot \|\mathbf{x}(t) - \mathbf{w}_i(t)\|\} \tag{3.12}$$

$$f_i(t) = \left[\frac{C_i}{t}\right]^z, \tag{3.13}$$

em que C_i é o número de vezes que o neurônio i foi escolhido vencedor até o instante t, e $z > 0$ é uma constante. O ajuste dos pesos na rede FSCL continua sendo feito de acordo com a Equação (3.9).

Nota-se que a presença do fator $f_i(t)$ como elemento ponderador da distância euclidiana ajuda a minimizar a ocorrência de unidades mortas. Caso alguns poucos neurônios dominem o processo de competição, ou seja, sejam escolhidos vencedores com maior frequência, eles tenderão a ter valores elevados de C_i com o passar do tempo. Isso fará com que outros neurônios, que antes eram selecionados com menor freqüência, passem a também ser selecionados. Com o passar do tempo, todos os neurônios terão sido escolhidos em um número aproximadamente equivalente de vezes, tornando a competição mais justa.

3.6 Rede SOM

O desenvolvimento de mapeamentos auto-organizáveis como um modelo neural é motivado por uma característica peculiar do cérebro humano. Muitas porções do córtex cerebral estão organizadas de uma forma que diferentes entradas sensoriais são representadas por *mapas computacionais topologicamente organizados*. Assim, mapas computacionais constituem um bloco construtivo básico na infraestrutura de processamento de informação do sistema nervoso. Um mapa computacional é definido por um *arranjo* de neurônios representando diferentes elementos processadores ou filtros, que atuam em paralelo sobre sinais portadores de informação.

A rede SOM foi proposta com o intuito de modelar algumas características essenciais dos mapas computacionais existentes no cérebro sem ter, contudo, a ambição de ser um modelo biologicamente plausível. Seu objetivo principal é transformar um sinal de en-

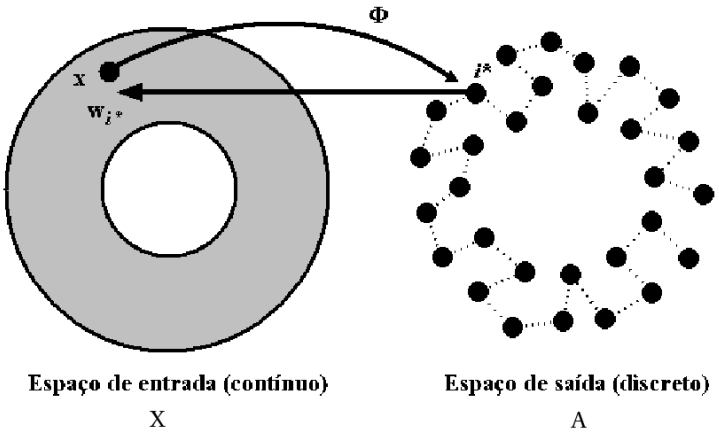


Figura 3.1 Esboço do mapeamento de características Φ e seus elementos constituintes para uma grade do tipo unidimensional.

trada de dimensão qualquer em um arranjo discreto de unidades de processamento. Esta transformação deve ser realizada de forma adaptativa e mantendo relações de similaridade entre os dois espaços.

Seja $\mathcal{X}$ um espaço contínuo de dados de entrada tal que sua topologia é definida por certas relações métricas entre vetores $\mathbf{x} \in \mathcal{X}$. Do espaço $\mathcal{X}$ conhece-se apenas um conjunto finito de vetores (amostras) $\mathbf{x} \in \mathcal{X}$ organizados segundo uma densidade de probabilidade $p(\mathbf{x})$. A topologia do espaço de saída $\mathcal{A}$ é definida pelo arranjo geométrico de um conjunto de neurônios $i \in \mathcal{A}$. Seja Φ uma transformação não-linear, chamada *mapeamento de características* (*feature map*, em inglês), que leva do espaço de entrada $\mathcal{X}$ ao espaço de saída $\mathcal{A}$ (ver Figura 3.1). Matematicamente, tem-se

$$\Phi : \mathcal{X} \rightarrow \mathcal{A}. \tag{3.14}$$

A Figura 3.2 mostra um diagrama esquemático de um arranjo de neurônios comumente usado como espaço discreto de saída. Cada neurônio i no arranjo está totalmente conectado às $p + q$ unidades do vetor de entrada por meio de termos ponderadores, w_{ij}, $j = 1, 2, ..., p+q$, chamados *pesos sinápticos*, agrupados no vetor $\mathbf{w}_i$. Esta rede possui uma estrutura de propagação para frente (feedforward) com uma única camada computacional consistindo de g neurônios normalmente dispostos em linhas e colunas (configuração retangular). Outra configuração muito comum é a hexagonal, bastante efetiva para fins de visualização do mapeamento Φ. Um arranjo unidimensional é um caso especial da configuração mostrada na Figura 3.2. Neste caso, a camada computacional consiste simplesmente de uma única linha ou coluna de neurônios (ver Figura 3.2).

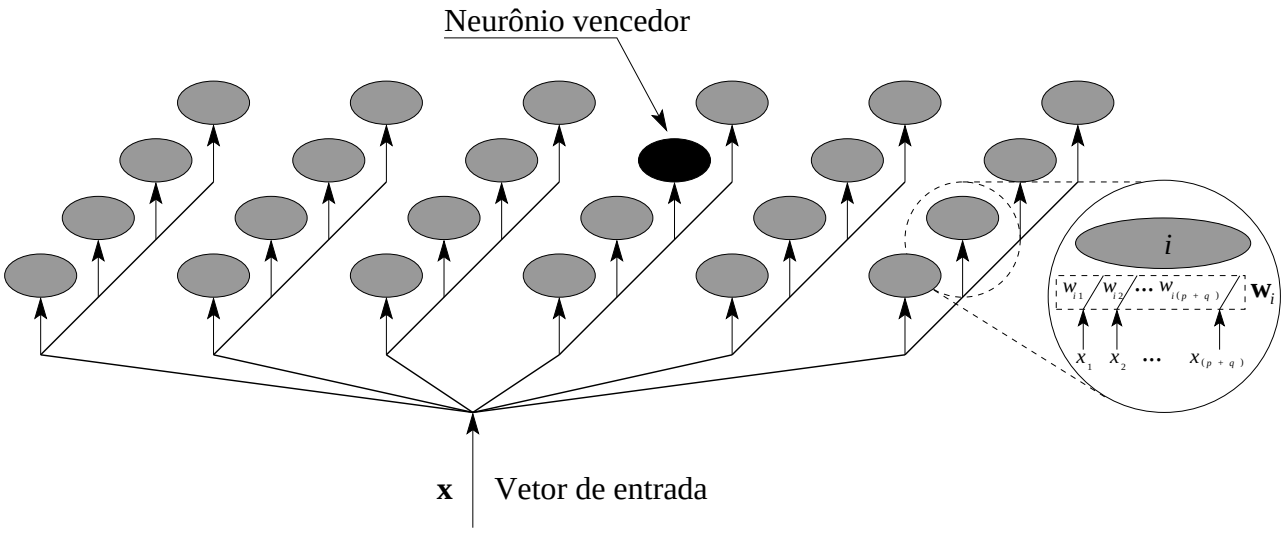


Figura 3.2 Rede SOM bidimensional. Cada conexão entre o vetor de entrada e os neurônios tem dimensão $p + q$, assim como o vetor de pesos $\mathbf{w}_i$ associado a cada neurônio $i, i = 1, 2, \ldots, g$.

Dado um vetor de entrada $\mathbf{x} \in \mathcal{X}$, o algoritmo SOM primeiro identifica um determinado neurônio $i^*(\mathbf{x})$ no espaço de saída $\mathcal{A}$ em concordância com o mapeamento característico Φ. O vetor de pesos $\mathbf{w}_{i^*}$ associado ao neurônio i^* pode então ser visto como um *ponteiro* para o espaço de entrada $\mathcal{X}$.

O algoritmo responsável pela formação do mapa auto-organizável tem como passo inicial a atribuição de valores iniciais aos pesos da rede. Assim procedendo, nenhum ordenamento inicial é imposto aos neurônios. Uma vez que os valores iniciais para os parâmetros da rede tenham sido devidamente escolhidos, três processos estão envolvidos na formação do mapa auto-organizável:

- *Competição.* Para cada vetor de entrada, os neurônios na rede calculam seus respectivos valores para uma certa função discriminante. Esta função provê a base para uma competição entre os neurônios visando escolher aquele que melhor represente o vetor de entrada atual. O neurônio com menor valor para a função discriminante é declarado o *vencedor* da competição.

- *Cooperação.* O neurônio vencedor e seus vizinhos no mapa discreto interagem através de uma *função vizinhança.* Essa interação é tanto mais positiva quanto mais próxima um determinado neurônio estiver do neurônio vencedor.

- *Adaptação.* Os dois processos (competição e cooperação) descritos nos itens anteriores atuam conjuntamente durante o ajuste dos pesos sinápticos da rede. O neurônio vencedor fica com a maior parcela do ajuste, mas seus vizinhos no mapa também têm

seus pesos ajustados de acordo com a sua distância ao neurônio vencedor. Quanto mais próximo do vencedor, maior o ajuste.

Uma análise detalhada do processo de competição, cooperação e adaptação é feita nas próximas subseções. Em seguida, são apresentadas outras duas subseções onde se comenta sobre o critério de convergência da rede SOM e, também, sobre a preservação de topologia feita por esta rede.

3.6.1 Competição: O Papel da Distância Euclidiana

Para encontrar o neurônio vencedor, o vetor de pesos de todos os neurônios da rede é comparado com o vetor de entrada. Essa comparação é, em geral, uma medida da distância entre cada um dos vetores de pesos e o vetor de entrada. O neurônio cujo vetor de pesos está mais próximo do vetor de entrada é considerado o *vencedor*. Matematicamente, a competição pode ser implementada em termos da distância euclidiana (aqui repetida da Eq. (3.8)) da seguinte forma

$$i^*(\mathbf{x}(t)) = \arg\min_{i \in \mathcal{A}} \|\mathbf{x}(t) - \mathbf{w}_i(t)\|, \tag{3.15}$$

em que $i^*(\mathbf{x}(t))$ é o índice que representa o neurônio vencedor para o padrão de entrada $\mathbf{x}(t)$. A norma euclidiana $\|\cdot\|$ é definida como

$$\|\mathbf{x}(t) - \mathbf{w}_i(t)\| = \sqrt{[\mathbf{x}(t) - \mathbf{w}_i(t)]^T[\mathbf{x}(t) - \mathbf{w}_i(t)]} = \sqrt{\sum_{j=1}^{p+q}[x_j(t) - w_{ij}(t)]^2}, \tag{3.16}$$

A Equação (3.15) permite a seguinte observação:

> *O espaço contínuo a que pertencem os vetores de entrada é mapeado em um espaço de saída discreto por meio de um processo de competição entre os neurônios.*

3.6.2 Cooperação: O Papel da Função de Vizinhança

O neurônio vencedor $i^*(t)$ determina o centro de um grupo espacialmente localizado de neurônios no mapa: a vizinhança de i^*. O neurônio vencedor e sua vizinhança interagem lateralmente de forma cooperativa. Existe evidência neurobiológica para esta *interação lateral* entre os neurônios excitados (KOHONEN, 2001). Em particular, um neurônio que está disparando tende a excitar neurônios em sua vizinhança imediata *mais* do que aqueles

mais distantes. A intensidade da interação lateral entre o vencedor i^* e um neurônio i qualquer é, em geral, descrita matematicamente na forma de uma função vizinhança $h(i^*, i; t)$. Esta função define o que se chama de vizinhança *topológica* centrada no neurônio vencedor $i^*(\mathbf{x}(t))$.

Considerando o estudo do arranjo discreto assumido pelos neurônios na rede SOM, têm-se dois modelos bastante utilizados (KOHONEN, 2001): (i) a grade unidimensional e, (ii) a grade bidimensional. Analisando o caso de que os neurônios estejam dispostos em uma grade unidimensional, tem-se que $\mathbf{r}_i(t) \in \mathbb{R}$, ou seja, a posição de um neurônio i qualquer coincide com seu próprio índice, ou seja, $\mathbf{r}_i(t) = i$. Neste caso, cada neurônio possui apenas vizinhos à direita e à esquerda (ver Figura 3.1). Contudo, se os neurônios da rede SOM estão dispostos em uma grade bidimensional, tem-se que $\mathbf{r}_i(t) \in \mathbb{R}^2$, ou seja, a posição de um neurônio i na grade é dada pelas coordenadas (x_i, y_i) em relação a uma origem pré-fixada. Neste caso, um neurônio pode ter vizinhos à esquerda, à direita, acima, abaixo e diagonalmente (ver Figura 3.2).

Seja $\mathbf{r}_i$ a localização (coordenadas) do neurônio i em um arranjo uni-dimensional, por exemplo. Então, assume-se que $h(i^*, i; t)$ é uma função unimodal da distância lateral $\|\mathbf{r}_{i^*} - \mathbf{r}_i\|$ entre o neurônio vencedor i^* e seu vizinho i no mapa. A função vizinhança deve ainda satisfazer os seguintes requisitos:

- A função vizinhança $h(i^*, i; t)$ alcança seu valor máximo para o neurônio vencedor i^* para o qual a distância lateral $\|\mathbf{r}_{i^*} - \mathbf{r}_i\|$ é nula;
- A função vizinhança $h(i^*, i; t)$ é simétrica em relação ao neurônio vencedor;
- A amplitude de $h(i^*, i; t)$ decai monotonicamente com o aumento da distância lateral $\|\mathbf{r}_{i^*} - \mathbf{r}_i\|$, ou seja:

$$\text{Se} \quad \|\mathbf{r}_{i^*} - \mathbf{r}_i\| \longrightarrow \infty \quad \Longrightarrow \quad h(i^*, i; t) \longrightarrow 0. \tag{3.17}$$

A Equação (3.17) é uma condição necessária para convergência.

Uma escolha comum para $h(i^*, i; t)$ que satisfaz os requisitos anteriores é a função Gaussiana, ou seja,

$$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_{i^*} - \mathbf{r}_i\|^2}{2\sigma^2(t)}\right), \tag{3.18}$$

sendo que o parâmetro $\sigma(t)$ define a "largura efetiva" da vizinhança topológica, i.e., ele define como os demais neurônios participam no processo de aprendizagem juntamente com o neurônio vencedor no instante atual.

É importante ressaltar que a forma original do algoritmo SOM descrita em Kohonen (1982) utiliza uma função vizinhança de amplitude constante (retangular):

$$h(i^*, i; t) = \begin{cases} 1, \text{ se } i \in K_{i^*}(t) \\ 0, \text{ caso contrário,} \end{cases} \tag{3.19}$$

em que $K_{i^*}(t)$, chamado de *conjunto vizinhança*, contém os neurônios vizinhos de $i^*(t)$. A função de vizinhança retangular é adotada por este trabalho pois ela apresenta um custo computacional consideravelmente menor do que a Gaussiana (KOHONEN, 1998).

Tanto para a função vizinhança Gaussiana quanto para a retangular, a largura da vizinhança topológica decresce monotonicamente com o passar do processo de aprendizagem. A diminuição no número de vizinhos é de fundamental importância para o ordenamento e convergência da rede. Maiores detalhes sobre como proceder para diminuir a largura da vizinhança com o tempo para a função retangular será discutido posteriormente.

3.6.3 Adaptação: Ajuste dos Pesos

O ajuste dos pesos sinápticos é o último passo na formação de um mapa auto-organizável. A questão que se coloca aqui é como realizar a alteração dos pesos. Como a rede SOM é não-supervisionada, o vetor de pesos de um dado neurônio i deve ser modificado como função do estímulo de entrada apenas. Conforme dito anteriormente, os processos de competição e cooperação atuam conjuntamente durante a adaptação sináptica com o objetivo de extrair algum tipo de regularidade presente no espaço de entrada $\mathcal{X}$. A regra de aprendizagem para o algoritmo SOM baseia-se em suposições feitas por Hebb (1949), em uma tentativa de relacionar a alteração estrutural de sinapses reais com a memória e, consequentemente, com aprendizagem ou experiência. Assim, uma abstração matemática destas suposições foi proposta por Kohonen como uma regra recursiva para ajuste dos pesos (KOHONEN, 1998):

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) h(i^*, i; t)[\mathbf{x}(t) - \mathbf{w}_i(t)], \tag{3.20}$$

em que o parâmetro $0 < \alpha(t) < 1$, chamado de *taxa de aprendizagem*, controla a intensidade com que os pesos sinápticos são modificados. Assim como no caso da largura da vizinhança topológica, a taxa de aprendizagem pode diminuir com o transcorrer do treinamento de modo a garantir convergência e, principalmente, estabilidade do mapa[1],

[1]Neste contexto, estabilidade se refere à manutenção de memória previamente aprendida quando novos dados são apresentados à rede SOM.

sendo que neste trabalho é adotado uma taxa de aprendizagem variável calculada pela Equação (3.10).

3.6.4 Ordenamento e Convergência

A partir de uma configuração inicial em que os pesos guardam valores atribuídos aleatoriamente, o algoritmo SOM modifica maximamente os valores dos pesos em direção a uma representação que reflita a estrutura (topologia) do espaço de entrada. Para que uma configuração "organizada" e estável do mapa seja atingida, faz-se necessária uma seleção criteriosa dos parâmetros $\alpha(t)$ e $\sigma(t)$. Quando uma configuração organizada final é alcançada, diz-se que o algoritmo convergiu ou atingiu um estado final.

O ponto de importância capital para o processo de convergência da rede é a diminuição da largura da vizinhança topológica. Para o caso de uma função vizinhança Gaussiana, esta largura é refletida no valor do parâmetro $\sigma(t)$. A largura deve ser inicialmente alta para promover um rápido ordenamento dos pesos e decrescer de modo a garantir convergência dos mesmos. Uma escolha comum para a dependência temporal de $\sigma(t)$ tem a forma apresentada a seguir:

$$\sigma(t) = \sigma_0 \cdot \left(\frac{\sigma_f}{\sigma_0}\right)^{\frac{t}{\tau_2}}, \quad t = 0, 1, ..., \tau_2, \tag{3.21}$$

em que τ_2 é uma outra constante de tempo desta vez associada ao decaimento do parâmetro σ. Uma maneira útil de entender a influência da função vizinhança $h(i^*, i; t)$ com o passar do tempo é a seguinte. O propósito de uma largura inicial grande para $h(i^*, i; t)$ é *correlacionar* as direções de ajuste dos pesos de um grande número de neurônios da rede. À medida que a largura de $h(i^*, i; t)$ decresce, também decresce o número de neurônios cujas direções de ajuste estão correlacionadas.

Como foi explicado acima, no processo de convergência para o caso da função vizinhança tida como Gaussiana, repete-se este mesmo procedimento para o caso retangular e segue que, para esta tese, a preferência no uso da vizinhança do tipo retangular sobre a do tipo Gaussiana deve-se ao menor custo computacional da primeira. Então, no decorrer da atualização dos pesos sinápticos o valor de $h(i^*, i; t)$ mantém-se constante e igual a um, fazendo uma atualização igualitária entre os neurônios vizinhos ao vencedor, i, e o próprio vencedor, $i^*(t)$, com a diminuição da largura da vizinhança topológica acontecendo gradativamente.

3.6.5 Preservação de Topologia

Pode-se expressar a propriedade de preservação de topologia da rede SOM da seguinte forma (HERTZ et al., 1991): sejam $\mathbf{x}_1$ e $\mathbf{x}_2$ dois vetores no espaço de entrada $\mathcal{X}$, $\mathbf{r}_{i_1^*}$ e $\mathbf{r}_{i_2^*}$ as coordenadas dos neurônios vencedores para $\mathbf{x}_1$ e $\mathbf{x}_2$, respectivamente. Diz-se que a rede SOM, corretamente treinada, preserva a topologia do espaço de entrada se a seguinte relação for observada

$$\|\mathbf{x}_1 - \mathbf{x}_2\| \longrightarrow 0 \quad \Longrightarrow \quad \|\mathbf{r}_{i_1^*} - \mathbf{r}_{i_2^*}\| \longrightarrow 0, \tag{3.22}$$

ou seja, se quaisquer dois vetores estão fisicamente próximos no espaço de entrada, então eles terão neurônios vencedores espacialmente próximos na rede. Dá-se a essa característica, o nome de *Propriedade de Preservação de Topologia*. Com base nesta propriedade, pode-se fazer a seguinte afirmação sobre o algoritmo SOM:

> *O espaço contínuo $\mathcal{X}$ é mapeado em um espaço discreto de saída $\mathcal{A}$ por um processo de competição-cooperação entre as unidades da rede, de forma tal que a sua topologia é preservada.*

Devido à propriedade de preservação de topologia, a rede SOM é capaz de construir uma **aproximação do espaço de entrada**, ou seja, ela constrói uma aproximação discreta do espaço de entrada, na qual cada neurônio da rede representa uma determinada região do espaço de entrada que define sua **região de atração** ou **campo receptivo**. Esta região é conhecida também como **célula de Voronoi**. Assim, uma das principais aplicações da rede SOM é a categorização de dados não-rotulados em agrupamentos (clusters) e sua posterior utilização na classificação de vetores de características que não estavam presentes durante o treinamento.

Com o uso das propriedades da rede SOM, tais como competição e cooperação, esta rede neural é capaz de implementar uma projeção Φ (ver Eq. (3.14)) que preserve relações de proximidade espacial entre os dados de entrada, ou seja, o mapeamento preserva a topologia do espaço de entrada no espaço de saída (HAYKIN, 2008), conforme ilustrado na Figura 3.1, na qual $dim(\mathcal{X}) = 2$ e $dim(\mathcal{A}) = 1$, e os pontos pretos correspondem às coordenadas dos vetores de pesos do i-ésimo neurônio. Neurônios que são vizinhos na grade unidimensional são conectados por linhas tracejadas.

A Figura 3.1 demonstra a propriedade de preservação da topologia vista na rede SOM para o caso da grade ser unidimensional (1D) e o espaço de entrada ser bidimensional (2D). De uma forma didática, esta propriedade pode também ser demonstrada usando

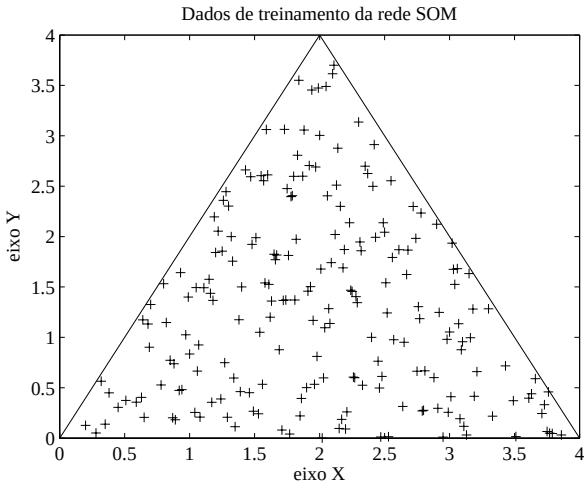


Figura 3.3 Conjunto de dados 2D utilizados para o treinamento da rede SOM.

um conjunto de dados artificial 2D para o treinamento da rede, em que serão modificados os tipos de grade e o número de neurônios usados pelo algoritmo SOM. Estes dados estão dispostos em uma estrutura de formato triangular como visto na Figura 3.3.

Na Figura 3.6.5, tem-se a representação de duas arquiteturas da rede SOM aplicadas ao conjunto de dados em questão, uma com grade unidimensional (1×36) e outra com grade bidimensional (6×6), em que ($x_g \times y_g$) corresponde a disposição dos neurônios distribuídos na grade, através dos eixos X e Y. Os vetores de pesos são inicializados aleatoriamente como visto nas Figs. 3.4(a) e (d). A convergência total da rede para os dois modelos é demonstrada nas Figs. 3.4(b) e (e), em que os pesos sinápticos são definidos por meio dos pontos pretos, e os mesmos estão interligados entre si através de linhas, representando a sua vizinhança topológica. A topologia dos dados de entrada pode ser visualizada através da disposição dos pesos sinápticos no espaço dos mesmos, após a convergência das redes SOM 1D e 2D, como visto nas Figs. 3.4(c) e (f). Os resultados fornecidos pela rede SOM foram obtidos após 30.000 iterações (épocas).

Uma outra situação pode ser vista usando 50 neurônios para o treinamento da rede SOM. As arquiteturas utilizadas na simulação possuem uma grade 1D (1×50), e outra grade 2D (10×5). Os vetores de pesos são inicializados aleatoriamente como visto nas Figs. 3.5(a) e (d). A convergência total da rede para os dois modelos é demonstrada nas Figs. 3.5(b) e (e), em que os pesos sinápticos são definidos por meio dos pontos pretos, e os mesmos estão interligados entre si através de linhas, representando a sua vizinhança topológica. A topologia dos dados de entrada pode ser visualizada através da disposição dos pesos sinápticos no espaço dos mesmos, após a convergência das redes SOM 1D e 2D, como visto nas Figs. 3.5(c) e (f). A mesma quantidade de épocas foram utilizadas nestas

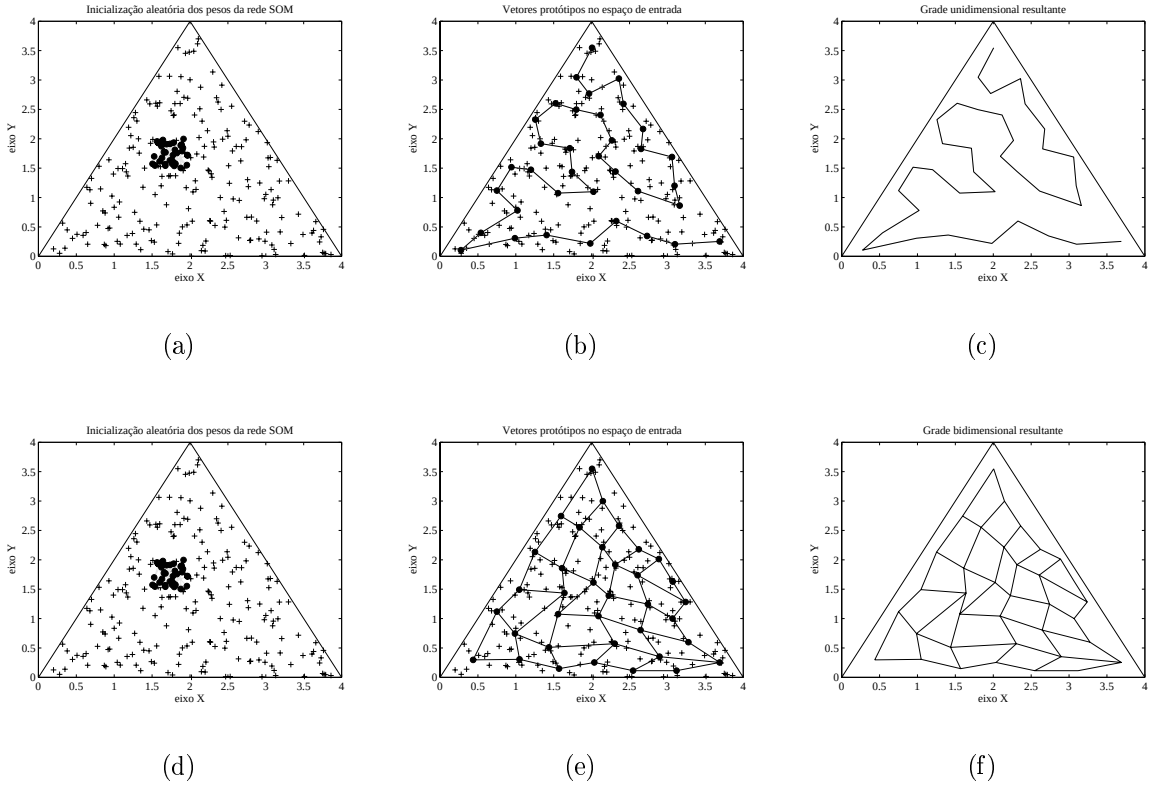


Figura 3.4 Mapeamento 2D: Rede SOM com grade uni- e bidimensional usando 36 neurônios. (a-d) Inicialização dos pesos, (b-e) Convergência da rede, e (c-f) Grades unidimensional (1×36) e bidimensional (6×6) resultantes.

simulações.

A motivação em realizar estes testes é poder verificar a convergência da rede SOM do tipo uni- e bidimensional, variando então a quantidade e a forma de disposição dos neurônios pertencentes as mesmas. No primeiro caso, foram utilizados 36 neurônios nas grades 1D e 2D. Para grade 2D, estes neurônios foram dispostos em forma de quadrado, 6×6. Já no segundo caso, o número de neurônios utilizados foi 50 e a disposição dos mesmos na grade 2D foi 10×5, no formato retangular. Pelos resultados obtidos, constata-se que a melhor convergência atingida foi para a rede SOM 2D com 36 neurônios, com base na forma assumida pela rede, seguindo a disposição apresentada pelos dados de entrada.

3.7 Algoritmos Fuzzy

Anteriormente foram descritos algoritmos de aprendizagem competitiva em que o vencedor é o único a ser atualizado a cada iteração, tal como a rede WTA. Este tipo de aprendizagem competitiva é denominada de competição dura (*hard competition*). Em

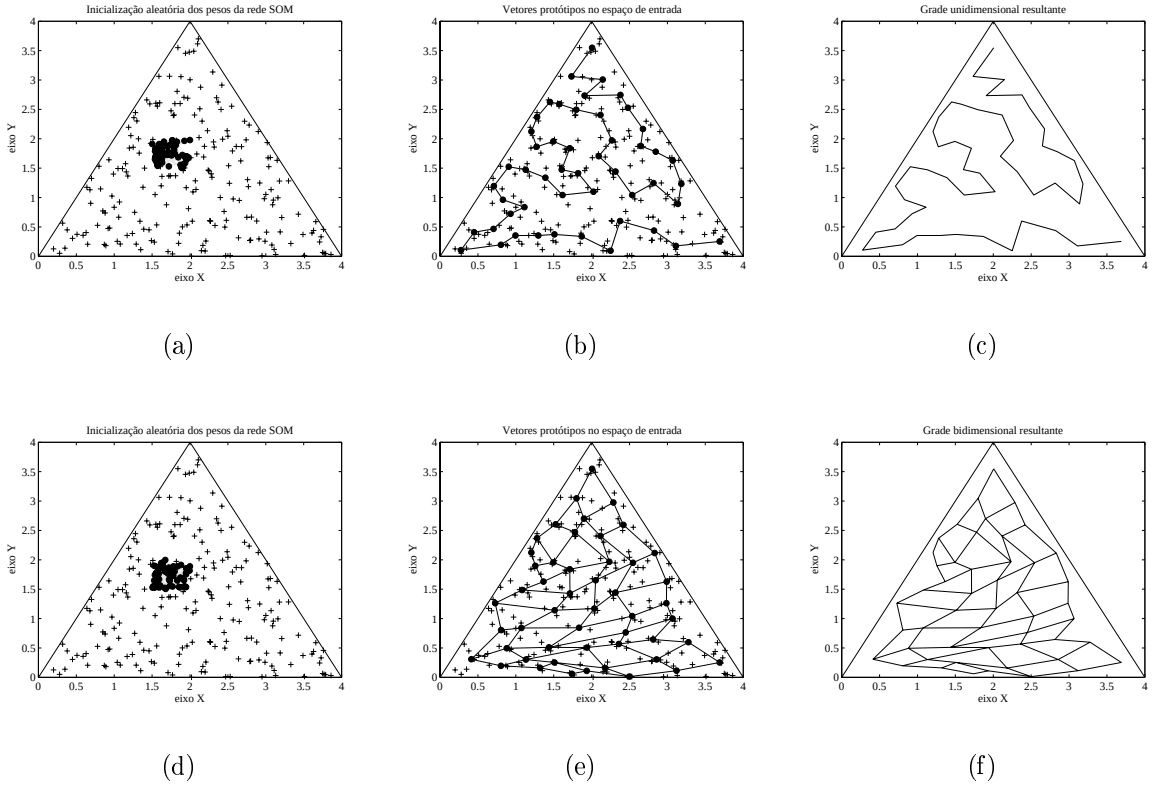


Figura 3.5 Mapeamento 2D: Rede SOM com grade uni- e bidimensional usando 50 neurônios. (a-d) Inicialização dos pesos, (b-e) Convergência da rede, e (c-f) Grades unidimensional (1×50) e bidimensional (10×5) resultantes.

outros algoritmos, como a rede SOM, não só o neurônio vencedor tem seu protótipo atualizado, mas também os protótipos de seus vizinhos físicos são modificados. Este tipo de aprendizagem competitiva é denominada de competição suave (*soft competition*). Contudo, qualquer que seja o tipo de competição, um vetor de atributos só pode pertencer a um único agrupamento de dados, que é aquele representado pelo protótipo mais próximo.

Já no caso das técnicas de quantização vetorial fuzzy (*fuzzy vector quantization*) (MASSULLI; ROVETTA, 2006; KARAYIANNIS; PAI, 1994), os dados podem possuir características que permitam que eles sejam mapeados em diversos vetores-protótipos com uma intensidade controlada por uma função de pertinência (ZADEH, 1965).

Seja $\mu_i(\mathbf{x})$ uma função que determina o grau de pertinência de um vetor $\mathbf{x}$ à célula de Voronoi representada pelo protótipo $\mathbf{w}_i$. As seguintes propriedades devem ser obedecidas por $\mu_i(\mathbf{x})$ para uma quantidade G de conjuntos fuzzy:

$$\mu_i(\mathbf{x}) \in [0,1] \quad \text{e} \quad \sum_{i=1}^{G} \mu_i(\mathbf{x}) = 1. \tag{3.23}$$

A incorporação da função de pertinência ao processo de quantização do espaço de entrada retarda a decisão sobre a qual célula de Voronoi cada vetor de entrada pertencerá até o fim do treinamento. A seguir são apresentados dois algoritmos de quantização vetorial fuzzy, um de treinamento ao estilo *batch* e o outro de treinamento sequencial.

3.7.1 Algoritmo Fuzzy *K*-Médias (*Batch*)

A função objetivo associada ao algoritmo K-médias *batch*, mostrada na Equação (3.4), pode ser estendida para uma versão fuzzy proposta por Dunn (1973):

$$D = \frac{1}{N_1} \sum_{\mathbf{x} \in V_i} \sum_{i=1}^{G} \|\mathbf{x} - \mathbf{w}_i\|^2 [\mu_i(\mathbf{x})]^z. \tag{3.24}$$

em que o expoente z representa o grau de nebulosidade da função. Usualmente atribui-se a ele um valor um pouco maior que 1. Se $z = 0$, recai-se no algoritmo K-médias *batch* clássico. Note que o somatório é feito a cada época, para todos os protótipos.

A atualização dos pesos usa a função de pertinência fuzzy. Para isso, é preciso calcular o grau de pertinência de um certo vetor a cada um dos agrupamentos, ou seja

$$\mu_i(\mathbf{x}) = \left(\sum_{j=1}^{G} \left(\frac{\|\mathbf{x} - \mathbf{w}_i\|^2}{\|\mathbf{x} - \mathbf{w}_j\|^2} \right)^{1/(z-1)} \right)^{-1}. \tag{3.25}$$

A minimização da Equação (3.24) leva à seguinte regra fuzzy de atualização dos protótipos:

$$\mathbf{w}_i = \frac{\sum_{\mathbf{x} \in V_i} [\mu_i(\mathbf{x})]^z \mathbf{x}}{\sum_{\forall \mu_i} [\mu_i(\mathbf{x})]^z}. \tag{3.26}$$

É importante notar que a regra da Equação (3.26) pode ser vista simplesmente como uma generalização para o caso de agrupamentos fuzzy daquela mostrada na Equação (3.5). Os critérios de inicialização dos protótipos do algoritmo K-médias *batch* nebuloso e seu critério de parada são os mesmos de sua versão não-nebulosa.

3.7.2 Rede FCL

O algoritmo descrito a seguir, denominado *Fuzzy Competitive Learning* (FCL) (CHUNG; LEE, 1994), é uma variante nebulosa do algoritmo WTA, com a diferença de que todos os protótipos são ajustados a cada iteração, em vez de um só como na rede WTA. A

intensidade do ajuste é regulada justamente pela função de pertinência fuzzy. De forma geral, o ajuste dos pesos é dado pela seguinte expressão:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t)[\mu_i(\mathbf{x}(t))]^z[\mathbf{x}(t) - \mathbf{w}_i(t)], \quad \forall i = 1, ..., g. \tag{3.27}$$

É importante destacar que, como os protótipos são atualizados a cada iteração t, os valores de $\mu_i(\mathbf{x}(t))$ devem ser atualizados também a cada apresentação de um novo vetor de entrada $\mathbf{x}$. O passo de aprendizagem $\alpha(t)$ apresenta valor variável e é calculado com base na Equação (3.10).

Outro aspecto importante a destacar é a semelhança da regra de aprendizagem da Eq. (3.27) com a regra de aprendizagem da rede SOM. É devido a esta semelhança que alguns autores (BARALDI; BLONDA, 1999) afirmam que algoritmos neurais baseados em competição suave são funcionalmente equivalentes a algoritmos de análise de agrupamentos nebulosos.

3.8 Resumo

Neste capítulo foram descritos os algoritmos de quantização vetorial, neurais, fuzzy e estatísticos, que serão usados no projeto de modelos lineares locais para identificação inversa de sistemas dinâmicos. O conhecimento adquirido neste capítulo servirá como insumo teórico aos capítulos seguintes, principalmente quando for analisada a influência do algoritmo de quantização vetorial no desempenho dos modelos lineares locais propostos.

4 *Modelos Lineares Locais para Identificação de Sistemas: Primeiras Tentativas*

Esse capítulo descreve as primeiras tentativas no âmbito desta pesquisa de se utilizar algoritmos de quantização vetorial para identificação inversa de sistemas dinâmicos. Estas primeiras tentativas baseiam-se na rede SOM no papel de quantizador vetorial, mas também podem ser usados quaisquer dos algoritmos de quantização vetorial descritos no Capítulo 3. Isto inclusive será avaliado posteriormente nas simulações computacionais a serem apresentadas no Capítulo 6.

Vale lembrar que a idéia básica por trás do uso de quantizadores vetoriais no projeto de modelos locais é a de particionar o espaço contínuo de entrada em regiões não-sobrepostas, as chamadas células de Voronoi, em que os centróides correspondem aos vetores-protótipos do algoritmo utilizado. Em seguida, a fim de estimar a saída do sistema, um hiperplano deve ser associado a cada célula de Voronoi ou para um pequeno conjunto delas.

4.1 Introdução

Os algoritmos a serem apresentados neste capítulo são vistos como um mecanismo para desenvolver modelos locais pois particionam os espaços de entrada e de saída em regiões específicas, que são caracterizadas pelo uso de modelos lineares para estimar a saída de um sistema associada à cada partição. Tais modelos lineares atuam conjuntamente com os vetores-protótipos da rede SOM, estes responsáveis em mapear os vetores de entrada ao modelo local adequado. Cada método a ser descrito neste capítulo visa aproximar a função entrada-saída de um sistema SISO (*Single-Input, Single-Output*).

O primeiro modelo local a ser apresentado, proposto na literatura no inicio da década

de 1990 (WALTER et al., 1990), possui um modelo linear associado a cada neurônio da rede SOM, sendo que sua aplicação em identificação de sistemas foi feita apenas recentemente no contexto da presente tese de doutorado. O segundo modelo local foi desenvolvido como uma extensão da rede SOM para problemas de identificação de sistemas dinâmicos na tese de doutorado de Barreto et al. (2003a). Finalmente, o terceiro modelo é uma extensão do modelo proposto por Barreto et al. (2004), originalmente aplicado em predição de séries temporais não-estacionárias, ao problema de identificação de sistemas dinâmicos.

4.2 Mapeamento Linear Local

O primeiro modelo a ser descrito é denominado **Mapeamento Linear Local** (*Local Linear Mapping*, LLM) (WALTER et al., 1990). Grosso modo, a idéia básica do modelo LLM é associar cada neurônio na rede SOM a um modelo linear ARX[1] treinado pelo algoritmo LMS (*least-mean squares*). A rede SOM é usada para quantizar o espaço de entrada em um número reduzido de protótipos (e, assim, de células de Voronoi), enquanto o preditor linear associado a cada neurônio fornece uma estimativa local da saída do mapeamento a ser aproximado.

De maneira mais formal, seja $p+q$ a dimensão do espaço (contínuo) de entrada $\mathcal{X}$ sobre o qual incidirá uma operação de quantização vetorial. Um elemento qualquer deste espaço é definido como o vetor de entrada $\mathbf{x}(t) \in \mathcal{X} \subset \mathbb{R}^{p+q}$, também chamado de vetor de regressores.

Para a tarefa de identificação inversa de interesse para esta tese, cada vetor de entrada $\mathbf{x}(t) \in \mathbb{R}^{p+q}$ é definido como

$$\mathbf{x}(t) = \begin{pmatrix} y(t-1) \\ y(t-2) \\ \vdots \\ y(t-p) \\ u(t-1) \\ \vdots \\ u(t-q) \end{pmatrix} \tag{4.1}$$

Para realizar a quantização vetorial do espaço $\mathcal{X}$, cada neurônio i de uma rede SOM

[1] Acrônimo de *AutoRegressive model with eXogenous inputs*.

possui um vetor de pesos $\mathbf{w}_i$, definido como

$$\mathbf{w}_i = \begin{bmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{i(p+q)} \end{bmatrix}, \qquad i = 1, 2, \ldots, g, \tag{4.2}$$

na qual g é o número total de neurônios da rede. Associado a cada vetor de pesos existe um vetor de coeficientes $\mathbf{a}_i \in \mathcal{X} \subset \mathbb{R}^{p+q}$ contendo os coeficientes do i-ésimo modelo ARX, dado por

$$\mathbf{a}_i = \begin{bmatrix} a_{i1} \\ \vdots \\ a_{ip} \\ b_{i1} \\ \vdots \\ b_{iq} \end{bmatrix}, \tag{4.3}$$

na qual $p+q$ é a ordem do modelo ARX correspondente. Assim, os parâmetros ajustáveis do modelo LLM são os conjuntos de vetores de pesos $\mathbf{w}_i$ e seus respectivos vetores de coeficientes $\mathbf{a}_i$, para $i = 1, \ldots, g$.

Seguindo a lógica competitiva da rede SOM, somente um neurônio por vez poderá ser usado para estimar a saída do modelo LLM. A seleção do neurônio vencedor se dá com base na distância euclidiana do seu vetor de pesos ao vetor de entrada, ou seja,

$$i^*(t) = \arg\min_{i \in \mathcal{A}} \|\mathbf{x}(t) - \mathbf{w}_i(t)\|. \tag{4.4}$$

Uma vez determinado o neurônio vencedor, a saída do modelo LLM é calculada da seguinte maneira:

$$\begin{aligned} \hat{u}(t) &= \sum_{l=1}^{p} a_{i^*,l}(t) y(t-l) + \sum_{k=1}^{q} b_{i^*,k}(t) u(t-k) \\ &= \mathbf{a}_{i^*}^T(t) \mathbf{x}(t). \end{aligned} \tag{4.5}$$

Resta, portanto, descrever o processo de aprendizagem do modelo LLM. Como se poderia esperar, o processo de adaptação dos vetores-protótipos e dos vetores de coeficientes associados a todos os neurônios segue a filosofia *cooperativa* da rede SOM, em que não apenas os parâmetros do neurônio vencedor, mas também os parâmetros dos neurônios vizinhos, são ajustados a cada iteração do algoritmo. Esta filosofia é levada a cabo por

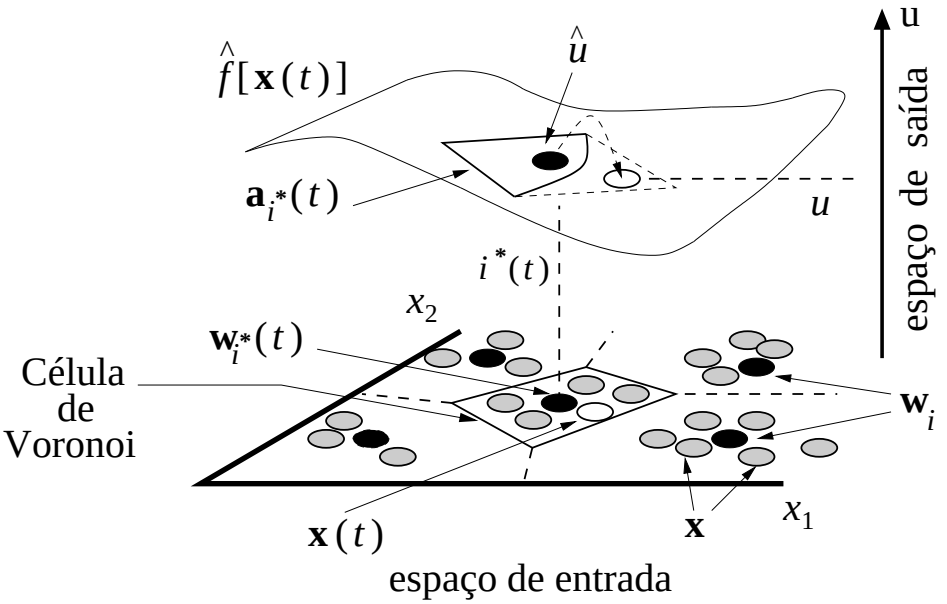


Figura 4.1 Representação da arquitetura do modelo LLM.

meio das seguintes regras de aprendizagem:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t)h(i^*, i; t)[\mathbf{x}(t) - \mathbf{w}_i(t)] \tag{4.6}$$

$$\mathbf{a}_i(t+1) = \mathbf{a}_i(t) + \alpha' h(i^*, i; t)\Delta\mathbf{a}_i, \tag{4.7}$$

na qual as constantes $\alpha(t)$ e α' definem as taxas de aprendizagem dos vetores-protótipos e dos vetores de coeficientes dos neurônios, respectivamente, e $h(i^*, i; t)$ é a função vizinhança (ver Equações (3.18) e (3.19)) definida entre o neurônio vencedor i^* e os demais neurônios i na sua vizinhança topológica.

Para completar a descrição da Equação (4.7), tem-se ainda que especificar o termo de ajuste dos coeficientes $\Delta\mathbf{a}_i$. Ele é escolhido para minimizar o erro de estimação quadrático definido por

$$\sum_t \left[u(t) - \mathbf{a}_{i^*}^T(t)\mathbf{x}(t)\right]^2 = \sum_t \left[u(t) - \hat{u}(t)\right]^2, \tag{4.8}$$

e é dado pela regra de Widrow-Hoff (também conhecida como regra LMS ou regra delta), representada por

$$\Delta\mathbf{a}_i(t) = \left[u(t) - \mathbf{a}_i^T(t)\mathbf{x}(t)\right] \frac{\mathbf{x}(t)}{\|\mathbf{x}(t)\|^2}, \tag{4.9}$$

sendo $u(t)$ a saída observada do mapeamento que se deseja aproximar.

Após a apresentação do algoritmo é interessante esclarecer de uma forma didática o princípio de funcionamento da rede LLM. Com base na Figura 4.1, pode-se ver que inicialmente o algoritmo particiona o espaço de entrada dos dados por meio de regiões de atração (ou Células de Voronoi) associado a cada vetor protótipo $\mathbf{w}_i$ pertencente a rede

SOM. Cada região contém vetores de dados que são mapeados com mesma característica estatística.

Encerrado o treinamento, a cada novo vetor de entrada $\mathbf{x}(t)$ apresentado à rede, há o mapeamento do mesmo para a Célula de Voronoi pertencente ao vetor protótipo mais próximo $\mathbf{w}_{i^*}(t)$. Para cálculo da saída estimada $\hat{u}$, é utilizado o vetor de coeficientes $\mathbf{a}_{i^*}(t)$ associado ao vetor protótipo vencedor. Por meio da interpolação linear, busca-se então estimar a função $f[\cdot]$ geradora da relação entre os dados de entrada e saída.

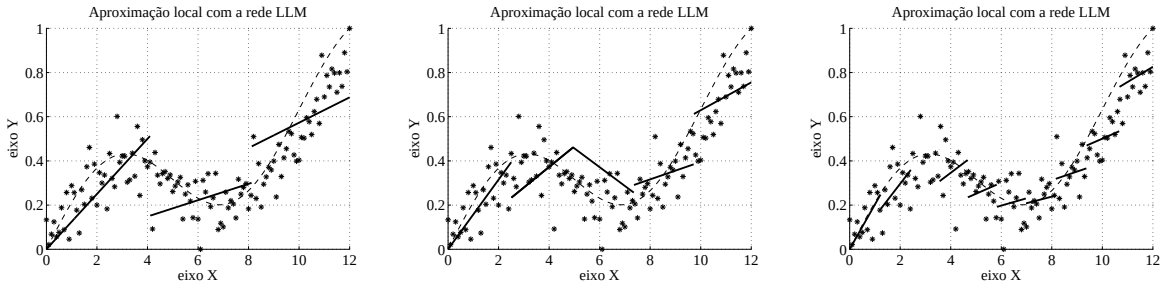
Uma vez apresentado o modelo LLM, é importante e construtivo tecer alguns comentários sobre as principais características deste algoritmo, que o diferenciam das abordagens lineares tradicionais.

- O modelo LLM pode ser entendido como um banco de modelos lineares do tipo ARX, um para cada neurônio, cujo o modelo local a ser usado no instante t é escolhido entre os g modelos locais disponíveis. A escolha de qual modelo local e, consequentemente, de qual vetor de coeficientes usar é definida pela Equação (4.4). Se a rede possuir apenas um neurônio (i.e. $g = 1$), o modelo LLM se torna equivalente ao modelo ARX/LMS convencional.

- Do ponto de vista computacional, pode-se dizer que o algoritmo LLM implementa uma *aproximação linear por região* da função não-linear que se deseja aproximar[2], ou seja, o sinal de entrada é vetorizado e o conjunto de vetores resultantes é quantizado pela rede SOM através de seus vetores-protótipos $\{\mathbf{w}_i\}_{i=1}^{g}$. Estes protótipos definem as coordenadas dos centróides de g regiões do espaço de entrada $\mathcal{X}$, chamadas de *células de Voronoi* (PRINCIPE et al., 2000) (ver Figura 4.1). Por sua vez, cada célula de Voronoi define o *campo receptivo* ou *região de atração* do i-ésimo neurônio, ou seja, a região do espaço de entrada para a qual o neurônio i é sempre escolhido vencedor. O que o modelo LLM faz é associar um vetor de coeficientes $\mathbf{a}_i$ a cada célula de Voronoi, tal que cada conjunto de coeficientes define um hiperplano aproximador da função não-linear de interesse.

A definição de modelagem local apresentada no início do Capítulo 1 pode ser demonstrada através do uso da rede LLM aplicada ao problema de aproximação de função. Como foi comentado no Capítulo 2, a função (vê Eq. (2.1)) que foi aproximada através do modelo global, tal como a rede MLP, será utilizada novamente para ilustrar o conceito desejado.

[2]Na presente tese, esta função corresponde a modelagem inversa de uma planta industrial.



(a) 3 neurônios (b) 5 neurônios (c) 10 neurônios

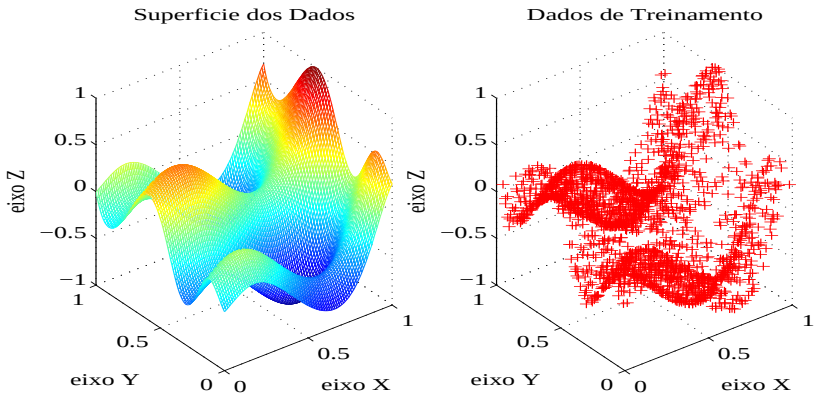
Figura 4.2 Aproximação local usando o modelo LLM para três configurações.

A rede LLM foi treinada com três diferentes quantidades de neurônios: 3, 5 e 10, com o intuito de analisar a aproximação realizada pela rede com base em cada configuração implementada. Foram utilizadas 2.000 épocas por etapa de treinamento das redes, e os dados empregados foram os mesmos da simulação com o modelo global MLP, com características ruidosas.

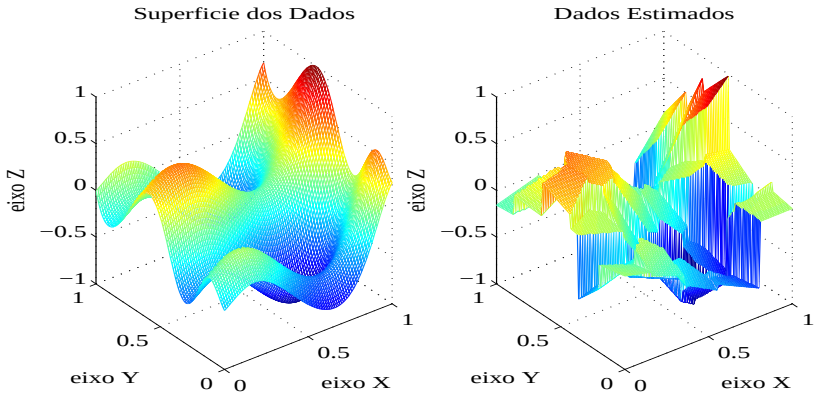
Através da Figura 4.2 pode-se ter uma ideia do tipo de aproximação linear local realizada pelo modelo LLM. Estas figuras ilustram a forma com que o modelo aproxima a função em questão, representada na figura pela linha tracejada. Conforme os vetores de protótipos dividem o espaço de entrada em regiões específicas, no caso os dados do eixo X, cada protótipo contém uma reta que busca estimar a saída em questão, que neste caso é o eixo Y.

São representadas diretamente nas figuras os aproximadores locais construídos sobre cada uma das regiões de atração definida pelo seu respectivo vetor de pesos (no caso, um escalar) no eixo X. Estas retas buscam representar a função por partes, destinando para cada percurso do gráfico, um aproximador linear específico.

Por exemplo, analisando a Figura 4.2(a), o que se tem são três retas distribuídas sobre trechos específicos da função. Verifica-se que uma reta corresponde aos valores do eixo X entre 0 e 4, outra reta corresponde ao intervalo entre 4 e 8, e por último, uma outra entre 8 e 12. A partir do segundo trecho que corresponde a uma descida suave no gráfico, a inclinação assumida pela reta não acompanha o mesmo, demonstrando um erro de aproximação. No último trecho, a reta assumiu a mesma inclinação do gráfico mas não o aproximou corretamente. Expandindo esta análise para os demais casos, constata-se a mesma dificuldade na aproximação local, demonstrando uma limitação no processo. Logo, as retas utilizadas em cada simulação não conseguiram compor o gráfico como o todo.



(a) Gráficos da superfície original dos dados e do conjunto de treinamento da rede LLM.



(b) Comparação entre a superfície original dos dados e os resultados obtidos pela rede LLM.

Figura 4.3 Rede LLM. Aproximação de uma superfície tridimensional.

Com base na análise feita a partir da aproximação local de uma função no espaço bidimensional, pode-se verificar o uso da rede LLM para uma aproximação no espaço tridimensional. A função utilizada é a mesma que foi definida na Equação (2.2), onde o modelo global MLP foi empregado para aproximá-la. Para o treinamento da rede LLM foram utilizadas 5000 épocas, sendo o número de neurônios empregados na simulação igual a 30.

Na Figura 4.3(a), apresenta-se a superfície real dos dados e ao lado desta figura tem-se o conjunto de dados de treinamento utilizado pela rede LLM. Estes dados foram gerados a partir da função apresentada na Equação (2.2), tendo seus valores acrescidos de ruído branco Gaussiano. Os dados de entrada e saída foram normalizados no intervalo de [0,1].

Tabela 4.1 Algoritmo de modelagem local LLM.

Modelo LLM	
Constantes	**Valores típicos**
α : Taxa de aprendizagem	$0.001 < \alpha < 0.5$
σ : Abertura da vizinhança topológica	$0.001 < \sigma < (g/2)$
α' : Taxa de aprendizagem do algoritmo LMS	$\alpha' = 0.01$
Entradas	
$\mathbf{x}(t)$: vetor de regressores, dimensão $(p+q) \times 1$	$u(t)$: variável observada
Algoritmo	
1. Inicialização $(t = 0)$	
$\mathbf{w}_i, \mathbf{a}_i \sim U(0,1)$, $i = 1, \ldots, g$	
2. Treinamento do modelo $(t = 1, 2, \ldots, N_1)$	
2.1 Escolha do neurônio vencedor:	
$i^*(t) = \arg\min_i \|\mathbf{x}(t) - \mathbf{w}_i(t)\|$.	
2.2 Atualização dos vetores de pesos e de coeficientes:	
$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_{i^*} - \mathbf{r}_i\|^2}{2\sigma^2}\right)$,	
$\Delta\mathbf{a}_i(t) = [u(t) - \mathbf{a}_i^T(t)\mathbf{x}(t)]\frac{\mathbf{x}(t)}{\|\mathbf{x}(t)\|^2}$,	
$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha h(i^*, i; t)[\mathbf{x}(t) - \mathbf{w}_i(t)]$,	
$\mathbf{a}_i(t+1) = \mathbf{a}_i(t) + \alpha' h(i^*, i; t)\Delta\mathbf{a}_i(t)$.	
3. Teste do modelo $(t = 1, 2, \ldots, N_2)$	
3.1 Escolha do neurônio vencedor:	
$i^*(t) = \arg\min_i \|\mathbf{x}(t) - \mathbf{w}_i(t)\|$.	
3.2 Cálculo da saída estimada:	
$\hat{u}(t) = \mathbf{a}_{i^*}^T(t)\mathbf{x}(t)$.	
3.3 Cálculo do erro:	
$e(t) = u(t) - \hat{u}(t)$.	
Saídas	
$\hat{u}(t)$: saída estimada e $e(t)$: erro	

Após a etapa de treinamento, a rede LLM foi testada com valores de (x, y) no intervalo de [0,1], e o resultado obtido pode ser visto na Figura 4.3(b). A superfície real dos dados é comparada juntamente com a superfície obtida a partir dos resultados de teste. O que se percebe na figura é que a rede LLM aproxima a superfície dos dados através de planos, buscando cobrir todo domínio da função. Esta aproximação não é precisa, tornando o resultado não satisfatório.

Na próxima seção, será apresentado um outro algoritmo que utiliza a filosofia de treinamento da rede SOM, mas que, diferentemente do modelo LLM, não faz uso de algoritmos lineares clássicos de filtragem. Este algoritmo é de grande importância neste trabalho e será discutido com detalhes. O resumo do algoritmo de modelagem LLM é apresentado na Tabela 4.1.

4.3 Memória Associativa Temporal por Quantização Vetorial

O modelo a ser descrito nesta seção, chamado de **Memória Associativa Temporal por Quantização Vetorial** (*Vector-Quantized Temporal Associative Memory* - VQTAM) (BARRETO; ARAÚJO, 2004), utiliza a rede SOM para implementar a quantização vetorial simultânea dos espaços de entrada e saída, a partir dos pares entrada-saída $\{\mathbf{x}(t), u(t)\}$, $t = 1, \ldots, N_1$.

Em Barreto & Araújo (2004), o modelo VQTAM foi aplicado a problemas de modelagem e controle não-linear, predição de séries temporais, controle preditivo e aprendizagem de trajetórias robóticas. O modelo VQTAM é uma simples extensão da rede SOM original, em que o vetor de entrada $\mathbf{x}(t)$ passa a ser composto de duas partes, ou seja:

- A primeira parte, representada por $\mathbf{x}^{in}(t)$, corresponde à informação de entrada do mapeamento dinâmico que se quer aproximar;
- A segunda, representada por $x^{out}(t)$, corresponde à informação de saída desse mesmo mapeamento.

Como conseqüência, os vetores de pesos dos neurônios também têm suas dimensões aumentadas. Formalmente, tem-se a seguinte nova representação para $\mathbf{x}(t)$ e $\mathbf{w}_i(t)$:

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{x}^{in}(t) \\ x^{out}(t) \end{pmatrix} \qquad \text{e} \qquad \mathbf{w}_i(t) = \begin{pmatrix} \mathbf{w}_i^{in}(t) \\ w_i^{out}(t) \end{pmatrix}. \tag{4.10}$$

Dependendo das variáveis escolhidas para definir o vetor $\mathbf{x}^{in}(t)$ e o escalar $x^{out}(t)$, com base na notação do modelo ARX, pode-se usar a rede SOM para aprender os modelos direto ou inverso de um dado sistema (e.g. planta industrial). Por exemplo, se o problema em mãos for a identificação do modelo direto de um determinado sistema dinâmico, então as seguintes definições se aplicam:

$$\mathbf{x}^{in}(t) = [y(t-1), y(t-2), \ldots, y(t-p); u(t), u(t-1), \ldots, u(t-q+1)]^T, \tag{4.11}$$

$$x^{out}(t) = y(t), \tag{4.12}$$

em que $\mathbf{x}^{in}(t)$ é o vetor que contém conjuntamente os valores prévios das variáveis de saída e entrada, enquanto $x^{out}(t)$ corresponde a saída da planta, sendo $q > 1$ e $p > 1$ as ordens de memória das sequências de entrada e de saída, respectivamente, e $q < p$.

Se a identificação do modelo inverso de um sistema dinâmico é requerida, então as seguintes definições se aplicam para que se possa aprender o mapeamento entrada-saída inverso:

$$\mathbf{x}^{in}(t) = [y(t-1), \ldots, y(t-p); u(t-1), \ldots, u(t-q)]^T, \tag{4.13}$$

$$x^{out}(t) = u(t), \tag{4.14}$$

em que $\mathbf{x}^{in}(t)$ é o vetor que contém conjuntamente os valores prévios das variáveis de saída e entrada, enquanto $x^{out}(t)$ corresponde a entrada da planta.

Durante a execução do algoritmo, o neurônio vencedor em t é determinado com base em $\mathbf{x}^{in}(t)$ apenas, logo,

$$i^*(t) = \arg\min_i \{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}. \tag{4.15}$$

Na atualização dos pesos, os termos $\mathbf{x}^{in}(t)$ e $x^{out}(t)$ são utilizados nas seguintes equações:

$$\mathbf{w}_i^{in}(t) = \mathbf{w}_i^{in}(t) + \alpha(t)h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)], \tag{4.16}$$

$$w_i^{out}(t) = w_i^{out}(t) + \alpha(t)h(i^*, i; t)[x^{out}(t) - w_i^{out}(t)], \tag{4.17}$$

em que $0 < \alpha(t) < 1$ é a taxa de aprendizagem e $h(i^*, i; t)$ é a função vizinhança definida na Seção 3.6.

É importante perceber que as regras de aprendizagem apresentadas nas Equações (4.16) e (4.17) seguem a formulação usual da rede SOM. A primeira regra executa uma quantização vetorial do espaço da entrada e a segunda atua de forma semelhante sobre o espaço da saída do mapeamento a ser aprendido. Assim, com o decorrer do treinamento, o modelo VQTAM aprende a associar os vetores-protótipos $\mathbf{w}_i^{in}(t)$ que formam o espaço quantizado de entrada com os protótipos $w_i^{out}(t)$ que formam o espaço quantizado de saída.

Para que a estratégia de *memória associativa* implementada pelo modelo VQTAM seja útil na identificação do modelo inverso de um sistema dinâmico, é necessário que a saída referente a um novo vetor de entrada possa ser estimada. Isto é feito a partir do protótipo do espaço de saída $w_i^{out}(t)$ associado ao neurônio vencedor $i^*(t)$, ou seja,

$$\hat{u}(t) \equiv w_{i^*}^{out}(t), \tag{4.18}$$

na qual o neurônio vencedor $i^*(t)$ é determinado segundo a Equação (4.15). De uma forma ilustrativa, esta relação associativa pode ser vista facilmente na Figura 4.4, em que um

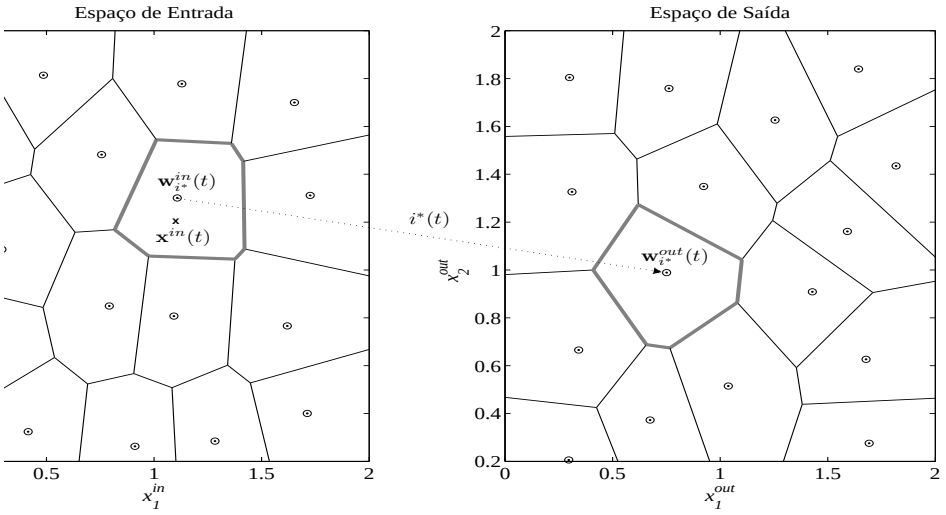


Figura 4.4 Representação da arquitetura do modelo VQTAM.

novo vetor de entrada $\mathbf{x}^{in}(t)$ é apresentado à rede em t, e o índice do neurônio vencedor $i^*(t)$ indicará que protótipo será usado para estimar a saída da rede.

É importante ressaltar alguns pontos importantes relativos ao modelo VQTAM:

(i) O neurônio vencedor $i^*(t)$ é o elemento responsável por "associar" ou "conectar" a porção de entrada do mapeamento, representada por $\mathbf{x}^{in}(t)$, com a porção de saída $x^{out}(t)$. Esta associação fica codificada no respectivo vetor de pesos, $\mathbf{w}_i^{in}(t)$ e o escalar $w_i^{out}(t)$.

(ii) Existe uma diferença *conceitual* muito importante entre a estratégia associativa do modelo VQTAM em relação àquela comumente usada no treinamento de redes supervisionadas (MLP ou RBF), que é baseada na redução *explícita* do erro de aproximação. Na abordagem supervisionada, o vetor $\mathbf{x}^{in}(t)$ é utilizado na entrada da rede, enquanto o escalar $x^{out}(t)$ é utilizado na saída (ver Figura 4.5) para calcular o erro de aproximação usado para guiar o ajuste dos pesos da rede. Quando se usa o modelo VQTAM, o escalar $x^{out}(t)$ é apresentado na entrada da rede juntamente com o vetor $\mathbf{x}^{in}(t)$, sem o cálculo *explícito* do erro de aproximação.

(iii) O modelo VQTAM, embora desenvolvido neste capítulo apenas com a rede SOM, em princípio também pode ser utilizado com outros algoritmos de quantização vetorial, tais como aqueles descritos no Capítulo 3.

(iv) O modelo VQTAM pode ser igualmente implementado via redes competitivas crescentes, tais como a rede *Growing Neural Gas* (FRITZKE, 1994) e *Growing SOM* (BAUER;

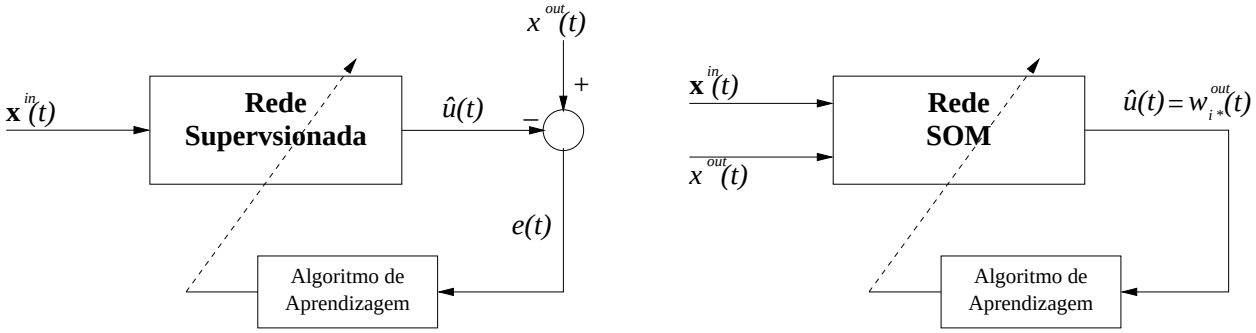


Figura 4.5 Diferença entre a abordagem supervisionada e a não-supervisionada na aproximação de funções, respectivamente.

VILLMANN, 1997), a fim de construir algoritmos que não necessitam de uma especificação prévia do número de neurônios.

Para finalizar, um sumário do funcionamento do modelo VQTAM é apresentado na Tabela 4.2. Na próxima seção, será apresentado a primeira proposta desta tese: um modelo linear local baseado na arquitetura VQTAM.

4.4 Modelo KSOM

Nesta seção o modelo VQTAM será usado para projetar modelos locais ARX, para uso subsequente em identificação de sistemas. O modelo a ser descrito utiliza uma abordagem baseado na escolha de $K \gg 1$ protótipos mais próximos do vetor de entrada $\mathbf{x}^{in}$, a cada t, sendo por isso doravante denotado pela sigla KSOM.

O modelo KSOM foi originalmente proposto por Barreto et al. (2003b) para aplicação em predição de séries temporais, sendo estendido nesta tese para aplicações em identificação de sistemas. Este modelo é construído a partir do modelo VQTAM, que é utilizado inicialmente para quantização conjunta dos espaços de entrada e de saída.

Após o treinamento do modelo VQTAM, determina-se o conjunto dos K protótipos mais próximos ao vetor de entrada atual $\mathbf{x}^{in}(t)$, denotados por $\{i_1^*, i_2^*, \ldots, i_K^*\}$:

$$\begin{aligned} i_1^*(t) &= \arg\min_i \left\{ \|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\| \right\} \\ i_2^*(t) &= \arg\min_{i \neq i_1^*} \left\{ \|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\| \right\} \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ i_K^*(t) &= \arg\min_{i \neq i_1^*, i_2^*, \ldots, i_{K-1}^*} \left\{ \|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\| \right\}. \end{aligned} \tag{4.19}$$

Tabela 4.2 Algoritmo de aprendizagem do modelo VQTAM.

Modelo VQTAM	
Constantes	**Valores típicos**
α : Taxa de aprendizagem	$0.001 < \alpha < 0.5$
σ : Abertura da vizinhança topológica	$0.001 < \sigma < (g/2)$
Entradas	
$\mathbf{x}^{in}(t)$: vetor de entrada, dimensão $(p+q) \times 1$	$x^{out}(t)$: variável de saída
Algoritmo	
1. Inicialização $(t = 0)$	
$\mathbf{w}_i^{in}, w_i^{out} \sim U(0,1)$, $i = 1, \ldots, g$	
2. Aprendizagem do modelo $(t = 1, 2, \ldots, N_1)$	
2.1 Escolha do neurônio vencedor:	
$i^*(t) = \text{argmin}_i \|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|$.	
2.2 Atualização dos vetores de pesos e de coeficientes:	
$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_{i^*} - \mathbf{r}_i\|^2}{2\sigma^2(t)}\right)$,	
$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t)h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)]$,	
$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t)h(i^*, i; t)[x^{out}(t) - w_i^{out}(t)]$.	
3. Saída do modelo $(t = 1, 2, \ldots, N_2)$	
3.1 Escolha do neurônio vencedor:	
$i^*(t) = \text{argmin}_i \|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|$.	
3.2 Cálculo da saída estimada:	
$\hat{u}(t) = w_{i^*}^{out}(t)$.	
3.3 Cálculo do erro:	
$e(t) = u(t) - \hat{u}(t)$.	
Saídas de interesse	
$\hat{u}(t)$: saída estimada, $e(t)$: erro	

A idéia subjacente ao modelo KSOM é usar os pares de protótipos $\{\mathbf{w}_{i_k^*}^{in}(t), w_{i_k^*}^{out}(t)\}_{k=1}^{K}$ dos K neurônios vencedores em t para construir um aproximador local de funções. Uma abordagem consiste em determinar um vetor de parâmetros que satisfaça a seguinte relação linear:

$$w_{i_k^*}^{out}(t) = \mathbf{a}^T(t)\mathbf{w}_{i_k^*}^{in}(t), \qquad k = 1, \ldots, K, \tag{4.20}$$

em que $\mathbf{a}(t) = [a_1(t), \ldots, a_p(t), b_1(t), \ldots, b_q(t)]^T$ é um vetor de coeficientes em t. A Equação (4.20) pode ser escrita na forma matricial da seguinte maneira:

$$\mathbf{p}(t) = \mathbf{R}(t)\mathbf{a}(t), \tag{4.21}$$

onde o vetor $\mathbf{p}$, chamado de vetor de predição, e a matriz $\mathbf{R}$, chamada de matriz de regressão, são

$$\mathbf{p}(t) = [w_{i_1^*}^{out}(t) \;\; w_{i_2^*}^{out}(t) \;\; \cdots \;\; w_{i_K^*}^{out}(t)]^T, \tag{4.22}$$

e

$$\mathbf{R}(t) = \begin{pmatrix} w^{in}_{i^*_1,1}(t) & w^{in}_{i^*_1,2}(t) & \cdots & w^{in}_{i^*_1,p+q}(t) \\ w^{in}_{i^*_2,1}(t) & w^{in}_{i^*_2,2}(t) & \cdots & w^{in}_{i^*_2,p+q}(t) \\ \vdots & \vdots & \vdots & \vdots \\ w^{in}_{i^*_K,1}(t) & w^{in}_{i^*_K,2}(t) & \cdots & w^{in}_{i^*_K,p+q}(t) \end{pmatrix}_{K\times(p+q)} . \tag{4.23}$$

Se $p+q=K$ na Equação (4.23), ou seja, se a dimensão do espaço de entrada da rede neural é igual ao número de neurônios vencedores utilizados, então a matriz $\mathbf{R}$ é quadrada. Neste caso, o vetor de coeficientes $\mathbf{a}(t)$ pode ser calculado simplesmente invertendo-se $\mathbf{R}$, ou seja,

$$\mathbf{a}(t) = \mathbf{R}^{-1}(t)\mathbf{p}(t). \tag{4.24}$$

Contudo, a situação mais usual ocorre para $(p+q) \ll K$, ou seja, para $\mathbf{R}$ sendo uma matriz retangular, com muito mais linhas do que colunas. Neste caso, pode-se utilizar o método dos *Mínimos Quadrados* (MQ) (AGUIRRE, 2007), tornando possível a inversão da matriz $\mathbf{R}$ condicionada ao caso em que ela tenha posto completo. De acordo com esta técnica, o vetor de coeficientes $\mathbf{a}(t)$ é determinado da seguinte maneira

$$\mathbf{a}(t) = \left(\mathbf{R}^T(t)\mathbf{R}(t) + \lambda\mathbf{I}\right)^{-1}\mathbf{R}^T(t)\mathbf{p}(t), \tag{4.25}$$

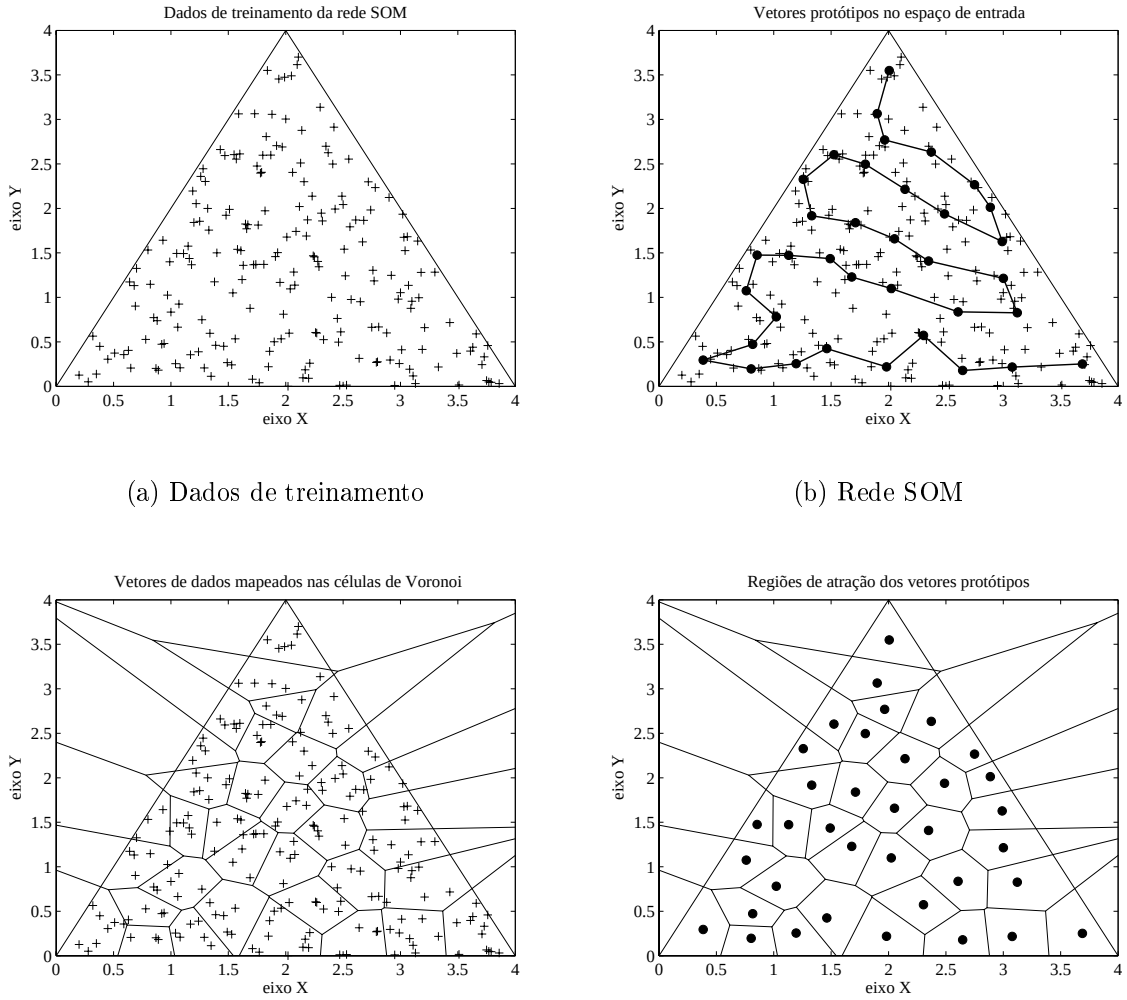
onde $\mathbf{I}$ é uma matriz identidade de ordem $p+q$ e $\lambda > 0$ (e.g. $\lambda = 0.001$) é uma constante positiva e de pequeno valor.

Uma vez determinado o vetor $\mathbf{a}(t)$ através da Equação (4.25), pode-se aproximar localmente a saída do mapeamento não-linear pela saída do seguinte modelo ARX:

$$\hat{u}(t) = \sum_{l=1}^{p} a_l(t)y(t-l) + \sum_{k=1}^{q} b_k(t)u(t-k) = \mathbf{a}^T(t)\mathbf{x}^{in}(t) \tag{4.26}$$

Em suma, para cada vetor de entrada $\mathbf{x}^{in}(t)$ determina-se um vetor de coeficientes $\mathbf{a}(t)$, calculado pela Equação (4.25), que será usado para construir o modelo linear local da Equação (4.26).

É importante ressaltar as diferenças do modelo KSOM em relação ao modelo LLM, que também implementa modelos locais lineares. A principal diferença está na forma como são construídos os modelos locais. No modelo LLM, cada neurônio tem um vetor de coeficientes associado, resultando num conjunto de g vetores-coeficientes. Uma vez encontrado o neurônio vencedor, basta usar a Equação (4.5). Já no modelo KSOM um único vetor de coeficientes é calculado dinamicamente a cada t, usando os vetores de pesos



(a) Dados de treinamento

(b) Rede SOM

(c) Células de Voronoi

(d) Vetores protótipos da rede SOM

Figura 4.6 Treinamento da rede SOM unidimensional.

dos K neurônios vencedores.

4.4.1 Trabalho Correlato

Um outro modelo linear local, muito semelhante ao modelo KSOM, foi proposto por Principe et al. (1998). Este modelo, aqui chamado de **Modelo SOM Local via Mínimos Quadrados** (*Local Least-Squares SOM* - LESSOM), também seleciona K neurônios para construir o modelo local através das Equações (4.21) a (4.25). Contudo, em vez de selecionar os K protótipos mais próximos do vetor de entrada $\mathbf{x}^{in}(t)$, são selecionados o protótipo do neurônio vencedor e os protótipos de seus $(K-1)$ vizinhos topológicos. Por causa da propriedade de preservação de topologia da rede SOM, os K neurônios se-

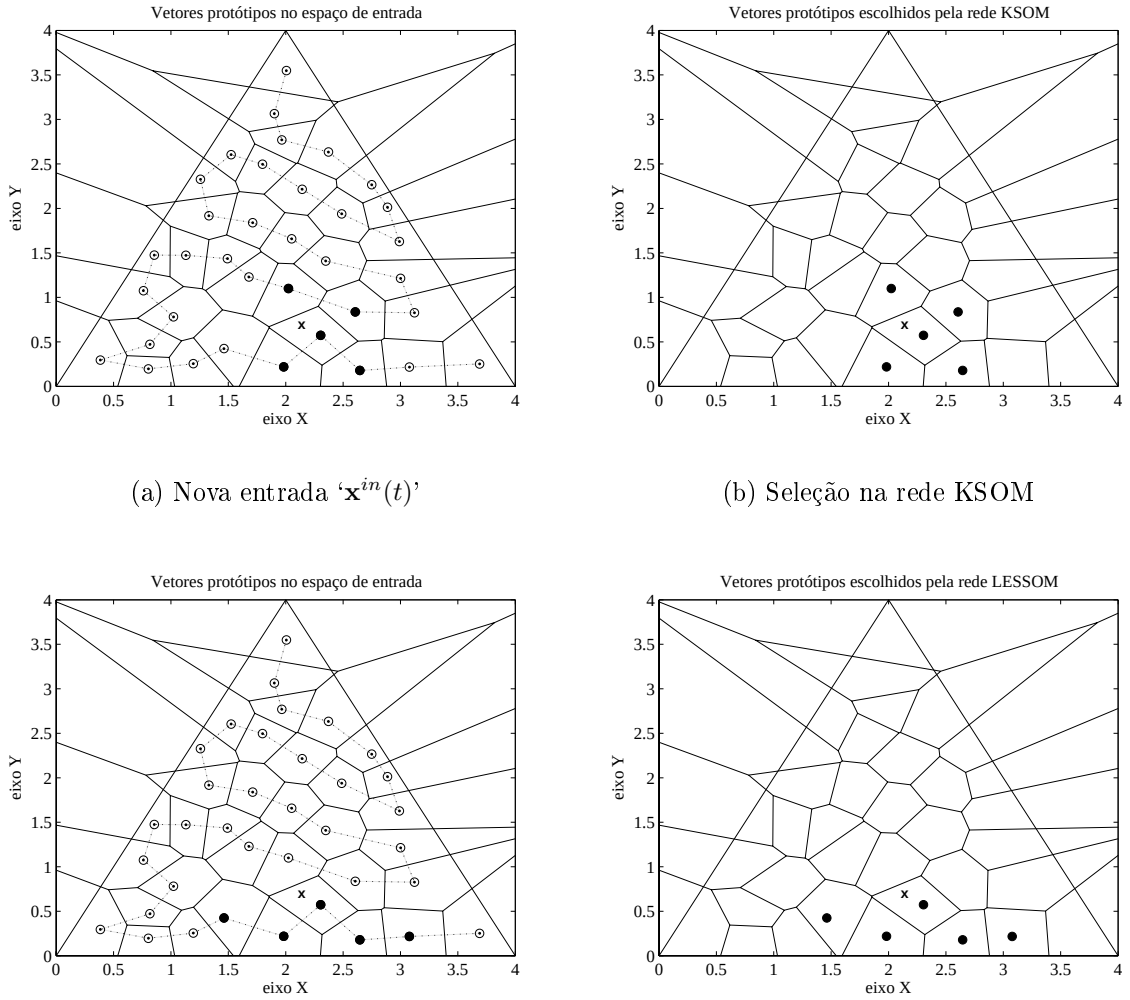
lecionados no modelo KSOM tendem a ser os mesmos do modelo LESSOM. A vantagem do algoritmo KSOM é que ele pode ser utilizado por qualquer rede neural competitiva, enquanto que o algoritmo LESSOM só é válido para a rede SOM.

A fim de ilustrar o processo de construção dos modelos KSOM e LESSOM, pode-se inicialmente destacar que o processo de aprendizado do modelo VQTAM, ilustrado na Figura 4.6 para uma rede SOM unidimensional (SOM-1D), é parte comum entre os dois modelos. Após esta etapa, tem-se a estimação de um novo vetor de coeficientes para cada novo vetor de entrada. A cada nova entrada, encontra-se um conjunto de K neurônios cujos vetores protótipos são determinados por critérios específicos dos modelos KSOM e LESSOM.

Na Figura 4.7(a), vê-se a escolha dos $K = 5$ protótipos mais próximos ao novo vetor $\mathbf{x}^{in}(t)$ realizado pelo modelo KSOM. Conforme destacado na figura, estes protótipos estão representados por pontos pretos e uma linha pontilhada interliga-os entre si determinando a vizinhança topológica da rede SOM-1D. A Figura 4.7(b) destaca o conjunto de vetores protótipos escolhidos, entre os demais pesos existentes na rede, acompanhado da entrada atual. Este conjunto de vetores de pesos são vistos como vizinhos espaciais, pois estão próximos fisicamente ao vetor de entrada atual.

O modelo LESSOM seleciona o vetor protótipo mais próximo ao vetor de entrada $\mathbf{x}^{in}(t)$ e mais $K-1$ vetores-protótipos correspondentes aos seus vizinhos topológicos para construção do modelo local. Este processo de seleção pode ser visto na Figura 4.7(c), em que os vetores-protótipos estão representados por pontos pretos e uma linha pontilhada interliga-os entre si determinando a vizinhança topológica da rede SOM-1D. A Figura 4.7(d) destaca o conjunto de vetores protótipos escolhidos, entre os demais pesos existentes na rede, acompanhado da entrada atual. Além do neurônio vencedor, quatro neurônios vizinhos a este são escolhidos, sendo dois de cada lado conforme visto na figura.

Os resultados obtidos para os algoritmos KSOM e LESSOM demonstram que, no caso do modelo KSOM, são escolhidos os $(K-1)$ protótipos mais próximos ao vetor de entrada atual além do neurônio vencedor, enquanto no caso do modelo LESSOM são escolhidos os $(K-1)$ protótipos mais próximos ao neurônio vencedor na sua vizinhança topológica. As células de Voronoi selecionadas pelo modelo KSOM delimitam a região do espaço de entrada que possui características mais semelhantes ao vetor de entrada atual, tornando a seleção mais localizada. Ao contrário do que é visto na rede LESSOM, em que as regiões selecionadas não estão tão próximas do vetor $\mathbf{x}^{in}(t)$.



(a) Nova entrada '$\mathbf{x}^{in}(t)$'

(b) Seleção na rede KSOM

(c) Nova entrada '$\mathbf{x}^{in}(t)$'

(d) Seleção na rede LESSOM

Figura 4.7 Metodologia utilizada pelos modelos KSOM e LESSOM para construção de modelos locais no espaço de entrada bidimensional.

4.4.2 Estratégias para Escolha do Parâmetro K

Nesta seção, comenta-se sobre a forma de escolha do valor do parâmetro K usado na construção do modelo KSOM. Duas estratégias são abordadas nesta tese: a primeira consiste em um método de busca exaustiva, guiando a escolha do parâmetro K pelo menor erro de generalização; a segunda consiste na análise do grau de condicionamento da matriz de regressores $\mathbf{R}$. O grau de condicionamento é uma grandeza importante a ser analisada, pois o mesmo define quanto a matriz de regressores é bem condicionada ou não, prevenindo assim problemas numéricos durante a inversão da matriz $\mathbf{R}$. Nesta estratégia, executa-se as etapas de treinamento/teste para diferentes valores do parâmetro K e verifica-se

o quão próximo o grau de condicionamento desta matriz está de 1, sugerindo assim um valor adequado para K.

A próxima seção será reservada para uma explanação rápida do modelo fuzzy Takagi-Sugeno (TS), pois as novas abordagens apresentadas neste trabalho terão seus desempenhos comparados a este modelo local linear. Na literatura é muito comum o emprego do modelo fuzzy TS em problemas de identificação direta de sistemas dinâmicos, como exemplo pode-se citar o trabalho de Rubio (2009), contudo, nesta tese a aplicação será a modelagem inversa de sistemas dinâmicos, logo, uma outra opção para o uso deste modelo. Para finalizar, o sumário do algoritmo KSOM é apresentado na Tabela 4.3.

4.5 Modelo Takagi-Sugeno

Identificação de modelo Fuzzy tem suas raízes nos trabalhos pioneiros de Sugeno e seus colaboradores (TAKAGI; SUGENO, 1985; SUGENO; YASUKAWA, 1993) e está associado com o tão conhecido modelo Fuzzy Takagi-Sugeno (TS), definido como um grupo especial de modelos baseados em regras com premissas fuzzy e lógicas funcionais que resultam do método de raciocínio Takagi-Sugeno-Kang:

$$R^i : \text{SE } x_1 \text{ é } \mathcal{X}_1^i \text{ E } \ldots \text{ E } x_n \text{ é } \mathcal{X}_n^i \text{ ENTÃO } \Rightarrow y_i = a_{i0} + a_{i1}x_1 + \ldots + a_{in}x_n;\ i = \{1, g^f\} \tag{4.27}$$

onde R^i denota a i-ésima regra fuzzy; g^f é o número de regras fuzzy; $\mathbf{x}$ é o vetor de entrada: $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$; $\mathcal{X}_j^i$ denota os conjuntos fuzzy de premissas, $j = \{1, n\}$; y_i é a saída do i-ésimo sub-sistema linear; a_{il} são seus parâmetros, $l = \{0, n\}$, sendo $n = p+q$.

O grau de pertinência μ_{ij} do vetor de entrada $\mathbf{x}$ associado a cada regra fuzzy R^i é proporcional ao nível de contribuição do correspondente modelo linear no cálculo da saída total do modelo TS. Para conjunto fuzzy Gaussianos, tem-se

$$\mu_{ij} = e^{-\gamma \|x_j - w_{ij}^f\|^2};\ \ i = \{1, g^f\} \text{ e } j = \{1, n\} \tag{4.28}$$

onde $\gamma = 4/r^2$, e r é uma constante positiva, que define a dispersão, isto é, a zona de influência do i-ésimo modelo (raio da vizinhança de um ponto de dados); um valor tão grande de r leva a média, um valor tão pequeno leva a *over-fitting*; (valores de $r \in [0.3; 0.5]$ podem ser recomendados (CHIU, 1994)); $\mathbf{w}_i^f$ é o ponto focal (ou vetor de pesos) da i-ésima regra premissa fuzzy.

O nível de disparo de cada regra é definido como produto dos respectivos conjuntos

Tabela 4.3 Algoritmo de aprendizagem do modelo KSOM.

Modelo KSOM

Constantes	**Valores típicos**
α : Taxa de aprendizagem	$0.001 < \alpha < 0.5$
σ : Abertura da vizinhança topológica	$0.001 < \sigma < (q/2)$
λ : Constante de regularização	$\lambda = 0.001$

Entradas

$\mathbf{x}^{in}(t)$: vetor de entrada, dimensão $(p+q) \times 1$

$x^{out}(t)$: escalar de saída

Algoritmo

1. Inicialização $(t = 0)$

$\mathbf{w}_i^{in}, w_i^{out} \sim U(0,1)$, $i = 1, \ldots, g$

2. Aprendizagem do modelo $(t = 1, 2, \ldots, N_1)$

2.1 Escolha do neurônio vencedor:

$i^*(t) = \text{argmin}_i \|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|$.

2.2 Atualização dos vetores de pesos e de coeficientes:

$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_{i^*} - \mathbf{r}_i\|^2}{2\sigma^2(t)}\right)$,

$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t)h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)]$,

$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t)h(i^*, i; t)[x^{out}(t) - w_i^{out}(t)]$.

3. Saída do modelo $(t = 1, 2, \ldots, N_2)$

3.1 Escolha dos K neurônios mais próximos (vencedores) ao vetor de entrada $\mathbf{x}^{in}(t)$:

$\{i_1^*, i_2^*, \ldots, i_K^*\}$.

3.2 Obtenção do vetor $\mathbf{p}(t)$ e da matriz de regressão $\mathbf{R}(t)$:

$$\mathbf{p}(t) = [w_{i_1^*}^{out}(t) \;\; w_{i_2^*}^{out}(t) \;\; \cdots \;\; w_{i_K^*}^{out}(t)]^T,$$

$$\mathbf{R}(t) = \begin{pmatrix} w_{i_1^*,1}^{in}(t) & w_{i_1^*,2}^{in}(t) & \cdots & w_{i_1^*,p+q}^{in}(t) \\ w_{i_2^*,1}^{in}(t) & w_{i_2^*,2}^{in}(t) & \cdots & w_{i_2^*,p+q}^{in}(t) \\ \vdots & \vdots & \vdots & \vdots \\ w_{i_K^*,1}^{in}(t) & w_{i_K^*,2}^{in}(t) & \cdots & w_{i_K^*,p+q}^{in}(t) \end{pmatrix}.$$

3.3 Obtenção do vetor de coeficientes no instante t:

$\mathbf{a}(t) = \left(\mathbf{R}^T(t)\mathbf{R}(t) + \lambda\mathbf{I}\right)^{-1}\mathbf{R}^T(t)\mathbf{p}(t)$.

3.4 Cálculo da saída estimada:

$\hat{u}(t) = \mathbf{a}^T(t)\mathbf{x}^{in}(t)$.

3.5 Cálculo do erro:

$e(t) = u(t) - \hat{u}(t)$.

Saídas

$\hat{u}(t)$: saída estimada, $e(t)$: erro

fuzzy para esta regra

$$\begin{aligned}
\tau_i &= \mu_{i1}(x_1) \times \mu_{i2}(x_2) \times \ldots \times \mu_{in}(x_n) = \prod_{j=1}^{n} \mu_{ij}(x_j) \\
&= e^{-\gamma(x_1 - w_{i1}^f)^2} \times \ldots \times e^{-\gamma(x_n - w_{in}^f)^2} \\
&= e^{-\gamma \sum_{j=1}^{n} (x_j - w_{ij}^f)^2} \\
\tau_i &= e^{-\gamma \|\mathbf{x} - \mathbf{w}_i^f\|^2}
\end{aligned} \tag{4.29}$$

A saída estimada do modelo TS é calculada pela média ponderada das contribuições individuais das regras

$$\begin{aligned}
\hat{y} &= \frac{\sum_{i=1}^{g^f} \tau_i y_i}{\sum_{i=1}^{g^f} \tau_i} \\
\hat{y} &= \sum_{i=1}^{g^f} \lambda_i y_i = \sum_{i=1}^{g^f} \lambda_i (\mathbf{x}_e^T \pi_i)
\end{aligned} \tag{4.30}$$

onde λ_i é o nível de disparo normalizado da i-ésima regra fuzzy; y_i representa a saída do i-ésimo modelo linear; $\pi_i = [a_{i0}\ a_{i1}\ a_{i2}\ \ldots\ a_{in}]^T$, é o vetor de parâmetros do i-ésimo modelo linear; $\mathbf{x}_e = [1\ \mathbf{x}^T]^T$ é o vetor de dados de entrada expandido.

Geralmente, o problema de identificação de um modelo TS é dividido em duas subtarefas (YAGER; FILEV, 1994; TAKAGI; SUGENO, 1985; CAUDELL; NEWMAN, 1993):

1) Aprendendo a parte da premissa do modelo (Eq. (4.27)), que consiste da determinação dos pontos focais das regras, isto é, os centróides ($\mathbf{w}_i^f; i = [1, g^f]$) e as dispersões ($r$) das funções de pertinência de cada premissa fuzzy;

2) Obtenção dos parâmetros dos sub-sistemas lineares ($a_{ij}; i = [1, g^f], j = [0, n]$) das lógicas funcionais.

4.5.1 Obtenção dos Pontos Focais por Quantização Vetorial

Nesta etapa da identificação dos modelos TS, busca-se através do uso de algoritmos de quantização vetorial, tais como K-médias, WTA, etc. a obtenção dos pontos focais de cada regra fuzzy. Os algoritmos utilizados neste trabalho foram apresentados anteriormente no Cap. 3, e esta etapa corresponde ao estágio de aprendizado da rede neural em que são apresentados os vetores de dados de treinamento.

4.5.2 Cálculo dos Vetores de Parâmetros dos Modelos Locais

Após a definição dos pontos focais ($\mathbf{w}_i^f$) para cada regra fuzzy, a etapa subsequente será destinada para o cálculo dos vetores de parâmetros (π_i) das respectivas regras. A estimação dos parâmetros pode ser transformado em um problema de mínimos quadrados (LJUNG, 1999). Inicialmente, isto é realizado eliminando-se a operação de somatório existente na Equação (4.30), como segue,

$$\begin{aligned}
\hat{y} &= \sum_{i=1}^{g^f} \lambda_i(\mathbf{x}_e^T \pi_i) \\
&= \lambda_1(\mathbf{x}_e^T \pi_1) + \lambda_2(\mathbf{x}_e^T \pi_2) + \cdots + \lambda_{g^f}(\mathbf{x}_e^T \pi_{g^f}) \\
&= \lambda_1(a_{10} + a_{11}x_1 + \ldots + a_{1n}x_n) + \cdots + \lambda_{g^f}(a_{g^f 0} + a_{g^f 1}x_1 + \ldots + a_{g^f n}x_n) \\
&= [\lambda_1 \ \lambda_1 x_1 \ \ldots \ \lambda_1 x_n]\pi_1 + \ldots + [\lambda_{g^f} \ \lambda_{g^f} x_1 \ \ldots \ \lambda_{g^f} x_n]\pi_{g^f} \\
\hat{y} &= [\lambda_1 \mathbf{x}_e^T]\pi_1 + \ldots + [\lambda_{g^f} \mathbf{x}_e^T]\pi_{g^f},
\end{aligned}$$

e depois substituindo-o por uma expressão equivalente vetorial de y

$$y = \psi^T \theta, \tag{4.31}$$

onde $\theta = [\pi_1^T, \pi_2^T, \ldots, \pi_{g^f}^T]^T$ é um vetor composto dos parâmetros dos modelos lineares; $\psi = [\lambda_1 \mathbf{x}_e^T, \lambda_2 \mathbf{x}_e^T, \ldots, \lambda_{g^f} \mathbf{x}_e^T]^T$ é um vetor de entradas que são ponderados pelos níveis de disparo normalizado de todas as regras fuzzy.

Para um dado conjunto de entrada-saída de treinamento $(\mathbf{x}_t^T, y_t), t = [1, N_1]$, busca-se o vetor de parâmetros do modelo linear θ que minimize a função custo a seguir,

$$J = \sum_{t=1}^{N_1} (y_t - \psi_t^T \theta)^2, \tag{4.32}$$

onde $\psi_t = [\lambda_1(\mathbf{x}_t^T)\mathbf{x}_{et}^T, \lambda_2(\mathbf{x}_t^T)\mathbf{x}_{et}^T, \ldots, \lambda_{g^f}(\mathbf{x}_t^T)\mathbf{x}_{et}^T]; \mathbf{x}_{et} = [1 \ \mathbf{x}_t^T]^T$ e N_1 é o tamanho do conjunto de treinamento. O vetor de parâmetros θ pode ser estimado pelo algoritmo dos mínimos quadrados recursivos (RLS) (TAKAGI; SUGENO, 1985; CHIU, 1994) ou pela pseudo-inversa (PRINCIPE et al., 2000).

Alternativamente, a função custo (Eq. (4.32)) pode ser escrita na forma vetorial como

$$J = (Y - \Psi^T \theta)^T (Y - \Psi^T \theta) \tag{4.33}$$

onde a matriz Ψ e o vetor Y são formados por ψ_t^T e y_t, $t = [1, N_1]$, respectivamente.

Logo o vetor θ que minimize a Eq. (4.33) poderá ser obtido por pseudo-inversa

$$\theta = (\Psi^T \Psi)^{-1} \Psi^T Y. \tag{4.34}$$

4.6 Conclusão

Neste capítulo, foram descritos em detalhes três arquiteturas neurais baseadas na rede SOM para construção de modelos lineares locais, a saber: LLM, VQTAM e KSOM. Duas destas arquiteturas, LLM e KSOM, embora já tivessem sido utilizadas em problemas correlatos ao de identificação de sistemas dinâmicos, não haviam sido aplicadas antes a este problema. Em princípio, qualquer algoritmo de quantização vetorial pode ser utilizado para implementar os três modelos supracitados. No caso do modelo fuzzy, pode-se também implementá-lo com diferentes algoritmos de quantização vetorial para obtenção dos pontos focais. Sobre este aspecto, será destinado uma análise a ser feita no Capítulo 6, em que se verificará quanto os modelos são semelhantes entre si, com base nos resíduos gerados pelos mesmos.

No próximo capítulo serão apresentadas as duas principais contribuições desta tese, ou seja, duas extensões do modelo KSOM que possuem a vantagem de serem modelos locais cujos vetores de parâmetros não precisam ser estimados a cada t, o que diminui consideravelmente o custo computacional dos modelos.

5 *Modelos Lineares Locais para Identificação de Sistemas: Novas Propostas*

Neste capítulo serão apresentadas mais duas novas contribuições deste trabalho no desenvolvimento de arquiteturas de modelos lineares locais aplicados ao problema de identificação de sistemas. Diferentemente do modelo KSOM, descrito no capítulo anterior, em que o vetor de coeficientes do modelo linear local é estimado em cada instante de tempo, nos dois modelos a serem propostos, cada vetor-protótipo terá seu próprio vetor de coeficientes. Esta estratégia visa diminuir o custo computacional do modelo KSOM durante a fase de utilização do modelo.

5.1 Introdução

Neste capítulo propõe-se dois novos métodos para o projeto de modelos lineares locais múltiplos para identificação de sistemas. Como mencionado anteriormente, os dois novos métodos são extensões do modelo KSOM. Mais especificamente, tais extensões podem ser compreendidas como versões de modelos múltiplos do algoritmo KSOM, isto é, enquanto que para a abordagem KSOM somente um único modelo linear é requerido, cujo vetor de coeficientes é estimado a cada passo, para as abordagens a serem descritas a seguir cada vetor-protótipo terá um modelo linear associado (representado por um vetor de coeficientes).

Mais especificamente, enquanto que no modelo KSOM o único vetor de coeficientes do modelo linear é variante no tempo, nos modelos a serem propostos o vetor de coeficientes de cada modelo local é fixado após a fase de aprendizagem. A diferença entre as abordagens propostas reside basicamente na forma com que os vetores de coeficientes são estimados. Um deles usa os vetores-protótipos da rede SOM, enquanto o outro usa

diretamente os vetores de dados mapeados para os protótipos de interesse. Detalhes sobre cada modelo proposto são apresentados a seguir.

5.2 Modelo P-MKSOM

O primeiro método será doravante chamado de modelo **KSOM múltiplo baseado em protótipos**, ou simplesmente modelo P-MKSOM. Este possui a mesma estrutura do modelo KSOM, porém cada vetor protótipo possui associado a ele um vetor de coeficientes de um modelo linear local para estimar a saída do mapeamento. O processo de estimação dos vetores de coeficientes emprega os vetores-protótipos do modelo VQTAM. Assim, o primeiro passo na construção do modelo P-MKSOM requer o treinamento do modelo VQTAM (veja Seção 4.3).

Para estimar o vetor de coeficientes associado a cada neurônio i, $i = 1, ..., g$, do modelo VQTAM, primeiro determina-se os K vizinhos mais próximos do vetor protótipo $\mathbf{w}_i^{in}$, como segue

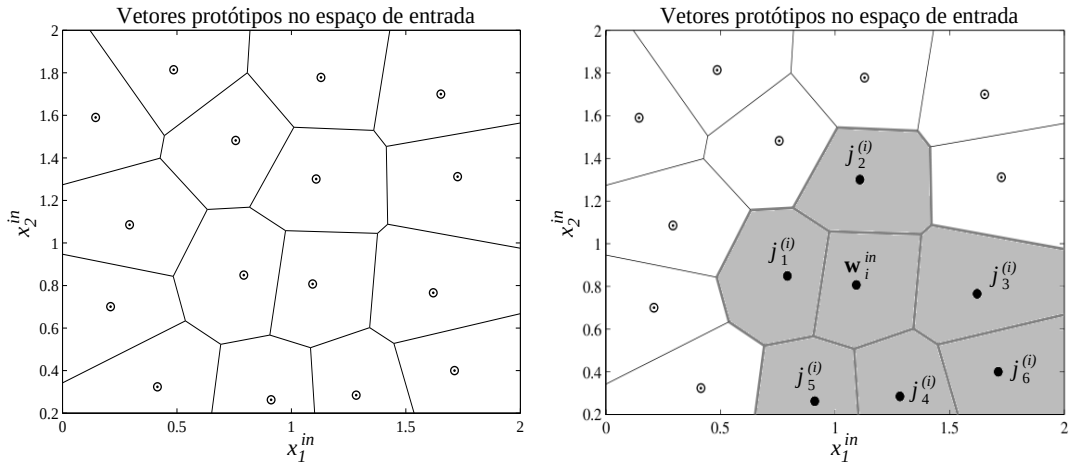
$$\begin{aligned} j_1^{(i)} &= \arg\min_{j \neq i} \left\{ \|\mathbf{w}_i^{in} - \mathbf{w}_j^{in}\| \right\}, \\ j_2^{(i)} &= \arg\min_{j \neq i, j_1} \left\{ \|\mathbf{w}_i^{in} - \mathbf{w}_j^{in}\| \right\}, \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ j_K^{(i)} &= \arg\min_{j \neq i, j_1, \ldots, j_{K-1}} \left\{ \|\mathbf{w}_i^{in} - \mathbf{w}_j^{in}\| \right\}. \end{aligned} \tag{5.1}$$

Seja $\mathcal{J}_i = i \cup \{j_k^{(i)}\}_{k=1}^K$ o conjunto contendo os índices dos K vetores protótipos que são mais próximos a $\mathbf{w}_i^{in}$, incluindo o do neurônio i. A Figura 5.1 ilustra um exemplo no processo de seleção de $K = 6$ vizinhos mais próximos do vetor protótipo $\mathbf{w}_i^{in}$ no espaço dos dados de entrada para o caso do plano bidimensional.

Uma vez que o conjunto $\mathcal{J}_i$ é determinado para cada neurônio i, constrói-se g modelos locais usando os vetores-protótipos cujos os índices pertencem a $\mathcal{J}_i$. Então, associado ao neurônio i, estima-se um vetor de coeficientes $\mathbf{a}_i \in \mathbb{R}^{p+q}$ calculado pelo método dos mínimos quadrados:

$$\mathbf{a}_i = \left(\mathbf{R}_i^T \mathbf{R}_i + \lambda \mathbf{I}\right)^{-1} \mathbf{R}_i^T \mathbf{b}_i^{out}, \tag{5.2}$$

onde $\mathbf{I}$ é uma matriz identidade de ordem $(p+q) \times (p+q)$ e $0 < \lambda \ll 1$ (e.g. $\lambda = 0.001$) é uma constante de regularização. Vale lembrar que as constantes p e q são as ordens de



(a) (b)

Figura 5.1 Primeira etapa na obtenção do i-ésimo neurônio do modelo P-MKSOM: (a) células de Voronoi associadas a cada vetor-protótipo depois do treinamento do modelo VQ-TAM; (b) conjunto de $K = 6$ vizinhos mais próximos de um dado vetor protótipo $\mathbf{w}_i^{in}$, cujo índices são armazenados no conjunto $(\mathcal{J}_i)$.

memória dos dados de entrada e de saída, o vetor de entrada em t do modelo VQTAM é

$$\mathbf{x}^{in}(t) = [y(t-1), \ldots, y(t-p); u(t-1), \ldots, u(t-q)]^T, \tag{5.3}$$

$$x^{out}(t) = u(t). \tag{5.4}$$

O vetor $\mathbf{b}_i^{out} \in \mathbb{R}^{K+1}$ é composto das partes de saída dos K vetores-protótipos cujos os índices pertençam ao conjunto $\mathcal{J}_i$, isto é,

$$\mathbf{b}_i^{out} = \left[w_i^{out} \;\; w_{j_1^{(i)}}^{out} \;\; \cdots \;\; w_{j_K^{(i)}}^{out} \right]^T, \tag{5.5}$$

e a matriz $\mathbf{R}_i \in \mathbb{R}^{(K+1)\times(p+q)}$ é composta das partes das entradas dos mesmos K vetores-protótipos:

$$\mathbf{R}_i = \begin{pmatrix} w_{i,1}^{in} & w_{i,2}^{in} & \cdots & w_{i,p+q}^{in} \\ w_{j_1^{(i)},1}^{in} & w_{j_1^{(i)},2}^{in} & \cdots & w_{j_1^{(i)},p+q}^{in} \\ \vdots & \vdots & \vdots & \vdots \\ w_{j_K^{(i)},1}^{in} & w_{j_K^{(i)},2}^{in} & \cdots & w_{j_K^{(i)},p+q}^{in} \end{pmatrix} = \begin{pmatrix} (\mathbf{w}_i^{in})^T \\ \left(\mathbf{w}_{j_1^{(i)}}^{in}\right)^T \\ \vdots \\ \left(\mathbf{w}_{j_K^{(i)}}^{in}\right)^T \end{pmatrix}. \tag{5.6}$$

A segunda etapa do processo de estimação do vetor de coeficientes do i-ésimo neurônio está ilustrada na Figura 5.2. Após a seleção dos K protótipos mais próximos (ver Equação (5.1)), os neurônios cujos índices pertencem ao conjunto $\mathcal{J}_i$ tem seus protótipos

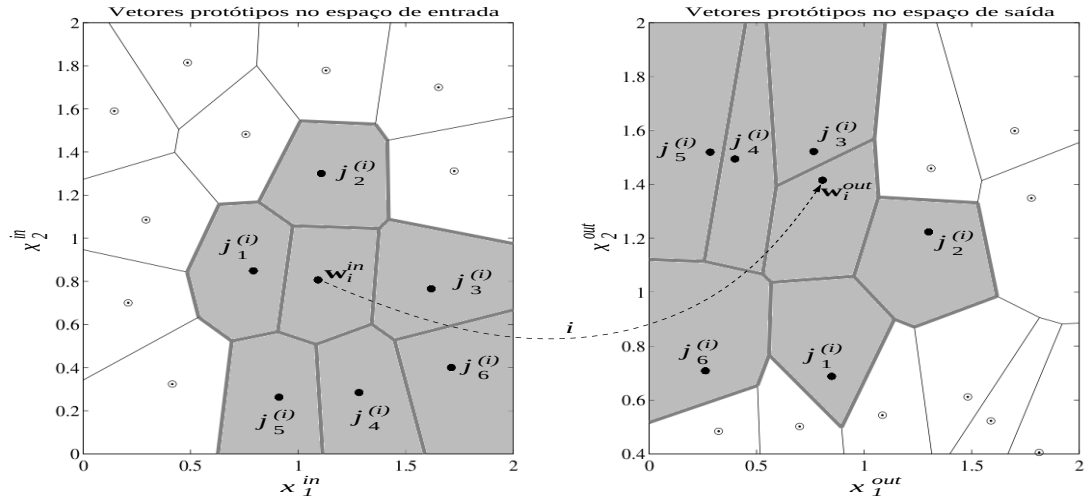


Figura 5.2 Segunda etapa na obtenção do modelo local do i-ésimo neurônio do modelo P-MKSOM. Separação dos $K = 6$ vetores-protótipos mais próximos em duas partes, uma referente ao espaço de entrada e a outra ao espaço de saída.

separados em duas partes, a fim de construir o vetor $\mathbf{b}_i^{out}$ e a matriz $\mathbf{R}_i$.

Uma vez que os modelos locais dos g neurônios tenham sido obtidos, eles podem ser usados para prever a saída do mapeamento de interesse para novos vetores de entrada. Nesta tese, o interesse está no uso do modelo P-MKSOM para identificar o modelo inverso de uma planta não-linear. Isto pode ser feito por meio da seguinte equação:

$$\hat{u}(t) = \mathbf{a}_{i^*}^T \mathbf{x}^{in}(t), \tag{5.7}$$

em que o vetor de entrada é definido como na Equação (5.3) e $i^*(t)$ é o índice do neurônio vencedor no instante t, determinado por meio da seguinte expressão:

$$i^*(t) = \arg\min_{i \in \mathcal{A}} \{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\}. \tag{5.8}$$

O erro de predição (resíduo) no instante t é então definido como $e(t) = u(t) - \hat{u}(t)$. Para finalizar, um sumário do modelo P-MKSOM é apresentado na Tabela 5.1. Na próxima seção será apresentada uma outra extensão do modelo KSOM, similar ao modelo P-MKSOM mas que usa os agrupamentos (*clusters*) de dados de entrada em vez dos protótipos para estimar os vetores de coeficientes dos g modelos locais.

Tabela 5.1 Algoritmo de treinamento e teste do modelo P-MKSOM.

Modelo P-MKSOM

Constantes	**Valor típico**
α : Taxa de aprendizagem	$0.001 < \alpha < 0.5$
σ : Abertura da vizinhança topológica	$0.001 < \sigma < (g/2)$
λ : Constante de regularização	$\lambda = 0.001$

Entradas

$\mathbf{x}^{in}(t)$: vetor de entrada, dimensão $(p+q) \times 1$

$x^{out}(t)$: variável de saída

Algoritmo

1. Inicialização $(t = 0)$

$\mathbf{w}_i^{in}, w_i^{out} \sim U(0,1)$, $i = 1, 2, \ldots, g$

2. Aprendizagem do modelo $(t = 1, 2, \ldots, N_1)$

2.1 Busca pelo neurônio vencedor:

$i^*(t) = \text{argmin}_i \|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|$.

2.2 Atualização dos vetores protótipos de entrada e escalares de saída do neurônio i:

$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_{i^*} - \mathbf{r}_i\|^2}{2\sigma^2(t)}\right)$,

$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t) h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)]$,

$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t) h(i^*, i; t)[x^{out}(t) - w_i^{out}(t)]$.

3. Estimação do vetor de coeficientes do neurônio *i*

3.1 Determinação dos índices dos K vetores protótipos mais próximos ao vetor $\mathbf{w}_i^{in}$:

$\mathcal{J}_i = i \cup \{j_k^{(i)}\}_{k=1}^{K}$.

3.2 Construção da matriz $\mathbf{R}_i$ e do vetor $\mathbf{b}_i^{out}$:

$$\mathbf{b}_i^{out} = \begin{bmatrix} w_i^{out} & w_{j_1^{(i)}}^{out} & \cdots & w_{j_K^{(i)}}^{out} \end{bmatrix}^T_{(K+1)\times 1}.$$

$$\mathbf{R}_i = \begin{pmatrix} w_{i,1}^{in} & w_{i,2}^{in} & \cdots & w_{i,p+q}^{in} \\ w_{j_1^{(i)},1}^{in} & w_{j_1^{(i)},2}^{in} & \cdots & w_{j_1^{(i)},p+q}^{in} \\ \vdots & \vdots & \vdots & \vdots \\ w_{j_K^{(i)},1}^{in} & w_{j_K^{(i)},2}^{in} & \cdots & w_{j_K^{(i)},p+q}^{in} \end{pmatrix}_{(K+1)\times(p+q)},$$

3.3 Cálculo do vetor de coeficientes associado a $\mathbf{w}_i^{in}$:

$\mathbf{a}_i = \left(\mathbf{R}_i^T \mathbf{R}_i + \lambda \mathbf{I}\right)^{-1} \mathbf{R}_i^T \mathbf{b}_i^{out}$.

4. Saída do modelo $(t = 1, 2, \ldots, N_2)$

4.1 Busca pelo neurônio vencedor:

$i^*(t) = \text{argmin}_i \|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|$.

4.2 Cálculo da saída estimada: $\hat{u}(t) = \mathbf{a}_{i^*}^T \mathbf{x}^{in}(t)$

3.4 Cálculo do erro: $e(t) = u(t) - \hat{u}(t)$

Saídas

$\hat{u}(t)$: saída estimada, $e(t)$: erro

5.3 Modelo D-MKSOM

O segundo método proposto será doravante chamado de modelo **KSOM múltiplo baseado em clusters de dados**, ou simplesmente modelo D-MKSOM. Este é similar ao modelo P-MKSOM, diferindo somente na forma como o vetor de coeficientes do i-ésimo neurônio, $\mathbf{a}_i$, $i = 1, ..., g$ é calculado. Em vez de usar o vetor-protótipo do i-ésimo neurônio e de seus K vizinhos mais próximos, o modelo D-MKSOM estima o vetor de coeficientes $\mathbf{a}_i$ usando os vetores de dados (de treinamento) pertencentes à subregião do espaço de entrada formada pela união das células de Voronoi do i-ésimo neurônio e de seus K vizinhos mais próximos.

O primeiro e o segundo passos na construção do modelo D-MKSOM são os mesmos do modelo P-MKSOM: (i) treinar o modelo VQTAM usando os dados de treinamento disponíveis. (ii) determinar o conjunto $\mathcal{J}_i = i \cup \{j_k^{(i)}\}_{k=1}^{K}$ dos índices dos K vizinhos mais próximos de $\mathbf{w}_i^{in}$, $i = 1, ..., g$, tal como definido na Equação (5.1).

O terceiro passo consiste em encontrar todos os vetores de dados que são mapeados para as células de Voronoi representadas pelos protótipos $\{\mathbf{w}_i^{in}, \mathbf{w}_{j_1^{(i)}}^{in}, \mathbf{w}_{j_2^{(i)}}^{in}, \ldots, \mathbf{w}_{j_K^{(i)}}^{in}\}$. Isto deve ser feito para cada neurônio da rede, ou seja, para $i = 1, \ldots, g$.

Matematicamente, o terceiro passo pode ser descrito da seguinte maneira. Seja $\mathbf{X}_i^{in}$ o conjunto de vetores de dados de entrada pertencentes à célula de Voronoi do neurônio i, isto é,

$$\mathbf{X}_i^{in} = \{\mathbf{x}^{in} \in \mathbb{R}^{p+q} | i = \arg\min_{l=1,..,g} \|\mathbf{x}^{in} - \mathbf{w}_l^{in}\|\}, \tag{5.9}$$

e seja $\mathbf{x}_i^{out}$ o correspondente conjunto de escalares de saída (x^{out}) associados com cada vetor $\mathbf{x}^{in} \in \mathbf{X}_i^{in}$, isto é,

$$\mathbf{x}_i^{out} = \{x^{out} \in \mathbb{R} | i = \arg\min_{l=1,..,g} \|\mathbf{x}^{in} - \mathbf{w}_l^{in}\|\}. \tag{5.10}$$

Da mesma forma, $\mathbf{X}_{j_k^{(i)}}^{in}$, $k = 1, \ldots, K$, é o conjunto de vetores de dados de entrada pertencentes à célula de Voronoi do neurônio $j_k^{(i)}$ (k-ésimo vizinho mais próximo do neurônio i):

$$\mathbf{X}_{j_k^{(i)}}^{in} = \{\mathbf{x}^{in} \in \mathbb{R}^{p+q} | j_k^{(i)} = \arg\min_{l=1,..,g} \|\mathbf{x}^{in} - \mathbf{w}_l^{in}\|\}, \tag{5.11}$$

e $\mathbf{x}_{j_k^{(i)}}^{out}$ é o conjunto correspondente de escalares de saída associados com cada vetor $\mathbf{x}^{in} \in \mathbf{X}_{j_k^{(i)}}^{in}$, isto é,

$$\mathbf{x}_{j_k^{(i)}}^{out} = \{x^{out} \in \mathbb{R} | j_k^{(i)} = \arg\min_{l=1,..,g} \|\mathbf{x}^{in} - \mathbf{w}_l^{in}\|\}. \tag{5.12}$$

Uma vez que os conjuntos $\{\mathbf{X}_i^{in}, \mathbf{x}_i^{out}\}$, $\{\mathbf{X}_{j_1^{(i)}}^{in}, \mathbf{x}_{j_1^{(i)}}^{out}\}$, $\{\mathbf{X}_{j_2^{(i)}}^{in}, \mathbf{x}_{j_2^{(i)}}^{out}\}$, ..., $\{\mathbf{X}_{j_K^{(i)}}^{in}, \mathbf{x}_{j_K^{(i)}}^{out}\}$ tenham sido determinados para o neurônio i e seus K vizinhos mais próximos, constrói-se o modelo linear local para este neurônio.

Assume-se que o conjunto $\mathbf{x}_i^{out}$ tem cardinalidade[1] $n^{(i)}$ e que os conjuntos $\mathbf{x}_{j_k^{(i)}}^{out}$ têm cardinalidades $n_{j_k}^{(i)}$, $k = 1, \ldots, K$. Então, o número total de vetores mapeados para o neurônio i e seus K vizinhos mais próximos é dado por

$$n_i = n^{(i)} + n_{j_1}^{(i)} + n_{j_2}^{(i)} + \cdots + n_{j_K}^{(i)}. \tag{5.13}$$

Então, assumindo que os conjuntos $\mathbf{x}_i^{out}$ e $\mathbf{x}_{j_k^{(i)}}^{out}$, $k = 1, \ldots, K$ são vetores-linha, pode-se definir o vetor $\mathbf{d}_i^{out} \in \mathbb{R}^{n_i}$ como segue

$$\mathbf{d}_i^{out} = \begin{bmatrix} \mathbf{x}_i^{out} & \mathbf{x}_{j_1^{(i)}}^{out} & \cdots & \mathbf{x}_{j_K^{(i)}}^{out} \end{bmatrix}_{n_i \times 1}^{T}. \tag{5.14}$$

Similarmente, assumindo que o conjunto $\mathbf{X}_i^{in}$ está organizado como uma matriz de dados de dimensão $(p+q) \times n^{(i)}$ e que os conjuntos $\mathbf{X}_{j_k^{(i)}}^{in}$ são organizados como matrizes de dados de dimensões $(p+q) \times n_{j_k}^{(i)}$, então a matriz de regressão $\mathbf{D}_i \in \mathbb{R}^{n_i \times (p+q)}$ pode ser definida como

$$\mathbf{D}_i = \begin{pmatrix} (\mathbf{X}_i^{in})^T \\ \left(\mathbf{X}_{j_1^{(i)}}^{in}\right)^T \\ \vdots \\ \left(\mathbf{X}_{j_K^{(i)}}^{in}\right)^T \end{pmatrix}_{n_i \times (p+q)}, \tag{5.15}$$

onde o sobrescrito T denota o operador transposto de uma matriz.

Finalmente, o vetor de coeficientes do i-ésimo neurônio, $\mathbf{a}_i \in \mathbb{R}^{p+q}$, é calculado usando o método dos mínimos quadrados como

$$\mathbf{a}_i = \left(\mathbf{D}_i^T \mathbf{D}_i + \lambda \mathbf{I}\right)^{-1} \mathbf{D}_i^T \mathbf{d}_i^{out}, \tag{5.16}$$

onde $\mathbf{I}$ é a matriz identidade de ordem $(p+q) \times (p+q)$ e $0 < \lambda \ll 1$ (e.g. $\lambda = 0.001$) é uma constante de regularização.

A Figura 5.3 traz uma ilustração de caráter didático da região (em cinza) formada pela união da célula de Voronoi do i-ésimo neurônio com as células de Voronoi dos seus K vizinhos mais próximos. Os centróides das células de Voronoi que compõem a região

[1] Ou seja, número de elementos.

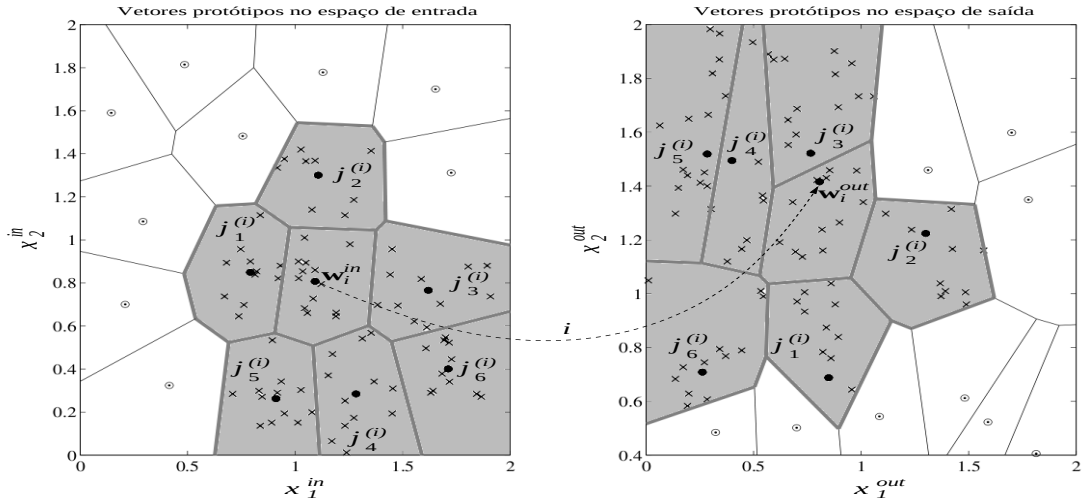


Figura 5.3 Representação ilustrativa dos dados (simbolizados por "×") que pertencem à região em cinza, formada pela união da célula de Voronoi do i-ésimo neurônio com as células de seus K vizinhos mais próximos, tanto para o espaço de entrada (figura à esquerda) quanto para o de saída (figura à direita).

acinzentada estão representados por pontos pretos. Os pontos mapeados nesta região são representados pelo símbolo "×". Os pontos mostrados na figura da esquerda formam a matriz $\mathbf{D}_i$, mostrada na Equação (5.15). Os pontos mostrados na figura da direita formam o vetor $\mathbf{d}_i^{out}$, mostrado na Equação (5.14).

Uma vez que os modelos locais dos g neurônios tenham sido estimados, eles podem ser usados para prever a saída do mapeamento de interesse para novos vetores de entrada. Nesta tese, o interesse está no uso do modelo D-MKSOM para identificar o modelo inverso de uma planta não-linear. Isto pode ser feito por meio da seguinte equação:

$$\hat{u}(t) = \mathbf{a}_{i^*}^T \mathbf{x}^{in}(t), \tag{5.17}$$

em que o vetor de entrada é definido como na Equação (5.3) e o índice do neurônio vencedor $i^*(t)$ é determinado como na Equação (5.8). O erro de predição (resíduo) no instante t é definido como $e(t) = u(t) - \hat{u}(t)$. Para finalizar, um sumário do modelo D-MKSOM é apresentado na Tabela (5.2).

5.4 Resumo

Os dois algoritmos vistos neste capítulo, P-MKSOM e D-MKSOM, são extensões do modelo KSOM, mas com algumas diferenças em relação ao último. Para o modelo KSOM, só há um vetor de coeficientes, que é re-estimado sempre que um novo dado de entrada

é apresentado à rede. Este vetor de coeficientes será estimado e usado apenas naquele instante, enquanto que no instante seguinte será estimado um outro vetor, e assim por diante. Isto ocorre porque o conjunto de K vetores-protótipos que são mais próximos do vetor de entrada no instante atual não necessariamente serão os mesmos no momento seguinte.

No caso dos modelos P-MKSOM e D-MKSOM, cada neurônio tem o seu próprio vetor de coeficientes representando o seu respectivo modelo linear local. Após o encerramento do treinamento, estes vetores são "congelados" e são utilizados apenas quando seu neurônio for o vencedor para um certo vetor de entrada. A diferença entre os modelos P-MKSOM e D-MKSOM reside na forma como os vetores de coeficientes de seus modelos locais são estimados. Para o modelo P-MKSOM, os modelos locais são construídos usando os vetores-protótipos do i-ésimo neurônio e de seus K vizinhos mais próximos. Para o modelo D-MKSOM, uma etapa a mais é adicionada, de forma a selecionar os dados que pertencem à subregião formada pela união das células de Voronoi do i-ésimo neurônio e de seus K vizinhos mais próximos. O modelo linear local do i-simo neurônio é então construído usando estes dados, em vez de seus vetores protótipos.

Os modelos D-MKSOM e P-MKSOM se somam aos demais modelos apresentados no Capítulo 4 na aplicação da modelagem inversa de plantas industriais, cujos os resultados serão apresentados no próximo capítulo.

Tabela 5.2 Algoritmo de treinamento e teste do modelo D-MKSOM.

Modelo D-MKSOM

Constantes	**Valor típico**
α : Taxa de aprendizagem	$0.001 < \alpha < 0.5$
σ : Abertura da vizinhança topológica	$0.001 < \sigma < (g/2)$
λ : Constante de regularização	$\lambda = 0.001$

Entradas

$\mathbf{x}^{in}(t)$: vetor de entrada, dimensão $(p+q) \times 1$ — $x^{out}(t)$: escalar de saída

Algoritmo

1. Inicialização $(t = 0)$

$\mathbf{w}_i^{in}, w_i^{out} \sim U(0,1)$, $i = 1, 2, \ldots, g$

2. Aprendizagem do modelo $(t = 1, 2, \ldots, N_1)$

2.1 Busca pelo neurônio vencedor:

$i^*(t) = \text{argmin}_i \|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|$.

2.2 Atualização dos vetores-protótipos de entrada e escalares de saída do neurônio i:

$$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_{i^*} - \mathbf{r}_i\|^2}{2\sigma^2(t)}\right),$$

$$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha(t) h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)],$$

$$w_i^{out}(t+1) = w_i^{out}(t) + \alpha(t) h(i^*, i; t)[x^{out}(t) - w_i^{out}(t)].$$

3. Estimação do vetor de coeficientes do neurônio *i*

3.1 Determinação dos índices dos K vetores-protótipos mais próximos ao vetor $\mathbf{w}_i^{in}$:

$\mathcal{J}_i = i \cup \{j_k^{(i)}\}_{k=1}^{K}$.

3.2 Obtenção da matriz $\mathbf{D}_i$ e do vetor $\mathbf{d}_i^{out}$:

$$\mathbf{d}_i^{out} = \begin{bmatrix} \mathbf{x}_i^{out} & \mathbf{x}_{j_1^{(i)}}^{out} & \cdots & \mathbf{x}_{j_K^{(i)}}^{out} \end{bmatrix}^T_{n_i \times 1},$$

$$\mathbf{D}_i = \begin{pmatrix} (\mathbf{X}_i^{in})^T \\ \left(\mathbf{X}_{j_1^{(i)}}^{in}\right)^T \\ \vdots \\ \left(\mathbf{X}_{j_K^{(i)}}^{in}\right)^T \end{pmatrix}_{n_i \times (p+q)}.$$

3.3 Cálculo do vetor de coeficientes associado a $\mathbf{w}_i^{in}$:

$\mathbf{a}_i = \left(\mathbf{D}_i^T \mathbf{D}_i + \lambda \mathbf{I}\right)^{-1} \mathbf{D}_i^T \mathbf{d}_i^{out}$.

4. Saída do modelo $(t = 1, 2, \ldots, N_2)$

4.1 Busca pelo neurônio vencedor:

$i^*(t) = \text{argmin}_i \|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|$.

4.2 Cálculo da saída estimada: $\hat{u}(t) = \mathbf{a}_{i^*}^T \mathbf{x}^{in}(t)$.

3.4 Cálculo do erro: $e(t) = u(t) - \hat{u}(t)$.

Saídas

$\hat{u}(t)$: saída estimada, $e(t)$: erro

6 Resultados das Simulações

Este capítulo se destina à apresentação e discussão dos resultados obtidos a partir da simulação dos modelos propostos nesta tese aplicados à tarefa de identificação inversa de sistemas dinâmicos. Os modelos serão avaliados usando os dados obtidos de quatro tipos de processos não lineares, que estão disponíveis na Internet em sítios destinados ao tema de identificação de sistemas. Os resultados obtidos pelos modelos LLM, KSOM e suas extensões (P-MKSOM e D-MKSOM) serão comparados com os resultados dos modelos globais não-lineares baseados nas redes MLP e ELM, e também com o modelo global linear ARX/LMS. Por último, o modelo local fuzzy Takagi-Sugeno será também utilizado nesta análise comparativa com os modelos locais propostos.

A validação será feita basicamente da análise dos resíduos (erros de generalização) para uma grande quantidade de realizações de treinamento-teste. Para este fim, análises quantitativas (e.g. erro médio quadrático e variância do erro) e qualitativas (e.g. boxplots, funções de autocorrelação linear e não-linear) feitas sobre a distribuição dos resíduos ajudarão a escolher os modelos que melhor se ajustam aos dados de interesse.

Por fim, um estudo abrangente da influência do algoritmo de quantização vetorial no desempenho dos modelos locais, tanto os propostos como os baseados no modelo Takagi-Sugeno, é apresentado neste capítulo. Esta avaliação terá como base testes estatísticos baseados no teste de Kolmogorov-Smirnov que compara a distribuição acumulada dos resíduos de um mesmo modelo implementado por diferentes algoritmos de quantização vetorial. O teste de Kolmogorov-Smirnov busca averiguar se há ou não diferenças significativas do ponto de vista estatístico entre as diferentes implementações de um dado modelo.

A seguir são descritos os conjuntos de dados utilizados nesta tese.

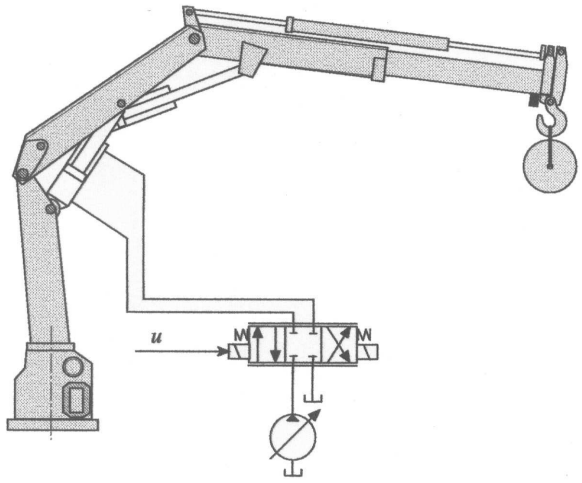


Figura 6.1 Esboço do atuador hidráulico cujos dados serão utilizados nesta tese para avaliação dos modelos lineares locais propostos.

6.1 Conjuntos de Dados Utilizados

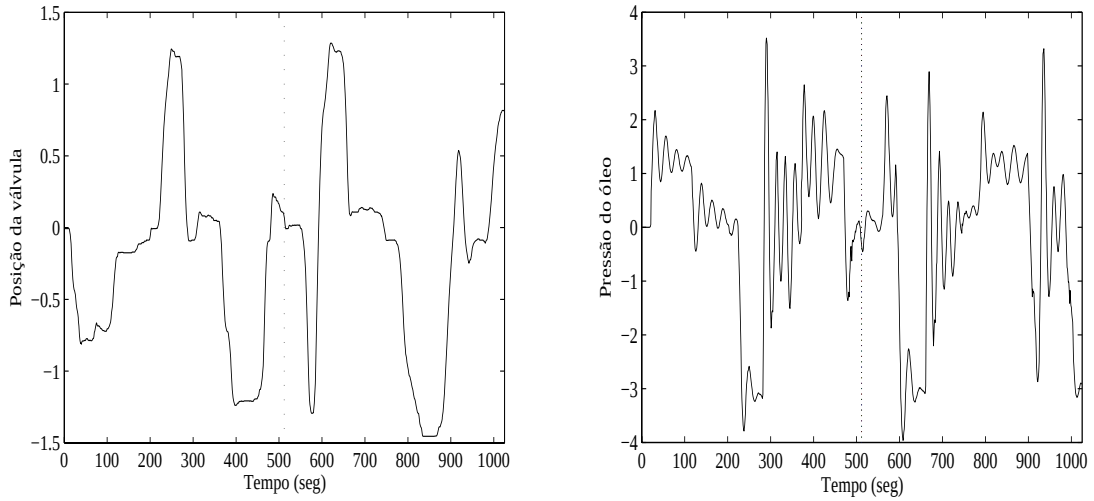
Esta seção traz breve explanações sobre cada conjunto de dados a serem utilizados nesta tese e que estão disponíveis para baixar do repositório ***DaISy***(*Database for the Identification of Systems*)[1]. Dois casos correspondem a sistemas industriais, enquanto dois outros correspondem a sistema mecânicos.

6.1.1 Atuador Hidráulico

Para primeiro estudo de caso, escolheu-se a tarefa de identificação de um atuador de uma estrutura mecânica hidráulica (grua), mostrada na Figura 6.1. Este equipamento possui 4 atuadores: um para a rotação de toda a estrutura, um para mover o braço, um para mover o antebraço e um para mover uma extensão telescópica do antebraço. No jargão de robótica, uma grua pode ser vista como um robô com três juntas rotacionais e uma prismática. Esta planta foi escolhida porque possui um longo braço e um longo antebraço com considerável flexibilidade na estrutura mecânica, tornando o movimento de toda a grua muito oscilatório e difícil de controlar. Este estudo de caso se concentra apenas na identificação do atuador do braço (em destaque na figura), visto que este é o mais afetado pelo comportamento oscilatório da estrutura. Os resultados da identificação deste atuador serão apresentados na próxima seção.

As séries temporais dos dados provenientes do atuador hidráulico são mostradas na

[1] http://homes.esat.kuleuven.be/ smc/daisy/daisydata.html



(a) (b)

Figura 6.2 Valores medidos da posição da válvula (a) e pressão do óleo (b).

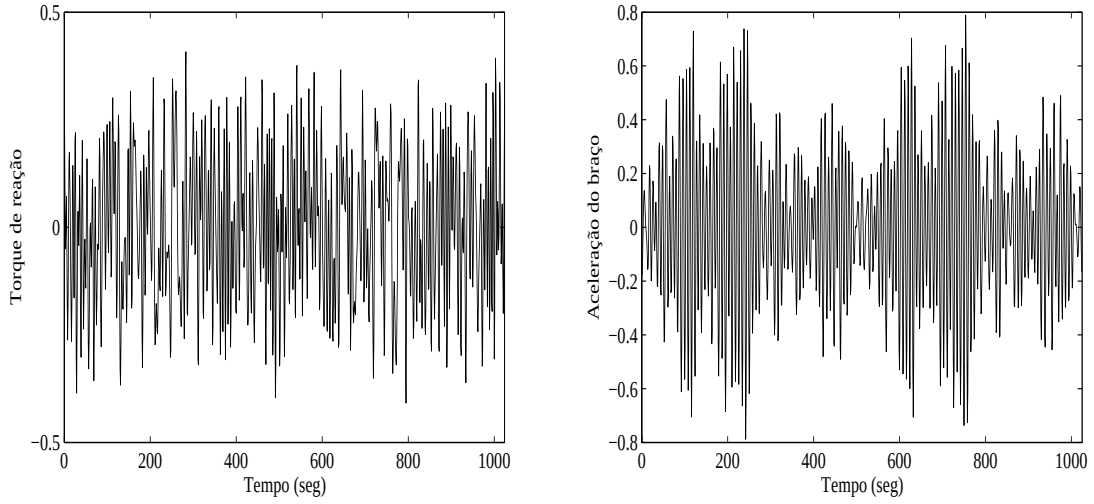
Figura 6.2, em que cada conjunto de dados corresponde à posição da válvula (série temporal de entrada, $\{u(t)\}$) e à pressão do óleo (série temporal de saída, $\{y(t)\}$). A sequência de sinal da pressão do óleo mostra um procedimento altamente oscilante causado por ressonâncias mecânicas (SJÖBERG et al., 1995).

6.1.2 Braço Robótico

Este segundo estudo de caso representa o funcionamento de um braço robótico flexível. O braço está acoplado à um motor elétrico. Foi modelado por função de transferência do torque de reação medido da estrutura para aceleração do braço flexível. As séries temporais dos dados provenientes do braço robótico flexível são mostradas na Figura 6.3, em que cada conjunto de dados corresponde ao torque de reação da estrutura (série temporal de entrada, $\{u(t)\}$) e à aceleração do braço robótico (série temporal de saída, $\{y(t)\}$).

6.1.3 Trocador de Calor

O terceiro conjunto de dados é de um trocador de calor de vapor de líquido saturado (BITTANTI; PIRODDI, 1997), onde a água é aquecida pelo vapor saturado pressurizado através de um tubo de cobre. A principal motivação para escolha do trocador de



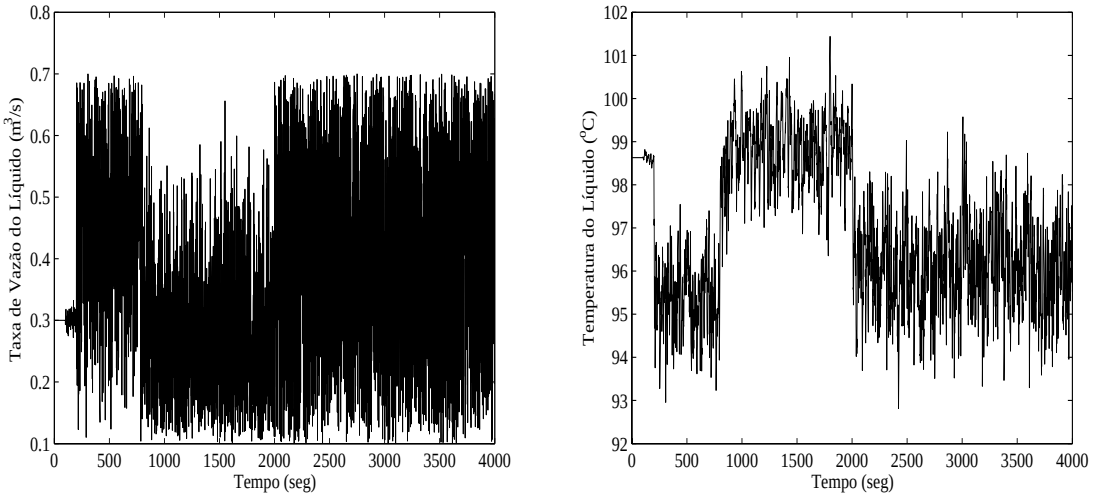
(a) (b)

Figura 6.3 Valores medidos do torque de reação (a) e aceleração do braço (b).

calor como um modelo significante para simulação é que esta planta é caracterizada por um comportamento de fase não-mínima. A Figura 6.4 mostra os valores medidos da taxa de vazão de líquido (série temporal de entrada, $\{u(t)\}$, m^3/s) e a temperatura de saída do líquido (série temporal de saída, $\{y(t)\}$, ^oC) e o tempo de amostragem é 1s.

6.1.4 Reator Tanque de Agitação Contínua

O último conjunto de dados usado é proveniente de um reator de um tanque de agitação contínuo (*Continuous Stirred Tank Reactor*, CSTR) (LIGHTBODY; IRWIN, 1997), também conhecido como reator vat- ou backmix, sendo um tipo de reator ideal comum na engenharia química. A reação no seu interior é exotérmica e a concentração é controlada pela regulação da corrente de resfriamento. Este reator possui uma agitação e seu interior permanece com composição uniforme. Existem muitas aplicações voltadas para o uso de redes neurais em modelagem inversa de reatores, inclusive com sistemas experimentais tal como um reator simulado parcialmente (HUSSAIN; KERSHENBAUM, 2000), projetado para testar o uso de tais algoritmos não-lineares. A Figura 6.5 mostra os valores medidos da corrente de resfriamento (série temporal de entrada, $\{u(t)\}$, l/min) e a concentração (série temporal de saída, $\{y(t)\}$, mol/l), e a taxa de amostragem é 0.1min.



(a) (b)

Figura 6.4 Valores medidos da taxa de vazão do líquido (a) e temperatura do líquido (b).

6.2 Análise do Erro de Generalização

Nesta seção comenta-se o método de avaliação dos modelos neurais usando o critério do erro de generalização adotado para cada conjunto de dados. Os resultados obtidos serão apresentados separadamente conforme os conjuntos de dados empregados para simulação.

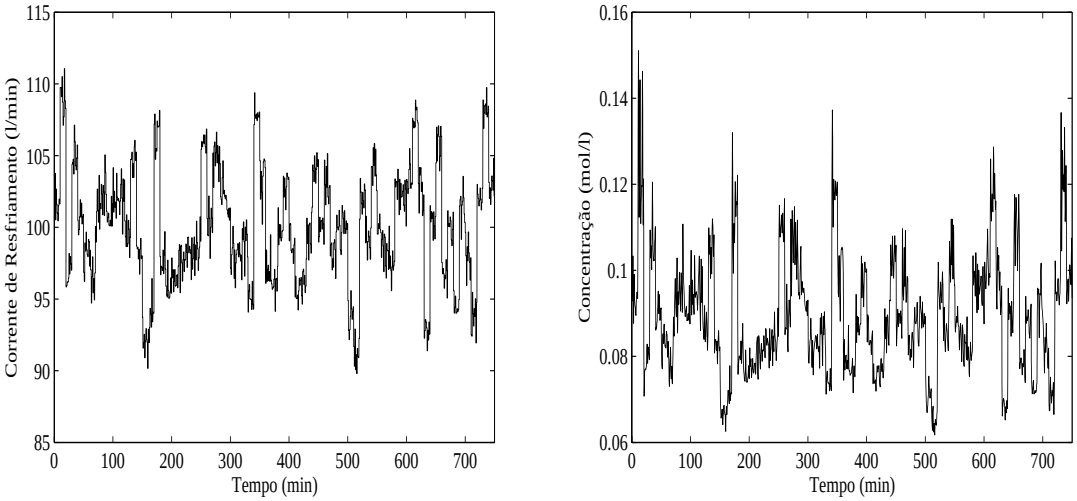
Durante a fase de teste, para propósitos de avaliação, cada modelo produz uma sequência de erros de predição (resíduos) $\{e(t)\}_{t=1}^{N_2}$, em que $e(t) = u(t) - \hat{u}(t)$, com $u(t)$ denotando a saída desejada de cada modelo neural e $\hat{u}(t)$ denotando a estimação fornecida por cada modelo neural. Para avaliação quantitativa do desempenho dos modelos neurais é usado o erro quadrático médio normalizado (*Normalized Mean Squared Error*, NMSE):

$$NMSE = \frac{\sum_{t=1}^{N_2} e^2(t)}{N_2 \cdot \hat{\sigma}_u^2} \tag{6.1}$$

sendo $\hat{\sigma}_u^2$ a variância da série temporal original $\{u(t)\}_{t=1}^{N_2}$ e N_2 é o comprimento da sequência de resíduos, ou seja, a quantidade de dados existente no conjunto de teste.

O modelo KSOM e suas extensões (P-MKSOM e D-MKSOM) são comparados com modelos globais baseados nas redes MLP e ELM, bem como com o modelo LLM. Para a rede MLP, em particular, são implementadas as seguintes topologias:

(1) MLP-1h: modelo com uma camada oculta treinada pelo algoritmo padrão de retro-



(a) (b)

Figura 6.5 Valores medidos da corrente de resfriamento (a) e concentração (b).

propagação do erro;

(2) MLP-LM: modelo com uma camada oculta treinada pelo algoritmo de segunda-ordem Levenberg-Marquardt;

(3) MLP-2h: modelo com duas camadas ocultas treinada pelo algoritmo padrão da retropropagação do erro.

Todos os modelos neurais, locais e globais, são também comparados com o modelo linear ARX, treinado por meio do algoritmo LMS.

6.2.1 Resultados - Atuador Hidráulico

Para este conjunto de dados, os modelos foram treinados por uma época apenas, usando as primeiras 512 amostras das sequências de entrada/saída ($N_1 = 512$) e testado com as 512 amostras restantes ($N_2 = 512$). Os valores referentes às ordens de memória dos dados de entrada e dados de saída são iguais a $q = 4$ e $p = 5$, respectivamente. Os valores aqui usados para as ordens de memória entrada/saída foram aqueles sugeridos por trabalhos anteriores (SOUZA; BARRETO, 2006; BARRETO; SOUZA, 2006). Os demais parâmetros de treinamento utilizados pelos modelos neurais estão dispostos na Tabela 6.1.

Inicialmente, para a rede MLP, foi feita uma busca exaustiva para obtenção do valor

Tabela 6.1 Parâmetros de treinamento dos modelos neurais (atuador hidráulico).

Parâmetros de Treinamento								
Modelos Usados	h_1	h_2	g	K	g^f	$(\alpha_0);(\alpha_T)$	α'	$(\sigma_0);(\sigma_f)$
MLP-1h	20	-	-	-	-	(0.1);(0.1)	-	-
MLP-LM	20	-	-	-	-	(0.1);(0.1)	-	-
MLP-2h	20	10	-	-	-	(0.1);(0.1)	-	-
ELM	20	-	-	-	-	-	-	-
ARX	-	-	1	-	-	-	0.01	-
LLM	-	-	20	-	-	(0.5);(0.01)	0.01	$(g/2)$;(0.001)
KSOM	-	-	20	15	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
P-MKSOM	-	-	20	15	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
D-MKSOM	-	-	20	15	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
fuzzy TS	-	-	-	-	20	(0.5);(0.01)	-	$(g^f/2)$;(0.001)

ótimo do número de neurônios ocultos que conduzisse ao menor error de generalização, com os valores de h_1 variando de 2 até 30. Para cada valor de h_1, 100 rodadas de treinamento/teste independentes foram realizadas. A cada rodada os pesos sinápticos das redes foram inicializados aleatoriamente no intervalo [0,1]. Para rede MLP-2h, o número de neurônios ocultos na segunda camada é definido heuristicamente como sendo metade do número de neurônios na primeira camada oculta. Para todos os modelos globais baseados na rede MLP, a função de ativação dos neurônios da camada oculta é a tangente hiperbólica, enquanto o neurônio da camada de saída usa uma função linear. Nenhum termo de momento é usado.

O mesmo procedimento para obtenção do valor ótimo do número de neurônios ocultos foi utilizado também para a arquitetura ELM, buscando o valor ótimo para h_1 no intervalo de 2 a 30. Para rede ELM, foram definidas as mesmas funções de ativação usadas na rede MLP, tanto para os neurônios ocultos quanto para o neurônio de saída. As melhores configurações obtidas para as duas arquiteturas estão dispostas na Tabela 6.1.

Os parâmetros utilizados na etapa de treinamento dos modelos baseados na rede SOM estão discriminados na Tabela 6.1, em que se faz referência aos valores das taxas de aprendizagem inicial e final, os valores dos raios da função vizinhança e, também, o número de neurônios utilizados por cada modelo. Para o modelo LLM, a taxa de aprendizagem dos vetores de coeficientes adotada (α') é constante. Para o modelo KSOM e extensões, também encontram-se nesta tabela os melhores valores obtidos para o parâmetro K.

Para o modelo local fuzzy Takagi-Sugeno (TS), foram adotados os mesmos parâmetros de treinamento apresentados pelos modelos baseados na rede SOM e que são vistos na Tabela 6.1. A nível de comparação entre os resultados obtidos pelos modelos locais,

Tabela 6.2 Desempenho dos modelos locais e globais usando os dados do atuador hidráulico.

Modelos Usados	**NMSE**			
	média	*min*	*max*	*variância*
D-MKSOM	1.26e-004	1.21e-004	1.38e-004	1.07e-011
P-MKSOM	5.82e-004	3.63e-004	0.0010	1.83e-008
fuzzy TS	6.92e-004	3.37e-004	0.0015	5.77e-008
ELM	0.0012	0.0001	0.0026	1.04e-007
KSOM	0.0019	0.0002	0.0247	1.15e-005
LLM	0.0347	0.0181	0.0651	1.58e-004
ARX	0.0380	0.0380	0.0380	0.0083
MLP-LM	0.0722	0.0048	0.3079	0.0041
MLP-1h	0.3485	0.2800	0.4146	4.96e-004
MLP-2h	0.3516	0.0980	2.6986	0.0963

buscou-se então implementar o modelo fuzzy TS usando como algoritmo de quantização vetorial a SOM. Esta escolha é fundamentada no tipo de algoritmo adotado pelos modelos locais propostos que é exatamente o mesmo.

Os resultados obtidos das simulações são apresentados na Tabela 6.2, onde estão mostrados os valores médio, mínimo, máximo e a variância do NMSE, coletados sobre 100 rodadas de treinamento/teste, com os pesos sinápticos dos modelos neurais sendo inicializados aleatoriamente no intervalo [0,1] a cada rodada. Uma rodada de treinamento/teste equivale a uma única passagem dos dados (i.e. uma época) na etapa de treinamento, e logo em seguida, a passagem dos dados restantes para o teste dos modelos. Nesta tabela, os modelos são organizados em ordem crescente dos valores médios do NMSE.

Nota-se que os desempenhos dos modelos D-MKSOM e P-MKSOM para este conjunto de dados são superiores aos demais modelos, com o modelo D-MKSOM apresentando o melhor desempenho. O modelo fuzzy TS apresentou também desempenho que pode ser considerado satisfatório, bem a frente do melhor modelo global que foi o ELM. Vale ressaltar que dos quatro melhores resultados listados na Tabela 6.2, três são produzidos por modelos locais.

Ainda sobre os resultados da Tabela 6.2, vê-se que a rede ELM obteve um desempenho melhor do que os dos três modelos globais baseados na rede MLP. Isto pode ser explicado pelo fato de que o treinamento da rede MLP por apenas uma época usando o algoritmo de retropropagação padrão (i.e. baseado no gradiente descendente) não é adequado. Em função disso, o modelo ARX também apresentou melhor resultado que as redes MLP. Entre os modelos baseados na rede MLP, o uso da informação de segunda ordem (e.g. matriz hessiana) poderia também explicar o melhor desempenho da rede

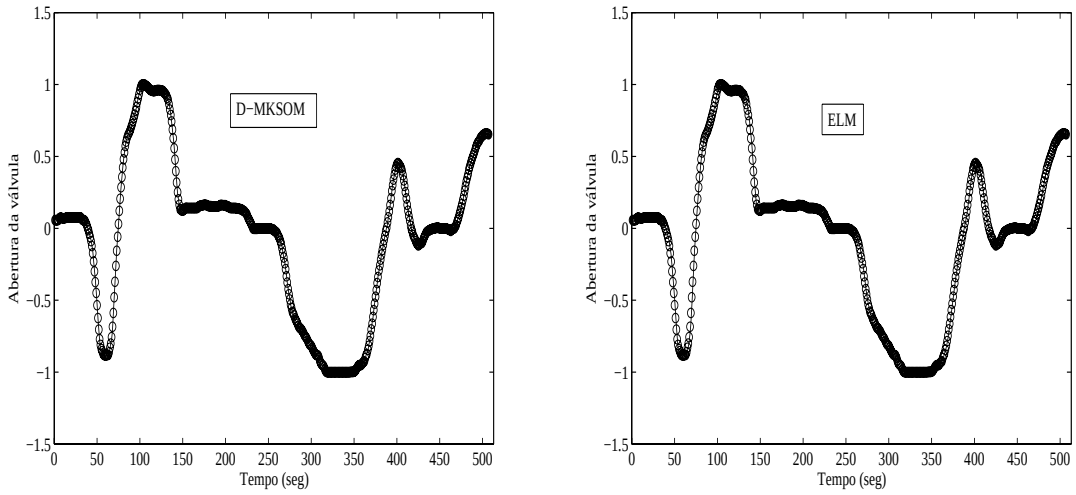


Figura 6.6 Sequências de valores previstos para a abertura da válvula fornecida pelos melhores modelos. Os pontos 'o' denotam valores reais e a linha sólida, a sequência prevista.

MLP-LM. A Figura 6.6 mostra sequências típicas dos valores estimados da abertura da válvula fornecidas pelo melhor modelo local (D-MKSOM) e pelo melhor modelo global (ELM). Os gráficos demonstram que os resultados previstos pelos modelos se aproximam satisfatoriamente aos valores reais do sistema não linear.

Para finalizar, avalia-se a influência do algoritmo de quantização vetorial (QV) sobre os desempenhos dos modelos locais D-MKSOM, P-MKSOM e fuzzy TS para o atuador hidráulico. Os resultados obtidos estão na Tabela 6.3, em que os mesmos são construídos usando os algoritmos K-médias, WTA, FSCL e FCL. Estes resultados são interessantes de serem mostrados pois demonstram como os modelos se comportam usando outros tipos de quantizadores vetoriais, abrindo espaço para uma análise futura do grau de similaridade entre eles.

Ao analisar esta tabela, observa-se que o modelo P-MKSOM, quando treinado pelo algoritmo SOM, apresenta desempenho similar aquele treinado pelo algoritmo K-médias, com ligeira vantagem para o algoritmo K-médias. Já para o modelo D-MKSOM, a utilização do algoritmo SOM conduz a resultados equivalentes ao do algoritmo FCL, com pequena vantagem para o algoritmo SOM (em termos da média e da variância). No caso do modelo fuzzy TS, o que se vê é o melhor desempenho apresentado pelo algoritmo FCL, superando e muito o valor atingido pelo SOM. Comparando-se os resultados obtidos pelos modelos fuzzy TS e D-MKSOM, constata-se que no caso do FCL para fuzzy TS seu resultado se aproxima ao do correspondente implementado pelo D-MKSOM.

Tabela 6.3 Desempenhos dos modelos locais D-MKSOM, P-MKSOM e fuzzy TS para diferentes algoritmos de quantização vetorial (atuador hidráulico).

Modelo D-MKSOM				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
SOM	1.25e-004	1.20e-004	1.31e-004	5.65e-012
FCL	1.33e-004	1.19e-004	8.50e-004	6.80e-009
WTA	2.32e-004	1.16e-004	0.0013	7.50e-008
FSCL	2.52e-004	1.19e-004	0.0013	8.72e-008
K-médias	6.33e-004	1.21e-004	0.0050	6.37e-007
Modelo P-MKSOM				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
K-médias	5.55e-004	2.76e-004	8.79e-004	1.48e-008
SOM	5.82e-004	3.63e-004	0.0010	1.83e-008
WTA	0.0193	0.0058	0.0379	4.69e-005
FSCL	0.0272	0.0128	0.0473	6.64e-005
FCL	0.0277	0.0163	0.0611	6.86e-005
Modelo fuzzy TS				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
FCL	1.39e-004	1.33e-004	1.46e-004	5.10e-012
WTA	1.66e-004	1.65e-004	1.68e-004	5.44e-013
FSCL	1.80e-004	1.78e-004	1.82e-004	1.24e-012
K-means	4.29e-004	4.27e-004	4.33e-004	2.31e-012
SOM	6.92e-004	3.37e-004	0.0015	5.77e-008

6.2.2 Resultados - Braço Robótico

Para este conjunto de dados, todos os parâmetros de treinamento utilizados pelos modelos neurais e fuzzy estão dispostos na Tabela 6.4. Os neurônios ocultos dos modelos globais baseados nas redes MLP e ELM usam função de ativação tangente hiperbólica, enquanto o neurônio de saída é linear. Os modelos são treinados usando as primeiras 820 amostras das sequências de sinal entrada/saída (aproximadamente, 80% dos dados) e testados com as 204 amostras restantes. Os valores atribuídos às ordens de memória dos sinais de entrada e de saída são iguais a $q = 4$ e $p = 5$, respectivamente.

Da mesma forma que foi adotada na simulação anterior, o modelo fuzzy TS será implementado com o uso do algoritmo de quantização vetorial SOM, e são listados os valores dos parâmetros de treinamento adotados para o modelo na tabela citada. Os resultados obtidos após o término de 100 rodadas de treinamento/teste são mostrados na Tabela 6.5, em ordem crescente dos valores médios do NMSE.

Tabela 6.4 Parâmetros de treinamento dos modelos neurais (braço robótico).

Parâmetros de Treinamento								
Modelos Usados	h_1	h_2	g	K	g^f	$(\alpha_0);(\alpha_T)$	α'	$(\sigma_0);(\sigma_f)$
MLP-1h	30	-	-	-	-	(0.1);(0.1)	-	-
MLP-LM	30	-	-	-	-	(0.1);(0.1)	-	-
MLP-2h	30	15	-	-	-	(0.1);(0.1)	-	-
ELM	30	-	-	-	-	-	-	-
ARX	-	-	1	-	-	-	0.01	-
LLM	-	-	30	-	-	(0.5);(0.01)	0.01	$(g/2)$;(0.001)
KSOM	-	-	30	25	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
P-MKSOM	-	-	30	25	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
D-MKSOM	-	-	30	25	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
fuzzy TS	-	-	-	-	30	(0.5);(0.01)	-	$(g^f/2)$;(0.001)

Tabela 6.5 Desempenho dos modelos locais e globais usando os dados do braço robótico.

Modelos Usados	**NMSE**			
	média	*min*	*max*	*variância*
D-MKSOM	0.0046	0.0045	0.0047	9.89e-010
KSOM	0.0064	0.0045	0.0117	1.83e-006
fuzzy TS	0.0066	0.0055	0.0079	2.32e-007
P-MKSOM	0.0203	0.0107	0.0313	1.90e-005
ELM	0.0285	0.0171	0.0457	2.73e-005
MLP-LM	0.1488	0.0657	0.4936	0.0107
MLP-1h	0.1622	0.1549	0.1699	1.03e-005
LLM	0.3176	0.2685	0.3558	2.23e-004
ARX	0.3848	0.3848	0.3848	0.0445
MLP-2h	0.6963	0.5978	1.5310	0.0368

Com base nos resultados da Tabela 6.5, vê-se que o modelo D-MKSOM destaca-se como o melhor dentre todos, seguido de perto pelo modelo KSOM. Logo em seguida vem o modelo fuzzy TS com o valor bem próximo ao KSOM, sendo praticamente igual. Entre os modelos globais, o modelo ELM alcançou o melhor desempenho, mas estando atrás do P-MKSOM. Os resultados demonstram claramente que os quatro melhores modelos listados nesta tabela são todos modelos locais. O desempenho atingido por outra arquitetura de modelo linear local, tal como a arquitetura LLM, superou somente os dos modelos ARX e MLP-2h. Os resultados destes dois foram os piores apresentados.

A Figura 6.7 mostra sequências típicas dos valores estimados do torque de reação da estrutura fornecidos pelo melhor modelo local (D-MKSOM) e pelo melhor modelo global (ELM). Os gráficos demonstram quanto os resultados previstos pelos modelos se aproximam satisfatoriamente aos valores reais do sistema não linear.

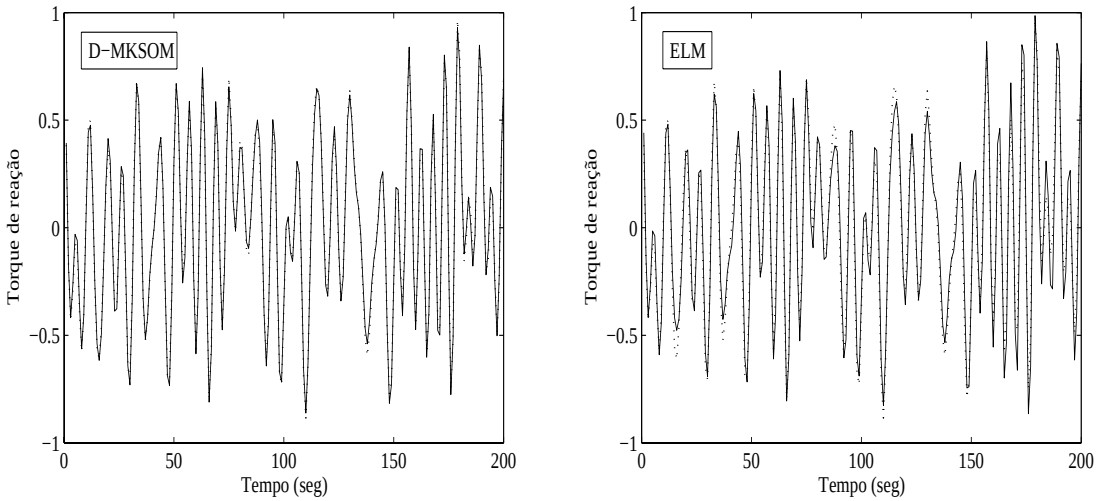


Figura 6.7 Seqüências de valores preditos do torque de reação da estrutura fornecida para os melhores modelos. As linhas tracejadas denotam valores reais e a linha sólida, a sequência predita.

Na Tabela 6.6 são apresentados os resultados obtidos para simulação que têm por objetivo avaliar a influência do algoritmo de quantização vetorial no desempenho dos modelos D-MKSOM, P-MKSOM e fuzzy TS. À primeira vista verifica-se um resultado interessante que ocorre com o modelo D-MKSOM. Todos os modelos implementados com os demais algoritmos de quantização vetorial apresentaram o mesmo desempenho do modelo D-MKSOM original, indicando que este modelo é menos dependente dos algoritmos usados. Neste caso, recomenda-se usar o algoritmo de quantização vetorial de menor custo computacional no projeto do modelo D-MKSOM.

Uma outra análise dessa tabela permite verificar que o uso dos algoritmos SOM e K-médias resultam em desempenhos equivalentes no modelo P-MKSOM. Neste caso, sugere-se o uso do algoritmo K-médias no projeto do modelo por ter custo computacional menor que o da rede SOM. Para os valores apresentados pelo modelo fuzzy TS o que se vê são resultados bem próximos entre si, principalmente aqueles apresentados pelos modelos, FCL e WTA, em que a diferença é bem pequena, e na hipótese de escolha entre eles, a preferência estar no que apresenta menor custo computacional, no caso o WTA. Por último, um detalhe importante é a proximidade dos resultados obtidos pelo D-MKSOM e o fuzzy TS.

Tabela 6.6 Desempenho dos modelos D-MKSOM, P-MKSOM e fuzzy TS para diferentes algoritmos de quantização vetorial (braço robótico).

Modelo D-MKSOM				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
SOM	0.0046	0.0046	0.0047	7.33e-010
K-médias	0.0046	0.0046	0.0047	1.00e-010
WTA	0.0046	0.0046	0.0047	1.62e-010
FSCL	0.0046	0.0046	0.0047	7.39e-011
FCL	0.0046	0.0046	0.0047	3.34e-010
Modelo P-MKSOM				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
K-médias	0.0196	0.0137	0.0298	2.79e-005
SOM	0.0198	0.0124	0.0325	2.10e-005
WTA	0.0987	0.0419	0.1515	7.22e-004
FSCL	0.3233	0.2321	0.4636	0.0025
FCL	0.7175	0.3930	1.1719	0.0244
Modelo fuzzy TS				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
FCL	0.0048	0.0047	0.0048	7.19e-011
WTA	0.0049	0.0048	0.0050	1.96e-009
FSCL	0.0052	0.0051	0.0052	5.87e-010
K-means	0.0065	0.0060	0.0069	1.25e-007
SOM	0.0066	0.0055	0.0079	2.32e-007

6.2.3 Resultados - Trocador de Calor

Para este conjunto de dados, os modelos também são treinados por uma época apenas, usando as primeiras 3200 amostras das sequências de entrada/saída (aproximadamente, 80% do total) e testado com as 800 amostras restantes. As séries de entrada/saída estão normalizadas para o intervalo [-1,+1]. Os valores das ordens de memória dos sinais de entrada e de saída são iguais a $q = 3$ e $p = 6$, respectivamente. Depois de algumas experimentações com os dados, as melhores configurações obtidas para os modelos, bem como os parâmetros de treinamento utilizados estão mostrados na Tabela 6.7. Os neurônios ocultos dos modelos globais baseados nas redes MLP e ELM usam função de ativação tangente hiperbólica, enquanto o neurônio de saída é linear.

Para os modelos locais, os valores dos parâmetros utilizados para o treinamento de cada modelo estão listados na tabela citada, e novamente o modelo fuzzy TS é implementado com o algoritmo de quantização vetorial SOM. A importância de utilizar este

Tabela 6.7 Parâmetros de treinamento dos modelos neurais (trocador de calor).

Parâmetros de Treinamento								
Modelos Usados	h_1	h_2	g	K	g^f	$(\alpha_0);(\alpha_T)$	α'	$(\sigma_0);(\sigma_f)$
MLP-1h	20	-	-	-	-	(0.1);(0.1)	-	-
MLP-LM	20	-	-	-	-	(0.1);(0.1)	-	-
MLP-2h	20	10	-	-	-	(0.1);(0.1)	-	-
ELM	20	-	-	-	-	-	-	-
ARX	-	-	1	-	-	-	0.01	-
LLM	-	-	30	-	-	(0.5);(0.01)	0.01	$(g/2)$;(0.001)
KSOM	-	-	30	20	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
P-MKSOM	-	-	30	20	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
D-MKSOM	-	-	30	20	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
fuzzy TS	-	-	-	-	30	(0.5);(0.01)	-	$(g^f/2)$;(0.001)

Tabela 6.8 Desempenho dos modelos locais e globais usando os dados do trocador de calor.

Modelos Usados	**NMSE**			
	média	*min*	*max*	*variância*
fuzzy TS	0.2122	0.1917	0.2351	8.36e-005
ELM	0.3346	0.3316	0.3391	2.22e-006
D-MKSOM	0.3798	0.3657	0.3918	2.15e-005
MLP-1h	0.4292	0.4173	0.4501	4.02e-005
P-MKSOM	0.5640	0.4137	0.8118	0.0109
MLP-LM	0.5672	0.2140	0.6016	0.0048
KSOM	0.5841	0.4139	1.3877	0.0205
LLM	0.7837	0.7378	0.8702	5.88e-004
ARX	0.9257	0.9257	0.9257	5.28e-014
MLP-2h	1.3003	1.2207	1.5466	0.0059

quantizador vetorial na obtenção dos pontos focais do modelo fuzzy TS é a necessidade da comparação entre os modelos adotando critérios semelhantes.

Os resultados obtidos são mostrados na Tabela 6.8, onde são apresentados a média, mínimo, máximo e variância do NMSE. Estes valores foram calculadas após 100 rodadas de treinamento/teste, em que os pesos dos modelos neurais foram inicializados aleatoriamente no intervalo [0,1] a cada rodada. Nesta tabela, os modelos são ordenados em ordem crescente (de cima pra baixo) do valor médio do NMSE.

Desta vez, o melhor desempenho foi atingido pelo modelo fuzzy TS. O segundo melhor resultado foi obtido pela ELM, sendo que a diferença entre os valores começa já na primeira casa decimal após a vírgula, seguido de perto pelo modelo D-MKSOM. É interessante notar que, dos quatro melhores modelos dois são globais e dois são locais. Também é digno de nota que as duas extensões do modelo KSOM propostos nesta tese (P-MKSOM

Tabela 6.9 Desempenho dos modelos D-MKSOM, P-MKSOM e fuzzy TS para diferentes algoritmos de quantização vetorial (trocador de calor).

Modelo D-MKSOM				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
SOM	0.3798	0.3657	0.3918	2.15e-005
FCL	0.4054	0.3770	0.4311	1.26e-004
WTA	0.4066	0.3941	0.4423	8.51e-005
FSCL	0.4082	0.3969	0.4304	6.08e-005
K-médias	0.4083	0.4001	0.4170	1.17e-005
Modelo P-MKSOM				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
SOM	0.5640	0.4137	0.8118	0.0109
WTA	0.6026	0.5382	0.6713	0.0010
K-médias	0.6744	0.6503	0.6758	1.89e-005
FCL	1.4722	1.3263	1.7196	0.0069
FSCL	1.5817	1.2887	2.4085	0.1062
Modelo fuzzy TS				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
FCL	0.2000	0.1924	0.2113	1.32e-005
SOM	0.2122	0.1917	0.2351	8.36e-005
WTA	0.2188	0.1925	0.2241	4.32e-005
K-means	0.2222	0.2209	0.2324	1.29e-005
FSCL	0.2230	0.2197	0.2309	3.35e-006

e D-MKSOM) apresentaram desempenho superior ao do modelo KSOM.

Para finalizar, avalia-se a influência do algoritmo de quantização vetorial (QV) sobre os desempenhos dos modelos para o trocador de calor. Os resultados obtidos estão representados na Tabela 6.9. Para o modelo D-MKSOM, todos os quantizadores vetoriais utilizados levaram praticamente ao mesmo desempenho que a SOM, indicando que este modelo é menos dependente do algoritmo de quantização vetorial adotado. Neste caso, recomenda-se usar o algoritmo de menor custo computacional que a rede SOM no projeto do modelo D-MKSOM.

Ao analisar esta tabela, observa-se que o modelo P-MKSOM, quando treinado pelo algoritmo SOM, apresenta desempenho similar ao modelo que é treinado pelo algoritmo WTA, com ligeira vantagem para o algoritmo SOM em termos do valor médio. Neste caso, sugere-se o uso do algoritmo WTA no projeto do modelo P-MKSOM por ter custo computacional bem menor que o da rede SOM. Já no caso do modelo fuzzy TS, os re-

Tabela 6.10 Parâmetros de treinamento dos modelos neurais (CSTR).

Parâmetros de Treinamento								
Modelos Usados	h_1	h_2	g	K	g^f	$(\alpha_0);(\alpha_T)$	α'	$(\sigma_0);(\sigma_f)$
MLP-1h	20	-	-	-	-	(0.1);(0.1)	-	-
MLP-LM	20	-	-	-	-	(0.1);(0.1)	-	-
MLP-2h	20	10	-	-	-	(0.1);(0.1)	-	-
ELM	20	-	-	-	-	-	-	-
ARX	-	-	1	-	-	-	0.01	-
LLM	-	-	30	-	-	(0.5);(0.01)	0.01	$(g/2)$;(0.001)
KSOM	-	-	30	20	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
P-MKSOM	-	-	30	20	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
D-MKSOM	-	-	30	20	-	(0.5);(0.01)	-	$(g/2)$;(0.001)
fuzzy TS	-	-	-	-	30	(0.5);(0.01)	-	$(g^f/2)$;(0.001)

sultados obtidos também foram próximos entre si, permitindo chegar a uma escolha do melhor modelo, que neste caso pode ser a SOM, pois este algoritmo possui um menor custo computacional que o FCL.

6.2.4 Resultados - CSTR

Para este conjunto de dados, os modelos novamente são treinados por uma época apenas, usando as primeiras 600 amostras das sequências de entrada/saída (aproximadamente, 80% do total), e testado com as 150 amostras restantes. As séries de entrada/saída são normalizadas para o intervalo [-1,+1]. Os valores das ordens de memória dos sinais de entrada e de saída são iguais a $q = 3$ e $p = 4$, respectivamente. Os neurônios ocultos dos modelos globais baseados nas redes MLP e ELM usam função de ativação tangente hiperbólica, enquanto o neurônio de saída é linear.

Os parâmetros adotados para o treinamento das redes neurais estão listados na Tabela 6.10. Como visto nos três conjuntos de dados anteriores, o modelo fuzzy TS é novamente implementado usando o algoritmo de quantização vetorial SOM. Esta escolha é feita em conformidade com os demais modelos locais baseados no KSOM para realizar comparações necessárias entre eles.

Os resultados obtidos são mostrados na Tabela 6.11 onde são apresentados a média, mínimo, máximo e variância do NMSE. Estes valores foram calculadas após 100 rodadas de treinamento/teste, em que os pesos dos modelos neurais foram aleatoriamente iniciados no intervalo [0,1] a cada rodada. Nesta tabela, os modelos são ordenados em ordem crescente (de cima pra baixo) do valor médio do NMSE.

Tabela 6.11 Desempenho dos modelos locais e globais usando os dados do CSTR.

Modelos Usados	**NMSE**			
	média	*min*	*max*	*variância*
ELM	0.0330	0.0330	0.0331	7.25e-010
D-MKSOM	0.0331	0.0330	0.0331	1.92e-010
P-MKSOM	0.0338	0.0333	0.0365	3.69e-007
fuzzy TS	0.0339	0.0338	0.0340	1.47e-009
KSOM	0.0345	0.0339	0.0366	2.09e-007
LLM	0.0409	0.0394	0.0426	9.51e-007
MLP-1h	0.0475	0.0455	0.0503	1.02e-006
ARX	0.0504	0.0504	0.0504	9.11e-027
MLP-LM	0.0863	0.0203	0.2882	0.0032
MLP-2h	0.3200	0.1686	0.8078	0.0142

Os modelos ELM e D-MKSOM alcançaram os melhores desempenhos. Na verdade, seus desempenhos foram estatisticamente iguais. O desempenho do modelo D-MKSOM foi ligeiramente melhor do que do modelo P-MKSOM. Os dois modelos propostos apresentaram resultados melhores do que o do modelo KSOM. Esta comparação é importante pois demonstra o desempenho superior das extensões do modelo KSOM em relação a este.

O resultado apresentado pelo modelo fuzzy TS é estatisticamente igual ao valor definido pelo P-MKSOM. Dos quatro melhores modelos listados na Tabela 6.11, três deles são modelos locais. Com base nos valores atingidos pelos quatro melhores modelos pode-se afirmar que os mesmos apresentam desempenhos semelhantes, pois os resultados são iguais até a terceira casa decimal.

Com relação à influência do algoritmo de quantização nos desempenhos dos modelos D-MKSOM, P-MKSOM e fuzzy TS, pode-se analisar os resultados mostrados na Tabela 6.12 para o conjunto de dados CSTR. É interessante notar que, para o modelo D-MKSOM, os desempenhos foram basicamente os mesmos, indicando que o modelo D-MKSOM, para este conjunto de dados, é muito menos dependente do algoritmo de quantização vetorial usado. Em casos como este, recomenda-se o uso do algoritmo de menor custo computacional que a rede SOM no projeto do modelo *D*-MKSOM.

No caso do modelo P-MKSOM, o uso do algoritmo SOM levou ao melhor desempenho, mas todos os valores vistos são diferentes entre si, demonstrando que o modelo P-MKSOM é sensível ao quantizador vetorial utilizado, ao contrário do que foi mostrado no D-MKSOM. Para o modelo fuzzy TS, o que se percebe é a repetição dos valores obtidos até a terceira casa decimal, permitindo concluir que os modelos apresentaram os mesmos desempenhos.

Tabela 6.12 Desempenho para os modelos baseados no D-MKSOM, P-MKSOM e fuzzy TS usando diferentes algoritmos de quantização vetorial (dados CSTR).

Modelo D-MKSOM				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
SOM	0.0331	0.0330	0.0331	1.92e-010
K-médias	0.0331	0.0330	0.0331	2.41e-010
FSCL	0.0331	0.0330	0.0332	3.91e-010
WTA	0.0331	0.0330	0.0333	1.04e-009
FCL	0.0331	0.0330	0.0338	7.76e-009
Modelo P-MKSOM				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
SOM	0.0338	0.0333	0.0365	3.69e-007
K-médias	0.0425	0.0355	0.0509	1.08e-005
FSCL	0.0485	0.0386	0.0704	2.92e-005
WTA	0.0568	0.0418	0.0790	6.33e-005
FCL	0.1219	0.0633	0.2662	0.0017
Modelo fuzzy TS				
Algoritmos QV	*NMSE*			
	média	*min*	*max*	*variância*
FCL	0.0331	0.0331	0.0331	8.97e-011
K-means	0.0332	0.0332	0.0332	1.56e-012
FSCL	0.0333	0.0333	0.0333	9.54e-013
WTA	0.0334	0.0334	0.0334	9.67e-013
SOM	0.0339	0.0338	0.0340	1.47e-009

6.3 Análise dos Resíduos

O próximo critério de avaliação dos modelos globais e locais estudados nesta tese consiste na análise dos resíduos (AGUIRRE, 2007) produzidos na etapa de teste das redes neurais. Na teoria de identificação de sistemas, esta análise é de grande importância para validar o modelo conforme sua capacidade de confirmar uma hipótese subjacente ao processo de modelagem que assume que o resíduo gerado corresponde a uma sequência de ruído branco Gaussiano de média zero e variância σ^2.

Por outro lado, se os resíduos não forem brancos, haverá informação contida neles indicando que o modelo não conseguiu extrair tudo que era explicável nos dados, ou seja, ainda existe dinâmica a ser determinada pelo modelo em questão. Portanto, a finalidade maior desta técnica é buscar validar a modelagem feita pelo modelo neural na etapa de treinamento, e conseguir identificar os dados de tal forma que a única informação que não pode ser predita está associada às imperfeições inerentes ao processo que aqui são

representadas pelo ruído de medição.

Para validação dos modelos obtidos, é importante utilizar o conjunto de dados de teste (AGUIRRE, 2007). Seguindo este procedimento evita-se a repetição dos dados ora empregados no treinamento, contornando possíveis falhas que surgem na validação do modelo final. Os dados a serem utilizados deverão conter em si a dinâmica da respectiva planta em análise. Nesta tese, serão mantidas as divisões proporcionais dos conjunto de dados que foram apresentados na Seção 6.2, ou seja, serão utilizados os mesmos particionamentos dos conjuntos de treinamento/teste usados nas simulações anteriores.

6.3.1 Validação dos Modelos Não-Lineares

A validação de modelos é um passo importante no processo de identificação dos dados provenientes de cada planta industrial listada anteriormente. A validação aplicada a um modelo indica a sua capacidade de representar o sistema original. A validação de modelos pode ser dividida em duas vertentes: (i) validação estatística e (ii) validação dinâmica.

Inúmeros métodos têm sido desenvolvidos para validação de modelos lineares. Alguns destes métodos baseiam-se nas funções de autocorrelação (ou de correlação cruzada) que verificam se os resíduos de identificação $e(t)$ são brancos e não correlacionados com a entrada (BOHLIN, 1971; BOX; JENKINS, 1976; LJUNG, 1999). Outro método é a validação baseada na comparação de modelos, que envolve a aplicação de testes estatísticos para comparar modelos e selecionar o melhor (AKAIKE, 1974; WADSWORTTH; BRYAN, 1974). Para sistemas não lineares estes testes não são suficientes por não verificarem não linearidades nos resíduos (BILLINGS; VOON, 1983, 1986).

Para verificar o quanto o modelo neural obtido independe dos conjuntos de dados de treinamento, é interessante constatar se há ou não correlação entre o sinal de entrada $u(t)$ de cada conjunto de dados com o resíduo gerado por este modelo. Para isto, a validação de modelos não-lineares deve verificar a validade das seguintes relações (BILLINGS; VOON, 1983, 1986; BILLINGS; ZHU, 1994):

$$\Phi_{ee}(\tau) = E\{e(t-\tau)e(t)\} = \delta(\tau), \tag{6.2}$$

$$\Phi_{ue}(\tau) = E\{u(t-\tau)e(t)\} = 0, \quad \forall\tau, \tag{6.3}$$

$$\Phi_{u^2e}(\tau) = E\{[u^2(t) - \overline{u^2(t)}]e(t-\tau)\} = 0, \quad \forall\tau, \tag{6.4}$$

$$\Phi_{u^2e^2}(\tau) = E\{[u^2(t) - \overline{u^2(t)}]e^2(t-\tau)\} = 0, \quad \forall\tau, \tag{6.5}$$

onde $E(\cdot)$ é o operador valor esperado, $\delta(\tau)$ é a função delta de Dirac e a barra indica o

valor médio do sinal, $e(t)$ é a sequência de resíduos obtida pelo modelo na etapa de teste, e $u(t)$ é o sinal de entrada usada na etapa de treinamento da rede neural.

Seja N_2 o número de amostras disponíveis para teste do modelo neural. Assim, define-se um intervalo de confiança de 95%, dentro do qual as estimativas das funções de autocorrelação e correlação cruzada são consideradas nulas. Os limites do intervalo de confiança de 95%, para funções normalizadas pelo desvio padrão, são: $\pm 1,96/N_2$ (BILLINGS; VOON, 1983, 1986).

A validação estatística sugere que não existem correlações não modeladas nos resíduos de teste. Contudo, os modelos assim validados podem não apresentar o mesmo comportamento dinâmico do sistema original (AGUIRRE; BILLINGS, 1994). Na validação dinâmica de um modelo, verifica-se se este apresenta características dinâmicas semelhantes àquelas do sistema sendo identificado. Este tipo de validação é muito usada na identificação de sistemas dinâmicos caóticos e, portanto, fora do escopo desta tese.

Nas subseções subsequentes, serão apresentados os resultados do uso das técnicas estatísticas de validação de modelos não-lineares supracitadas aplicadas aos três melhores modelos apresentados na Seção 6.2, para cada conjunto de dados.

6.3.1.1 Resultados - Atuador Hidráulico

Os resultados da análise dos resíduos para os três melhores modelos listados na Tabela 6.2, a saber, modelos D-MKSOM, P-MKSOM e ELM (nesta ordem), são mostrados na Figura 6.8. O uso desta análise, como comentado anteriormente, é de grande importância para verificar o quanto o modelo conseguiu extrair de informação relevante da dinâmica existente nos dados. Após a etapa de treinamento, a validação do modelo será feita em cima dos resíduos obtidos a partir do conjunto de teste.

Analisando os resultados obtidos, percebe-se que os gráficos referentes ao modelo D-MKSOM demonstraram que a modelagem realizada foi validada pelas condições definidas por cada função nos testes estatísticos apresentados nas Equações (6.2) a (6.5). Contudo, os gráficos para os modelos P-MKSOM e fuzzy TS apresentaram resultados que não satisfazem plenamente os mesmos testes. Logo, o modelo linear local múltiplo D-MKSOM é o mais indicado para identificação inversa desta planta por ter tido seus resultados validados estatisticamente.

6.3.1.2 Resultados - Braço Robótico

Os resultados da análise dos resíduos para os três melhores modelos listados na Tabela 6.5, a saber, modelos D-MKSOM, KSOM, fuzzy TS (nesta ordem), são mostrados na Figura 6.9. Analisando os gráficos, percebe-se que as três arquiteturas apresentaram resultados semelhantes. Em particular, destaca-se as FACs dos resíduos, e espera-se que as mesmas se assemelhem a uma função delta de Dirac (ver Figuras 6.9(a),(b),(c)). Se o modelo fosse linear, isto já seria suficiente para indicar que o modelo não é adequado. Porém, todas as demais funções de correlação apresentaram comportamento dentro do esperado. Por isso, analisando conjuntamente as quatro funções de correlação, o processo de identificação realizado pelos três modelos pode ser considerado adequado.

6.3.1.3 Resultados - Trocador de Calor

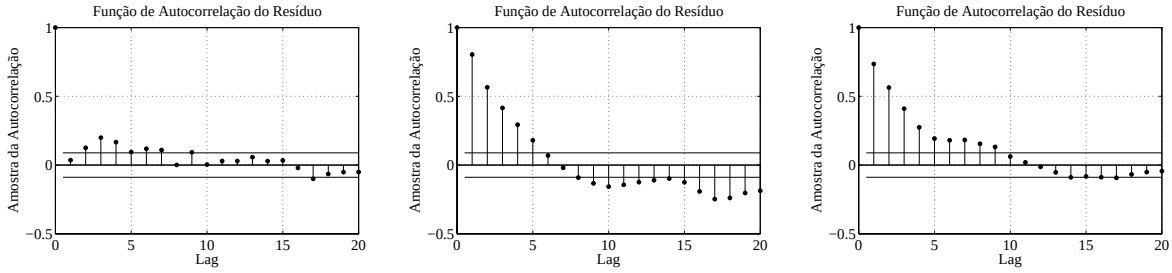
Os resultados da análise dos resíduos para os três melhores modelos listados na Tabela 6.8, a saber, modelos fuzzy TS, ELM e D-MKSOM (nesta ordem), são mostrados na Figura 6.10. Ao analisar esta figura, pode-se concluir que todos os modelos são estatisticamente válidos. No caso da função linear $\Phi_{ee}(\tau)$, parece haver uma pequena correlação de curto prazo (até o lag 2). Para o caso da função não-linear $\Phi_{u^2e^2}(\tau)$ em alguns lags as amplitudes saem dos limites do intervalo de confiança de 95%, mas este comportamento parece ser fruto do acaso. Para concluir, quando as quatro funções de correlação são analisadas conjuntamente, o processo de identificação para os três modelos pode ser considerado satisfatório.

6.3.1.4 Resultados - CSTR

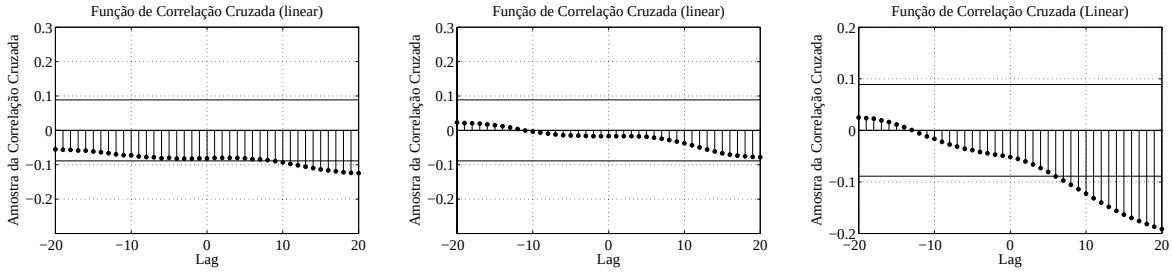
Finalmente, os resultados da análise dos resíduos para os três melhores modelos listados na Tabela 6.11, a saber, modelos ELM, D-MKSOM e P-MKSOM (nesta ordem), são mostrados na Figura 6.11. Avaliando os gráficos obtidos, é perceptível que todos os resultados foram semelhantes entre si demonstrando claramente a adequação dos modelos à tarefa de identificação da dinâmica inversa deste conjunto de dados.

6.4 Teste de Kolmogorov-Smirnov

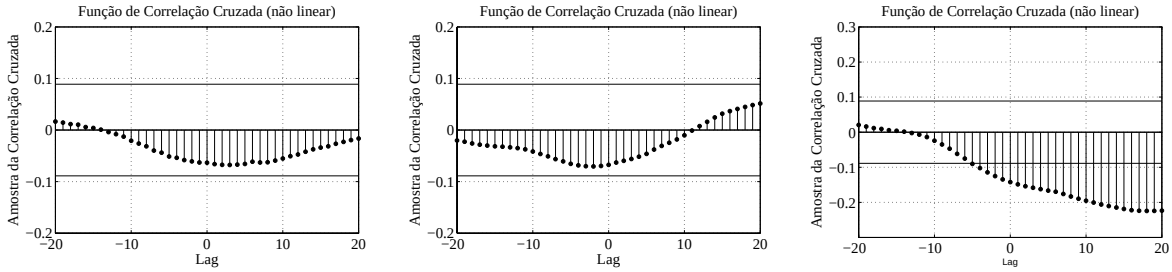
O conjunto final de experimentos tem como objetivo avaliar o grau de similaridade, de um ponto de vista estatístico, dentre a sequência de resíduos gerados pelos modelos



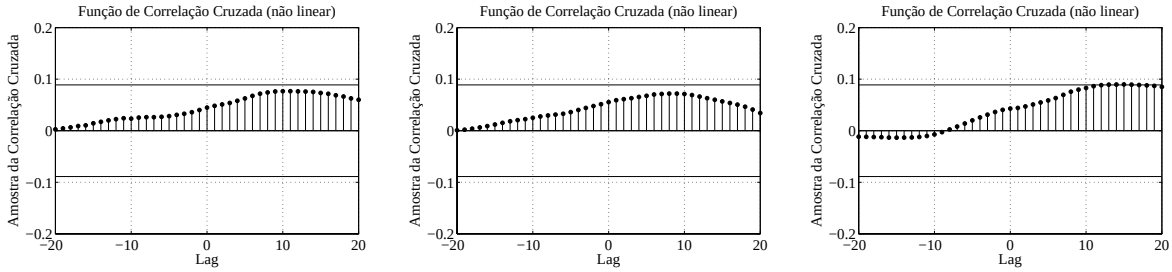
(a) $\Phi_{ee}(\tau)$ - D-MKSOM (b) $\Phi_{ee}(\tau)$ - P-MKSOM (c) $\Phi_{ee}(\tau)$ - fuzzy TS



(d) $\Phi_{ue}(\tau)$ - D-MKSOM (e) $\Phi_{ue}(\tau)$ - P-MKSOM (f) $\Phi_{ue}(\tau)$ - fuzzy TS

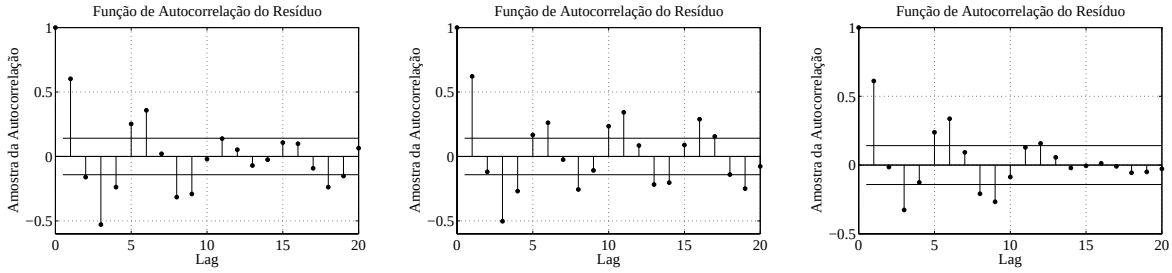


(g) $\Phi_{u^2e}(\tau)$ - D-MKSOM (h) $\Phi_{u^2e}(\tau)$ - P-MKSOM (i) $\Phi_{u^2e}(\tau)$ - fuzzy TS

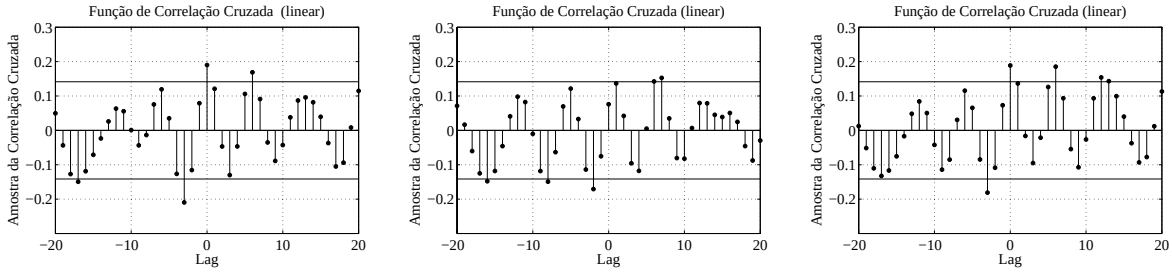


(j) $\Phi_{u^2e^2}(\tau)$ - D-MKSOM (k) $\Phi_{u^2e^2}(\tau)$ - P-MKSOM (l) $\Phi_{u^2e^2}(\tau)$ - fuzzy TS

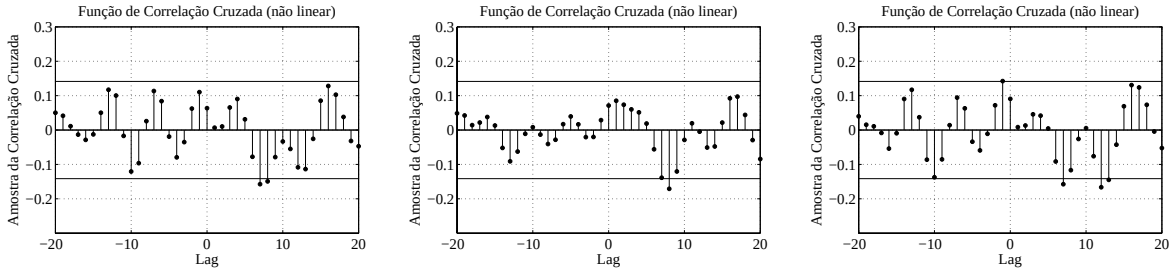
Figura 6.8 Análise dos resíduos para os modelos D-MKSOM (a,d,g,j), P-MKSOM (b,e,h,k) e fuzzy TS (c,f,i,l) (atuador hidráulico).



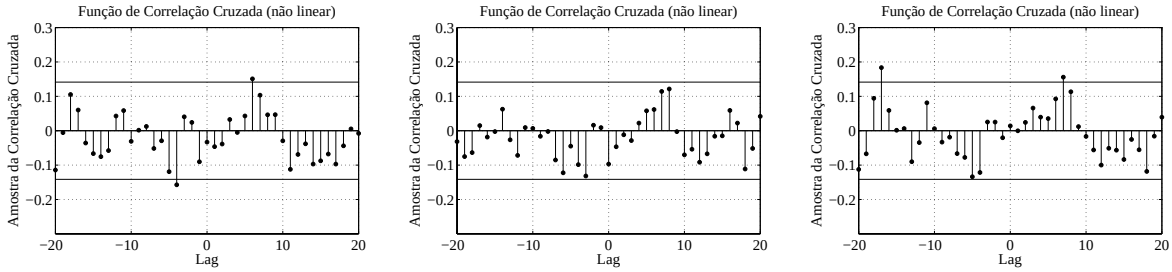
(a) $\Phi_{ee}(\tau)$ - D-MKSOM (b) $\Phi_{ee}(\tau)$ - KSOM (c) $\Phi_{ee}(\tau)$ - fuzzy TS



(d) $\Phi_{ue}(\tau)$ - D-MKSOM (e) $\Phi_{ue}(\tau)$ - KSOM (f) $\Phi_{ue}(\tau)$ - fuzzy TS

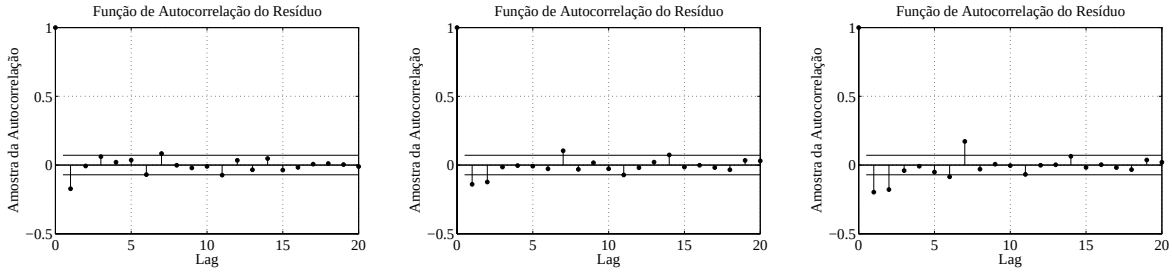


(g) $\Phi_{u^2e}(\tau)$ - D-MKSOM (h) $\Phi_{u^2e}(\tau)$ - KSOM (i) $\Phi_{u^2e}(\tau)$ - fuzzy TS



(j) $\Phi_{u^2e^2}(\tau)$ - D-MKSOM (k) $\Phi_{u^2e^2}(\tau)$ - KSOM (l) $\Phi_{u^2e^2}(\tau)$ - fuzzy TS

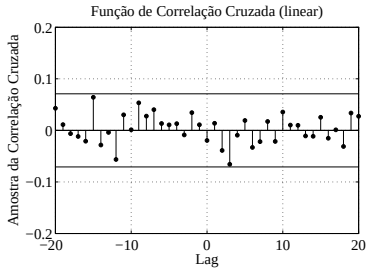
Figura 6.9 Análise dos resíduos para os modelos D-MKSOM (a,d,g,j), KSOM (b,e,h,k) e fuzzy TS (c,f,i,l) (braço robótico).



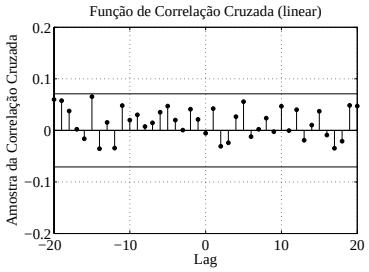
(a) $\Phi_{ee}(\tau)$ - fuzzy TS

(b) $\Phi_{ee}(\tau)$ - ELM

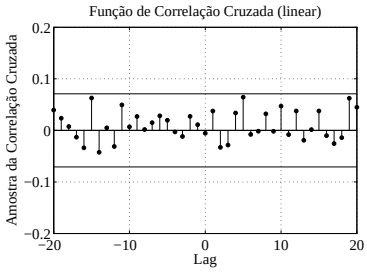
(c) $\Phi_{ee}(\tau)$ - D-MKSOM



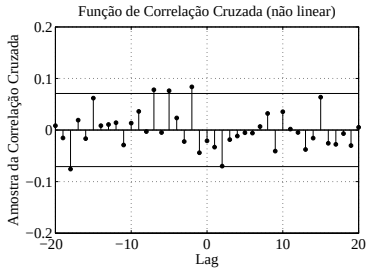
(d) $\Phi_{ue}(\tau)$ - fuzzy TS



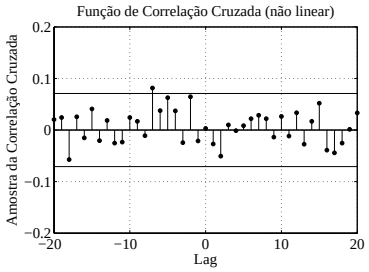
(e) $\Phi_{ue}(\tau)$ - ELM



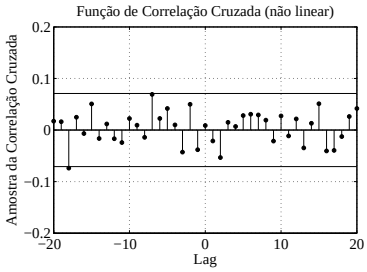
(f) $\Phi_{ue}(\tau)$ - D-MKSOM



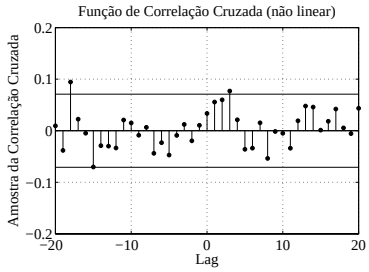
(g) $\Phi_{u^2e}(\tau)$ - fuzzy TS



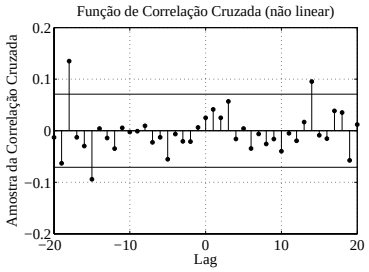
(h) $\Phi_{u^2e}(\tau)$ - ELM



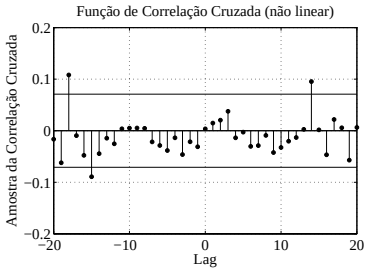
(i) $\Phi_{u^2e}(\tau)$ - D-MKSOM



(j) $\Phi_{u^2e^2}(\tau)$ - fuzzy TS

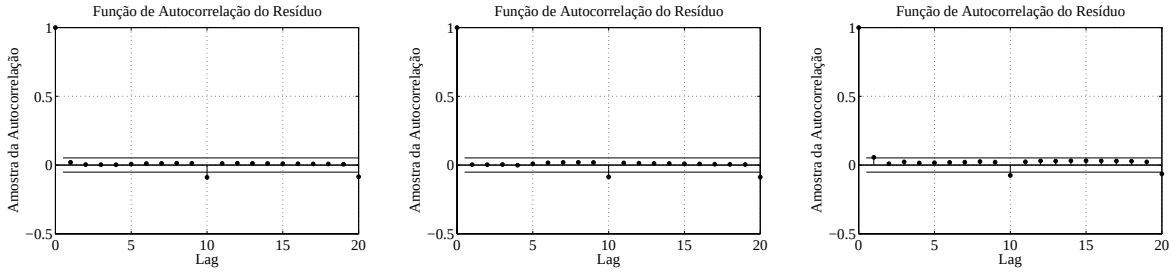


(k) $\Phi_{u^2e^2}(\tau)$ - ELM



(l) $\Phi_{u^2e^2}(\tau)$ - D-MKSOM

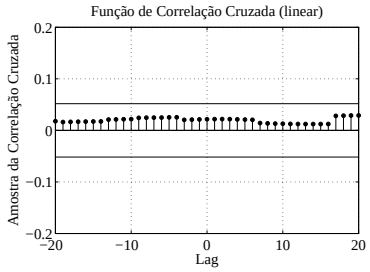
Figura 6.10 Análise dos resíduos para os modelos fuzzy TS (a,d,g,j), ELM (b,e,h,k) e D-MKSOM (c,f,i,l) (trocador de calor).



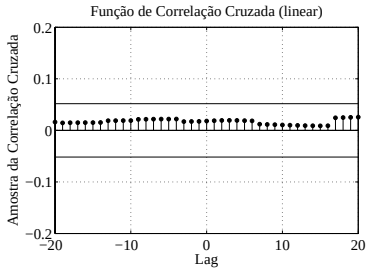
(a) $\Phi_{ee}(\tau)$ - ELM

(b) $\Phi_{ee}(\tau)$ - D-MKSOM

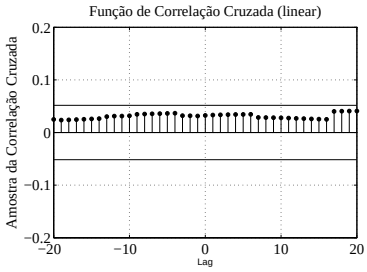
(c) $\Phi_{ee}(\tau)$ - P-MKSOM



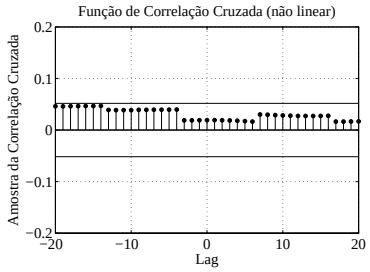
(d) $\Phi_{ue}(\tau)$ - ELM



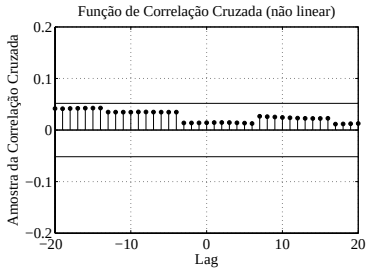
(e) $\Phi_{ue}(\tau)$ - D-MKSOM



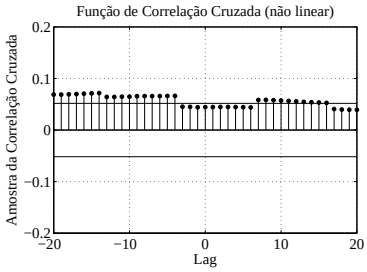
(f) $\Phi_{ue}(\tau)$ - P-MKSOM



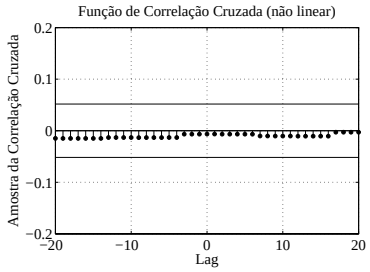
(g) $\Phi_{u^2e}(\tau)$ - ELM



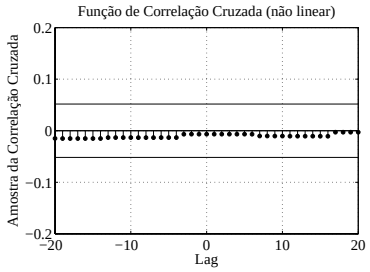
(h) $\Phi_{u^2e}(\tau)$ - D-MKSOM



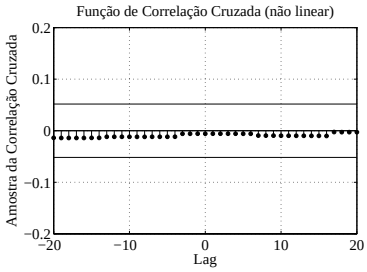
(i) $\Phi_{u^2e}(\tau)$ - P-MKSOM



(j) $\Phi_{u^2e^2}(\tau)$ - ELM



(k) $\Phi_{u^2e^2}(\tau)$ - D-MKSOM



(l) $\Phi_{u^2e^2}(\tau)$ - P-MKSOM

Figura 6.11 Análise dos resíduos para os modelos ELM (a,d,g,j), D-MKSOM (b,e,h,k) e P-MKSOM (c,f,i,l) (CSTR).

P-MKSOM, D-MKSOM e fuzzy TS para diferentes algoritmos de quantização vetorial. Para este propósito, foi usado o teste Kolmogorov-Smirnov (teste KS) (SOONG, 2004). O teste KS quantifica uma distância entre as CDFs[2] de duas sequências de resíduos. A hipótese nula para ser testada é que as sequências geradas possuem a mesma distribuição, indicando que não há diferenças entre elas.

A razão para a aplicação do teste KS no âmbito desta tese é avaliar, por exemplo, se dois modelos P-MKSOM (D-MKSOM ou fuzzy TS) implementados usando diferentes algoritmos de quantização vetorial (QV) geram sequências de resíduos estatisticamente equivalentes ou não. Se for equivalentes, então os desempenhos dos modelos locais são independentes dos algoritmos de quantização vetorial utilizados.

Os resultados possíveis a serem obtidos a partir da aplicação deste teste de hipótese são os seguintes:

- A aceitação da hipótese nula indica que a CDF dos resíduos gerados pelos modelos P-MKSOM, D-MKSOM e fuzzy TS treinados com o algoritmo SOM é equivalente à CDF dos resíduos gerados pelos modelos P-MKSOM, D-MKSOM e fuzzy TS treinados com um outro algoritmo de quantização vetorial.
- Uma rejeição da hipótese nula indica que a CDF dos resíduos gerados pelos modelos P-MKSOM, D-MKSOM e fuzzy TS treinados com o algoritmo SOM é diferente da CDF dos resíduos gerados pelos modelos P-MKSOM, D-MKSOM e fuzzy TS treinados com um outro algoritmo de quantização vetorial.

Da Tabela 6.13 pode-se inferir que, para os conjunto de dados do atuador hidráulico e CSTR, o desempenho do modelo P-MKSOM treinado via algoritmo SOM não é estatisticamente equivalente àqueles obtidos quando o modelo P-MKSOM é treinado via demais algoritmos de quantização vetorial. Esta observação se repete em parte para os dados do braço robótico, exceto para o algoritmo K-médias que apresentou equivalência estatística com o algoritmo SOM. Para os dados do trocador de calor, o desempenho do modelo P-MKSOM treinado como algoritmo SOM equivale ao de quando é treinado com os algoritmos WTA e K-médias. Para todos os casos de equivalência, recomenda-se ao usuário escolher o algoritmo de quantização vetorial de menor custo computacional para construir o modelo local de interesse (nestes casos, os algoritmos K-médias e WTA).

É importante ressaltar que os resultados da aplicação do teste KS ao modelo P-MKSOM estão em plena concordância com os resultados mostrados nas Tabelas 6.6, 6.9

[2]Acrônimo de *cumulative distribution function* ou função de distribuição acumulada, em português.

Tabela 6.13 Resultados do teste KS sobre o desempenho do modelo P-MKSOM.

Atuador Hidráulico	
Algoritmos QV	**Resultados Teste KS**
FCL	Rejeita
FSCL	Rejeita
WTA	Rejeita
K-médias	Rejeita

Braço Robótico	
Algoritmos QV	**Resultados Teste KS**
FCL	Rejeita
WTA	Rejeita
FSCL	Rejeita
K-médias	Aceita

Trocador de Calor	
Algoritmos QV	**Resultados Teste KS**
FCL	Rejeita
FSCL	Rejeita
WTA	Aceita
K-médias	Aceita

CSTR	
Algoritmos QV	**Resultados Teste KS**
FCL	Rejeita
FSCL	Rejeita
WTA	Rejeita
K-médias	Rejeita

e 6.12 em que o critério de decisão é baseado nos valores do NMSE. Há uma discrepância, porém, com os resultados para o atuador hidráulico. Na Tabela 6.3 a avaliação pelo critério do NMSE sugere que os modelos P-MKSOM treinados pelos algoritmos SOM e K-médias são equivalentes, enquanto na Tabela 6.13 o critério do teste KS sugere que não são equivalentes. Nestes casos, recomenda-se seguir a decisão pelo critério do teste KS, por este ser uma ferramenta mais precisa, do ponto de vista matemático.

Analisando agora os resultados mostrados na Tabela 6.14, constata-se que o desempenho do modelo D-MKSOM treinado via algoritmo SOM é estatisticamente equivalente àqueles obtidos pelo treinamento por todos os outros algoritmos de quantização vetorial. Esta conclusão se aplica a todos os conjuntos de dados utilizados, sem exceção. Como já mencionado, em caso de equivalência, recomenda-se ao usuário escolher o algoritmo de quantização vetorial com menor custo computacional.

Com base nos resultados mostrados nas Tabelas 6.13 e 6.14, conclui-se que o modelo D-MKSOM é mais robusto (ou menos sensível) do que o modelo P-MKSOM quanto a mudanças no algoritmo de quantização vetorial utilizado no treinamento do modelo. Esta robustez pode estar associada a questão de que o modelo D-MKSOM utiliza os dados mapeados nas células de Voronoi dos vetores protótipos para obtenção dos vetores de coeficientes associado a cada vetor. Já no caso do P-MKSOM, o que se tem é o emprego dos próprios vetores protótipos para obtenção do seu respectivo vetor de coeficientes.

Agora analisando o modelo fuzzy TS, são utilizados os mesmos algoritmos de quantização vetorial para implementação dos modelos em questão. Todos os algoritmos foram

Tabela 6.14 Resultados do Teste KS sobre o desempenho do modelo D-MKSOM.

Atuador Hidráulico	
Algoritmos QV	**Resultados Teste KS**
FCL	Aceita
FSCL	Aceita
WTA	Aceita
K-médias	Aceita

Braço Robótico	
Algoritmos QV	**Resultados Teste KS**
FCL	Aceita
FSCL	Aceita
WTA	Aceita
K-médias	Aceita

Heat exchanger	
Algoritmos QV	**Resultados Teste KS**
FCL	Aceita
FSCL	Aceita
WTA	Aceita
K-médias	Aceita

CSTR	
Algoritmos QV	**Resultados Teste KS**
FCL	Aceita
FSCL	Aceita
WTA	Aceita
K-médias	Aceita

testados para as quatro plantas industriais já comentadas, e os testes foram feitos comparando os resíduos gerados pela rede fuzzy TS com algoritmo SOM e com os demais modelos concebidos pelos quantizadores vetoriais. Os resultados obtidos estão representados na Tabela 6.15.

Como visto na tabela, a família dos modelos fuzzy TS apresentaram resultados satisfatórios em relação aos conjuntos de dados do braço robótico e do trocador de calor. Todos os modelos utilizados apresentaram aceitação da hipótese nula comparados com o modelo baseado no algoritmo SOM. Esta conclusão é importante pois entre as arquiteturas existentes pode-se escolher aquela que é mais leve computacionalmente.

Já para os conjunto de dados do atuador hidráulico e do CSTR o mesmo não aconteceu, pois houve uma rejeição maciça da hipótese nula entre os modelos utilizados e o algoritmo SOM, excluindo somente o caso da rede K-means no atuador hidráulico, onde os resíduos gerados pelos dois possuem uma mesma distribuição de probabilidade desconhecida.

Comparando-se os resultados obtidos pelo modelo fuzzy TS e os modelos propostos, vê-se os valores contidos nas Tabelas 6.13 e 6.14 que no caso do P-MKSOM os modelos não apresentaram tanta similaridade, ao contrário do D-MKSOM em que para todos os conjuntos de dados os mesmos se sairão similares em tudo, demonstrando que esta arquitetura é menos sensível ao modelo de algoritmo de quantização vetorial escolhido. Este aspecto já não é tão presente na arquitetura fuzzy TS pois o mesmo apresentou dois casos de similaridade com os dados do braço robótico e trocador de calor, e outros dois

Tabela 6.15 Resultados do Teste KS sobre o desempenho do modelo fuzzy TS.

Atuador Hidráulico	
Algoritmos QV	**Resultados Teste KS**
FCL	Rejeita
FSCL	Rejeita
WTA	Rejeita
K-means	Aceita

Braço Robótico	
Algoritmos QV	**Resultados Teste KS**
FCL	Aceita
WTA	Aceita
FSCL	Aceita
K-means	Aceita

Trocador de Calor	
Algoritmos QV	**Resultados Teste KS**
FCL	Aceita
FSCL	Aceita
WTA	Aceita
K-means	Aceita

CSTR	
Algoritmos QV	**Resultados Teste KS**
FCL	Rejeita
FSCL	Rejeita
WTA	Rejeita
K-means	Rejeita

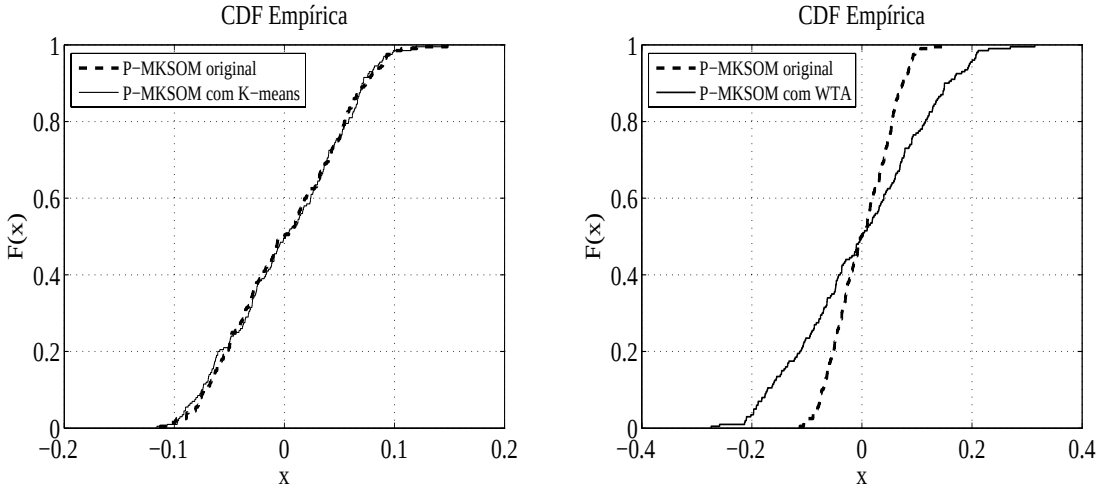
não similares com os dados do atuador hidráulico e CSTR.

Para finalizar os testes com os modelos, a simulação a seguir tem o intuito de ilustrar o funcionamento do teste KS, ao comparar a distância entre as CDFs empíricas dos resíduos gerados por um mesmo modelo (e.g. P-MKSOM), que foi treinado por dois algoritmos de quantização vetorial distintos, para um certo conjunto de dados. São discutidos dois casos, um em que há equivalência entre os desempenhos (implicando na aceitação da hipótese nula) e outro em que não há equivalência entre os desempenhos (implicando na rejeição da hipótese nula).

A Figura 6.12(a) ilustra a sobreposição das CDFs empíricas do modelo P-MKSOM treinado com os algoritmos SOM e K-médias, enquanto a Figura 6.12(b) ilustra o caso em que o modelo é treinado com os algoritmos SOM e WTA para o conjunto de dados do braço robótico. No primeiro caso, a diferença entre as CDFs é praticamente imperceptível (o que faz com que o teste KS aceite a hipótese nula), enquanto no segundo caso a diferença entre as CDFs é nítida (o que leva o teste KS rejeitar a hipótese nula). Análise similar pode ser levada a cabo para os resultados mostrados nas Figuras 6.13(a) e (b) para o conjunto de dados do trocador de calor.

6.5 Resumo

Neste capítulo foram apresentados os resultados da avaliação abrangente dos desempenhos dos modelos propostos nesta tese segundo três diferentes critérios, a saber: (i)



(a) Aceitação da hipótese nula

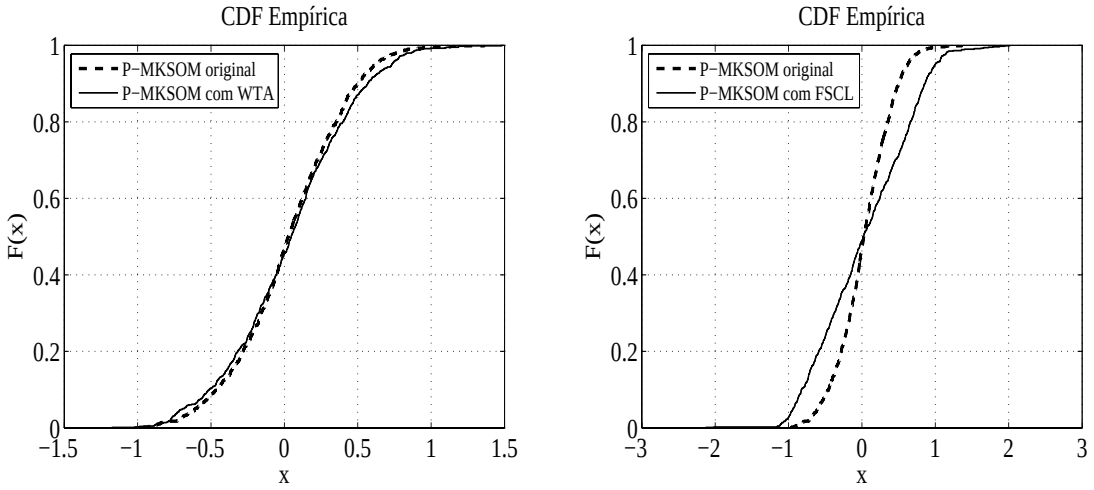
(b) Rejeição da hipótese nula

Figura 6.12 Comparação entre as CDF empíricas do modelo P-MKSOM treinado com diferentes algoritmos de quantização vetorial (braço robótico).

erro médio quadrático normalizado (*NMSE*), (*ii*) análise dos resíduos e (*iii*) teste de Kolmogorov-Smirnov. Para este fim, foram utilizados quatro conjuntos de dados gerados por diferentes sistemas dinâmicos comumente usados em *benchmarks* de desempenho. Além disso, os modelos locais propostos (KSOM, P-MKSOM e D-MKSOM) tiveram seus desempenhos comparados aos de outros modelos locais (modelos LLM e fuzzy TS) e aos de modelos globais baseados nas redes MLP e ELM.

De particular interesse para esta tese foi a avaliação da robustez dos algoritmos propostos (P-MKSOM e D-MKSOM) com relação ao algoritmo de quantização vetorial usado no treinamento do modelo. Esta validação foi formalizada através do uso do teste estatístico de Kolmogorov-Smirnov. E por último, foram apresentados os resultados referentes à avaliação do modelo fuzzy Takagi-Sugeno utilizando as mesmas ferramentas estatísticas, possibilitando a comparação deste com os modelos propostos. Resultados adicionais, referentes à avaliação da distribuição estatística dos resíduos geradas pelo modelos discutidos neste capítulo são apresentadas no Apêndice A.

No próximo capítulo são apresentadas as principais conclusões desta tese e também são sugeridos temas para pesquisas futuras na área de identificação de sistemas dinâmicos usando modelos locais baseados em algoritmos de quantização vetorial.



(a) Aceitação da hipótese nula

(b) Rejeição da hipótese nula

Figura 6.13 Comparação entre as CDF empíricas do modelo P-MKSOM treinado com diferentes algoritmos de quantização vetorial (trocador de calor).

7 Conclusão

7.1 Introdução

Esta tese reportou os resultados de um estudo comparativo de antigas e novas propostas de modelos lineares locais baseados na rede SOM e modelos globais baseados nas redes ELM e MLP. O problema escolhido foi a identificação de sistemas dinâmicos, mais especificamente a modelagem inversa de sistemas dinâmicos.

A área de identificação de sistemas usando modelos neurais supervisionados, principalmente as redes MLP e RBF, e mais recentemente a rede ELM (HUANG et al., 2006), é bem mais consagrada do que a que usa modelos locais, principalmente porque é sabido que as arquiteturas neurais supervisionadas são reconhecidas como aproximadores universais de funções (HORNIK et al., 1989).

A abordagem local vem a ser usada principalmente para tratar problemas em que há fortes suspeitas da presença de não-linearidades e descontinuidades em todo o domínio dos dados. Analisando o problema de uma forma local, consegue-se dividir todo espaço de dados em partições para as quais se constroem modelos mais simples usando os dados pertencentes àquelas partições. Estes modelos se tornam especializados (ou localizados) na dinâmica local do sistema referente a uma dada partição. A idéia básica consiste em aproximar uma relação funcional global, supostamente existente entre múltiplas variáveis de entrada e uma de saída, por meio de um conjunto de hiperplanos (isto é, modelos lineares), cada qual com seu vetor de coeficientes estimado com dados pertencentes apenas à partição local. Didaticamente, é como aproximar uma superfície não-linear qualquer por um conjunto de hiperplanos.

Como visto nos capítulos desta tese, houve uma sequência lógica de apresentação dos algoritmos a serem utilizados no problema de identificação inversa de sistemas dinâmicos. Começou-se então pela abordagem global (Capítulo 2), mais comum, descrevendo as redes MLP e ELM. Em seguida, foram apresentados algoritmos de quantização vetorial

usados para particionamento do domínio do problema (Capítulo 3), para então descrever modelos lineares locais já disponíveis na literatura especializada (Capítulo 4), a saber os modelos LLM, VQTAM, KSOM e LESSOM. No Capítulo 5 foram introduzidos os modelos P-MKSOM e D-MKSOM, principais propostas deste trabalho. Os resultados do amplo estudo comparativo entre modelos globais e locais levado a cabo nesta tese foram apresentados no Capítulo 6.

A seguir, apresenta-se um resumo das principais contribuições desta tese e discutem-se as conclusões que se podem tirar delas.

7.2 Resumo das Contribuições da Tese

O objetivo último deste trabalho foi o de mostrar, fruto já de alguma experiência prévia, que modelos lineares locais apresentam desempenhos equivalentes e até superiores que as abordagens neurais convencionais ao problema de identificação de sistemas dinâmicos. Acredita-se que este objetivo foi atingido em função dos resultados apresentados no capítulo anterior.

De modo mais específico as principais contribuições da tese foram as seguintes:

- Oferecer à comunidade científica, por meio de uma ampla revisão bibliográfica sobre o tema, um panorama geral das abordagens disponíveis na literatura sobre o projeto de modelos lineares locais baseados em redes neurais competitivas. Destacam-se aqui os modelos LLM, VQTAM, KSOM e Takagi-Sugeno[1].

- Proposição de mais duas estratégias de projeto de modelos lineares locais baseados na rede SOM, a saber, os modelos P-MKSOM e D-MKSOM. Para os quatro conjuntos de dados utilizados neste estudo, os modelos propostos sempre estiveram entre os de melhor desempenho, sem exceção, competindo de igual para igual com os modelos globais baseados em redes neurais supervisionadas.

- Aplicação da rede ELM em identificação de sistemas dinâmicos, comparando seu desempenho com modelos baseados na rede MLP. Os resultados obtidos são favoráveis à rede ELM, que também tem a seu favor o fato de ser bem mais simples de treinar que a rede MLP.

[1]Este é um modelo fuzzy, mas que usa algoritmos de quantização vetorial para seleção dos pontos focais.

- Proposição de uma metodologia de comparação de desempenho e validação de resultados baseada na aplicação conjunta de três critérios de avaliação: (*i*) erro médio quadrático normalizado, (*ii*) análise dos resíduos usando funções de autocorrelação e de correlação cruzada lineares e não-lineares e (*iii*) teste de Kolmogorov-Smirnov.

- Análise da sensibilidade (robustez) dos modelos propostos em relação ao algoritmo de quantização vetorial usado na partição do domínio do problema, qualitativamente por meio das estatísticas do erro quadrático médio e do teste de Kolmogorov-Smirnov.

7.3 Trabalhos Futuros

Um grande número de propostas de trabalhos na linha de pesquisa desta tese pode ainda ser desenvolvida. Dentre os potenciais temas de pesquisa futura podem ser listadas as seguintes direções:

- Aplicações dos modelos locais em diversas estratégias de sistemas de controle (e.g. controle preditivo).

- Extensão dos modelos propostos usando redes neurais competitivas crescentes, tais como a rede *Growing Neural Gas* (FRITZKE, 1994).

- Sintonia de parâmetros dos modelos propostos, tais como número de neurônios (g), número de vizinhos mais próximos (K) e ordens de regressão (p e q), por computação evolucionária.

- Extensão dos modelos propostos para identificação de sistemas MIMO.

- Extensão dos modelos locais estudados nesta tese (chamados de homogêneos por usarem o mesmo modelo local para todas as partições) para modelos locais heterogêneos (OH et al., 2005), em que os modelos locais podem ser diferentes.

APÊNDICE A – Diagrama de Caixas dos Resíduos

A.0.1 O Diagrama de Caixas (*Boxplot*)

Um diagrama de caixas apresenta os três quartis, o mínimo e o máximo dos dados em uma caixa retangular, alinhada horizontal ou verticalmente. A caixa inclui o intervalo interquartil, com a linha à esquerda (ou inferior) no primeiro quartil Q1, e a linha à direita (ou superior) no terceiro quartil Q3. Traça-se uma linha através da caixa no segundo quartil (que é o 50^o percentil ou a mediana) Q2=$\tilde{x}$. Uma linha, em ambas as pontas, se estende até os valores extremos. Essas linhas, algumas vezes chamadas "bigodes", podem se estender apenas até o 10^o e o 90^o percentis, ou o 5^o e o 95^o percentis em conjuntos de dados grandes (HINES et al., 2006). A Figura A.1 mostra o exemplo do uso do diagrama de caixas para os dados referentes à temperatura de saída do ar de um secador industrial [1]. Pelo diagrama percebe-se que a distribuição da temperatura de saída do ar não é bastante simétrica em torno do valor central.

Nas Figuras A.2 a A.5, apresentam-se os resultados da aplicação da ferramenta visual diagrama de caixas nos quatro conjuntos de dados, para verificar a distribuição dos valores dos resíduos provenientes dos melhores modelos locais, tais como D-MKSOM, P-MKSOM e fuzzy TS. Nestas figuras, além do diagrama de caixas, tem-se também a representação por histograma e o gráfico da distribuição normal em função dos dados. Todas estas representações são úteis na análise final da natureza estatística dos resíduos dos modelos.

No histograma tem-se a representação da distribuição dos dados através do uso de barras. A curva desenhada sobreposta representa o gráfico de uma distribuição normal que se deve confrontar com o comportamento destas barras. O último gráfico associa novamente os dados com a distribuição normal, só que a disposição dos mesmos em relação à distribuição é definida através de uma linha tracejada, ou seja, os dados assumirão este comportamento estatístico caso os mesmos se encontrem por sobre esta linha.

[1] http://homes.esat.kuleuven.be/ smc/daisy/daisydata.html

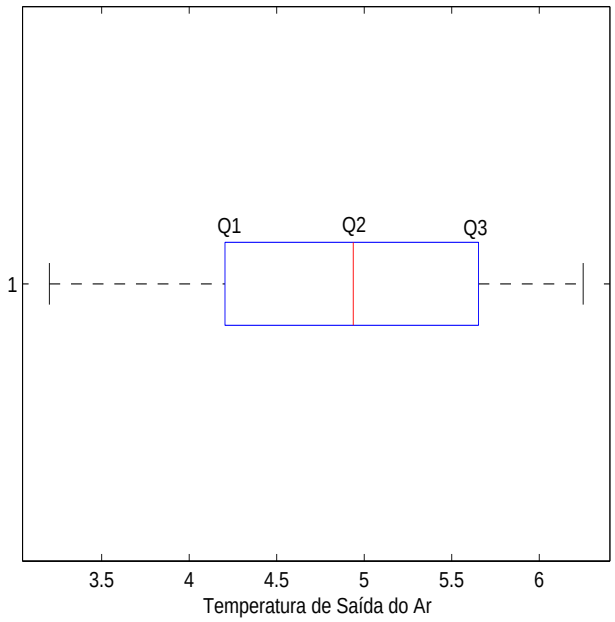


Figura A.1 Diagrama de caixas para temperatura de saída de um secador industrial.

Na Figura A.2(a)-(c), os diagramas de caixa apresentam *Outliers*, aqui representados pelo símbolo "+". Os *Outliers* são valores numericamente distantes em relação aos demais dados existentes, podendo assumir valores muito pequenos ou grandes. São valores localizados além dos "bigodes", ou *percentis*. Neste caso, o modelo D-MKSOM apresentou maior quantidade de valores extremos, enquanto o P-MKSOM apresentou menor quantidade. A distribuição dos dados em torno da mediana ficou bem equilibrada, definida pelo comprimento da linha tracejada que liga os "bigodes" até o primeiro e o terceiro *quartis*.

Os gráficos dos histogramas apresentados nas Figuras A.2(d)-(f) demonstram que os resíduos gerados pelos modelos D-MKSOM, P-MKSOM e fuzzy TS se aproximam ao comportamento estatístico apresentado por uma distribuição normal, aqui representado pela sobreposição da curva gaussiana sobre os mesmos. Nas Figuras A.2(g)-(i), tem-se uma outra forma de se visualizar os dados com base na disposição dos mesmos através de uma linha tracejada que representa o comportamento de uma distribuição normal. Portanto, com estas informações pode-se dizer que os resíduos assumem em sua grande maioria o comportamento estatístico de uma distribuição normal. Nestas figuras, pode-se visualizar também a presença dos valores extremos (*Outliers*) assumidos pelos resíduos através da representação dos pontos bem além da linha tracejada.

Nas Figuras A.3(a)-(c), ilustram-se claramente a ausência de *Outliers* nos gráficos de caixa dos três modelos D-MKSOM, P-MKSOM e fuzzy TS. A diferença perceptível entre

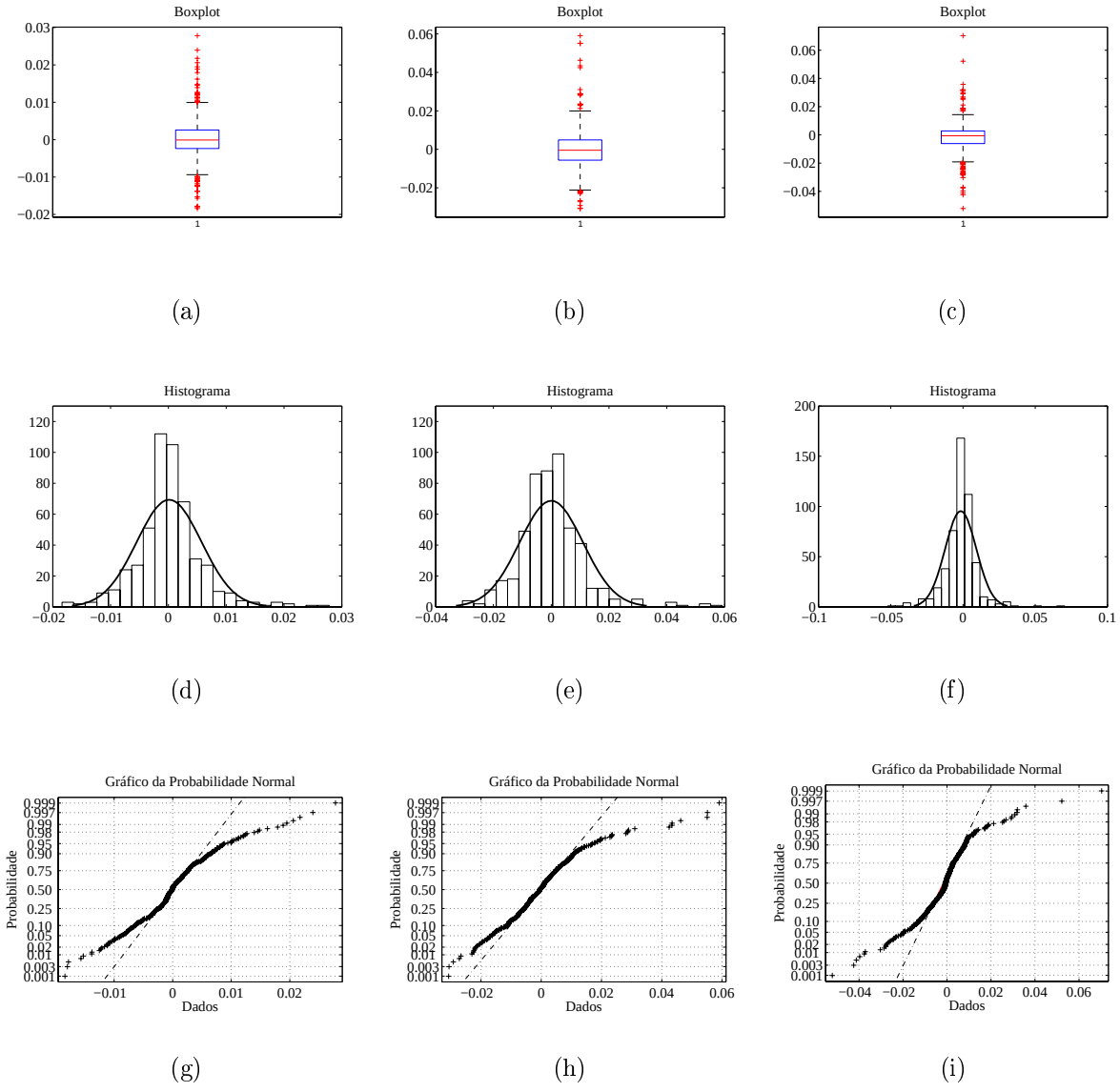


Figura A.2 Validação estatística dos modelos D-MKSOM (a,d,g), P-MKSOM (b,e,h) e fuzzy TS (c,f,i) para os dados do conjunto de teste do atuador hidráulico.

os resultados obtidos está principalmente na distribuição dos dados em torno da mediana. O comportamento visto entre os modelos demonstra claramente que o D-MKSOM possui melhor distribuição ao contrário dos que são apresentados pelo P-MKSOM e fuzzy TS, em que os comprimentos vistos das linhas tracejadas que ligam os "bigodes" até o primeiro e o terceiro *quartis* são diferentes.

Analisando os histogramas plotados nas Figuras A.3(d)-(f), verificam-se que as distribuições apresentadas pelos resíduos de todos os modelos se aproximam bem ao formato da Gaussiana e apresentam simetria em torno da média nula, permitindo concluir que os mesmos possuem aproximadamente uma distribuição normal. Esta informação pode ser confirmada com a verificação dos gráficos vistos nas Figuras A.3(g)-(i), em que o modelo D-MKSOM apresenta os resíduos que mais bem se ajustam à linha tracejada.

A validação estatística dos resíduos obtidos pelos modelos para o conjunto de dados do trocador de calor é apresentado na Figura A.4. Os diagramas de caixas apresentam *Outliers* sendo em uma quantidade menor comparado ao visto para o conjunto de dados do atuador hidráulico. Os comprimentos das linhas tracejadas são iguais e isso corresponde a uma distribuição de dados em torno da mediana de uma forma equilibrada.

Novamente, analisando os histogramas dos resíduos gerados pelos modelos nas Figuras A.4(d)-(f), verifica-se facilmente que os gráficos obtidos possuem o formato semelhante ao da curva Gaussiana, demonstrando um comportamento estatístico típico de uma distribuição normal. Fato este novamente constatado com a análise feita sobre os gráficos apresentados nas Figuras A.4(g)-(i), onde os resíduos se ajustam bem à linha tracejada.

Por último, são analisados os resíduos gerados pelos três modelos locais para os dados do CSTR. Nas Figuras A.5(a)-(c), mostra-se o diagrama de caixas dos resíduos dos modelos. Percebe-se que há ocorrência de *outliers* em todos os casos e que o desvio-padrão é nulo. Outro detalhe importante é que a distância entre os *quartis* é praticamente nula, sendo verificado isso através do comprimento das caixas. Com base nestas informações percebe-se que todos os modelos obtiveram resíduos aproximadamente iguais.

Na análise feita sobre os histogramas dos resíduos vistos nas Figuras A.5(d)-(f), verifica-se que há uma frequência maior dos valores em torno da média nula, aqui definido pela presença de picos. Isso pode ser constatado também nas Figuras A.5(g)-(i), em que uma grande parte dos dados apresenta valores próximos a zero, na ordem de 95%.

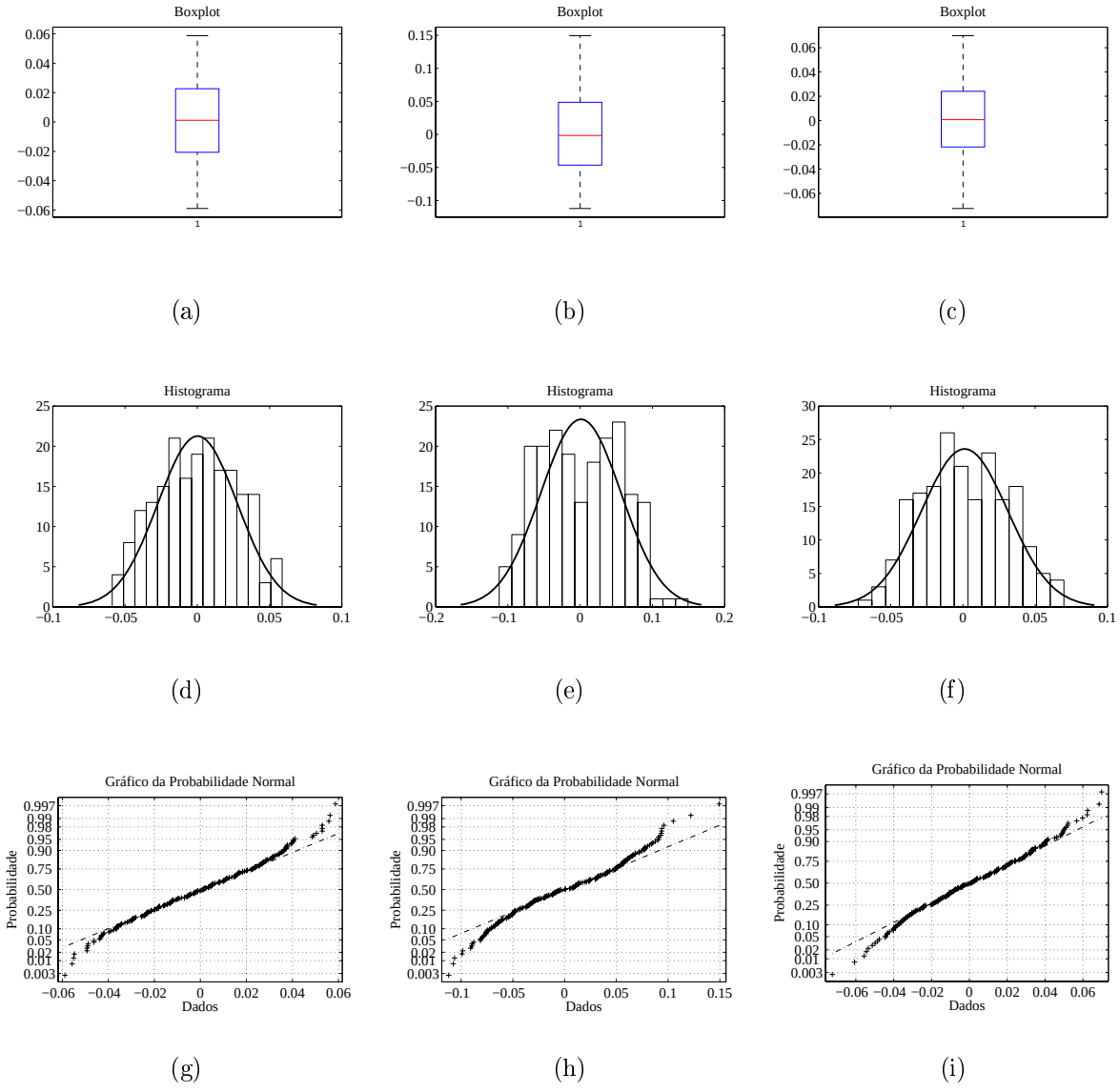


Figura A.3 Validação estatística dos modelos D-MKSOM (a,d,g), P-MKSOM (b,e,h) e fuzzy TS (c,f,i) para os dados do conjunto de teste do braço robótico.

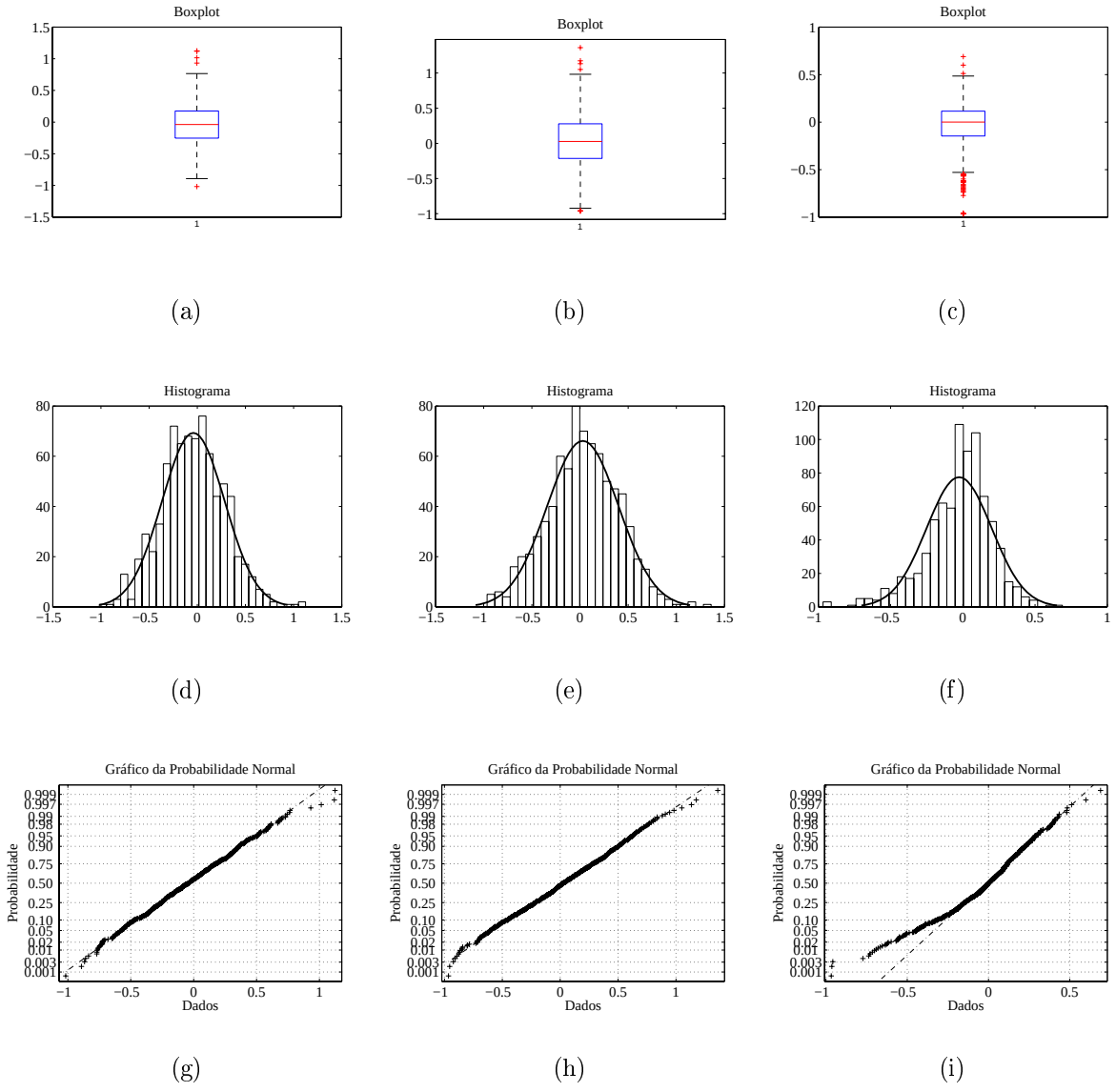


Figura A.4 Validação estatística dos modelos D-MKSOM (a,d,g), P-MKSOM (b,e,h) e fuzzy TS (c,f,i) para os dados do conjunto de teste do trocador de calor.

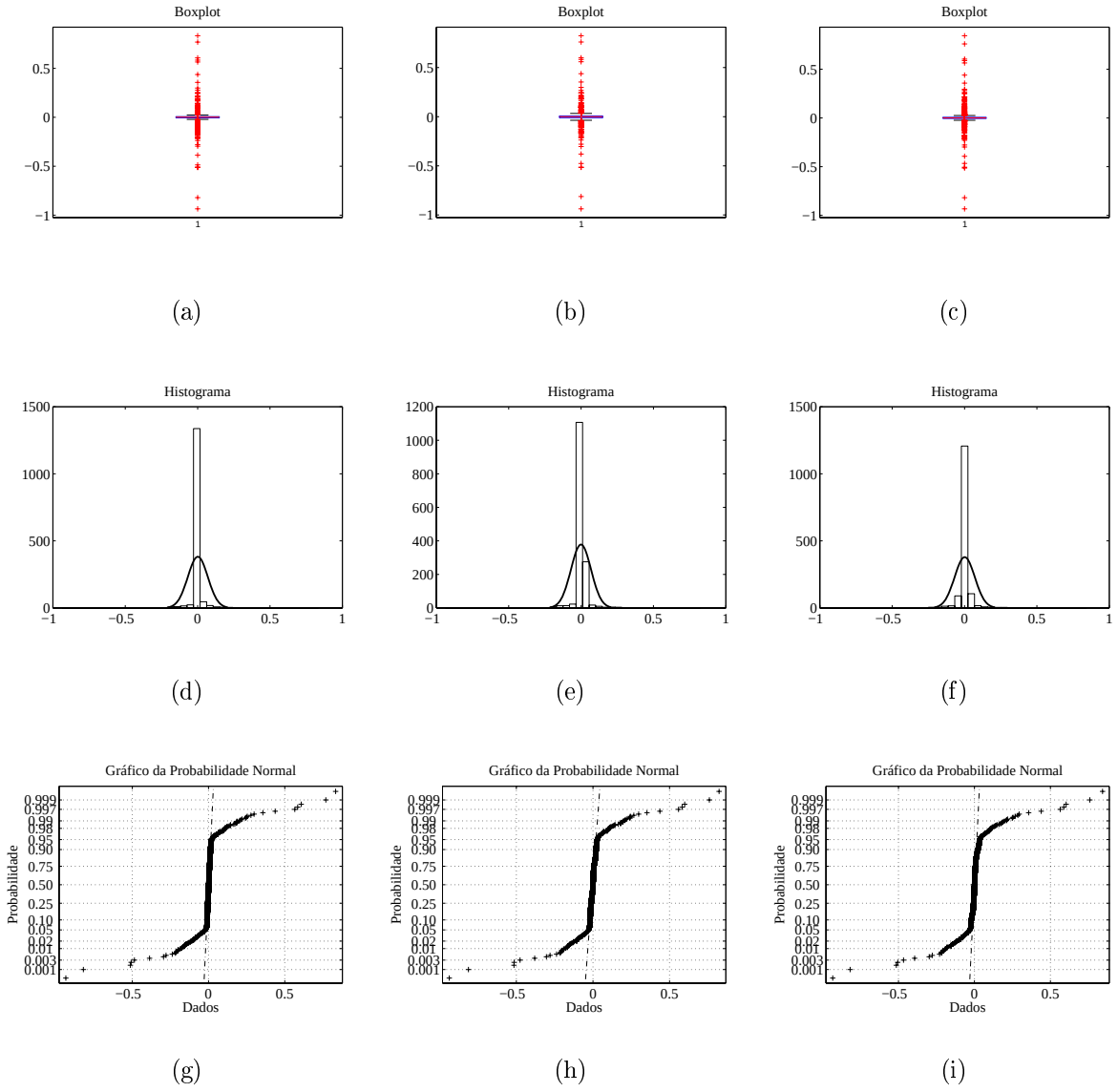


Figura A.5 Validação estatística dos modelos D-MKSOM (a,d,g), P-MKSOM (b,e,h) e fuzzy TS (c,f,i) para os dados do conjunto de teste CSTR.

Referências

AGUIRRE, L. A. *Introdução à Identificação de Sistemas: técnicas lineares e não-lineares aplicadas a sistemas reais*. Minas Gerais: Editora da UFMG, 2007. 3ª edição.

AGUIRRE, L. A.; BILLINGS, S. A. Validating identification nonlinear models with chaotic dynamics. *International Journal of Bifurcation and Chaos*, v. 4, n. 1, p. 109–125, 1994.

AHALT, S. et al. Competitive learning algorithms for vector quantization. *Neural Networks*, v. 3, n. 3, p. 277–290, 1990.

AKAIKE, H. A new look at statistical model identification. *IEE Transations on Automatic Control*, v. 19, n. 6, p. 716–723, 1974.

AZEEM, M. F.; HANMANDLU, M.; AHMAD, N. Generalization of adaptive neuro-fuzzy inference systems. *IEEE Transactions on Neural Networks*, v. 11, n. 6, p. 1332–1346, 2000.

BABUSKA, R.; VERBRUGGEN, H. Neuro-fuzzy methods for nonlinear system identification. *Annual Reviews in Control*, v. 27, p. 73–85, 2003.

BARALDI, A.; BLONDA, P. A survey of fuzzy clustering algorithms for pattern recognition - part i. *IEEE Transactions on Systems, Man and Cybernetics*, B-29, n. 6, p. 778–785, 1999.

BARRETO, G. et al. Nonstationary time series prediction using local models based on competitive neural networks. *Lecture Notes in Computer Science*, v. 3029, p. 1146–1155, 2004.

BARRETO, G. A.; ARAÚJO, A. F. R. Identification and control of dynamical systems using the self-organizing map. *IEEE Transactions on Neural Networks*, v. 15, n. 5, p. 1244–1259, 2004.

BARRETO, G. A.; ARAÚJO, A. F. R.; RITTER, H. J. Self-organizing feature maps for modeling and control of robotic manipulators. *Journal of Intelligent and Robotic Systems*, v. 36, n. 4, p. 407–450, 2003.

BARRETO, G. A. et al. Previsão de séries temporais não-estacionárias usando modelos locais baseados em redes neurais competitivas. In: *Anais do VI Simpósio Brasileiro de Automação Inteligente (SBAI'03)*. [S.l.: s.n.], 2003. p. 941–946.

BARRETO, G. A.; SOUZA, L. G. M. Adaptive filtering with the self-organizing maps: A performance comparison. *Neural Networks*, v. 19, n. 6, p. 785–798, 2006.

BAUER, H.-U.; VILLMANN, T. Growing a hypercubical output space in a self-organizing feature map. *IEEE Transactions on Neural Networks*, v. 8, n. 2, p. 218–226, 1997.

BILLINGS, S. A.; VOON, W. S. F. Structure detection and model validity tests in the identification of nonlinear systems. *IEE Proceedings, Part D*, v. 130, n. 4, p. 193–199, 1983.

BILLINGS, S. A.; VOON, W. S. F. Correlation based model validity tests for nonlinear models. *International Journal of Control*, v. 44, n. 1, p. 235–244, 1986.

BILLINGS, S. A.; ZHU, Q. M. Nonlinear model validation using correlation tests. *International Journal of Control*, v. 60, n. 6, p. 1107–1120, 1994.

BITTANTI, S.; PIRODDI, L. Nonlinear identification and control of a heat exchanger: A neural network approach. *Journal of the Franklin Institute*, v. 334, n. 1, p. 135–153, 1997.

BOHLIN, T. On the problem of ambiguities in maximum likelihood identification. *Automatica*, p. 199–210, 1971.

BOX, G. E. P.; JENKINS, G. M. *Time Series Analysis, Forecasting and Control*. [S.l.]: Holden-Day, San Francisco, 1976.

CAUDELL, T. P.; NEWMAN, D. S. An adaptive resonance architecture to define normality and detect novelties in time series and databases. In: *Proceedings of the IEEE World Congress on Neural Networks*. [S.l.: s.n.], 1993. p. 166–176.

CHEN, J.-Q.; XI, Y.-G. Nonlinear system modeling by competitive learning and adaptive fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics-Part C*, v. 28, n. 2, p. 231–238, 1998.

CHIU, S. L. Fuzzy model identification based on cluster estimation. *Journal of Intelligent & Fuzzy Systems*, v. 2, n. 3, p. 267–278, 1994.

CHO, J. et al. Modeling and inverse controller design for an unmanned aerial vehicle based on the self-organizing map. *IEEE Transactions on Neural Networks*, v. 17, n. 2, p. 445–460, 2006.

CHO, J. et al. Quasi-sliding mode control strategy based on multiple linear models. *Neurocomputing*, v. 70, n. 4-6, p. 962–974, 2007.

CHUNG, F.-L.; LEE, T. Fuzzy competitive learning. *Neural Networks*, v. 7, n. 3, p. 539–551, 1994.

DARKEN, C.; MOODY, J. Fast adaptive k-means clustering: Some empirical results. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN'90)*. [S.l.: s.n.], 1990. v. 2, p. 233–238.

DÍAZ-BLANCO, I. et al. Visualization of dynamics using local dynamic modelling with self-organizing maps. *Lecture Notes on Computer Science*, v. 4668, p. 609–617, 2007.

DUNN, J. C. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, v. 3, n. 3, p. 32–57, 1973.

FRITZKE, B. A growing neural gas network learns topologies. In: *Advances in Neural Information Processing Systems (NIPS)*. [S.l.: s.n.], 1994. p. 625–632.

FROTA, R. A. *Avaliação de algoritmos de redes neurais artificiais em tarefas de detecção de novidades: uma abordagem unificadora*. Dissertação (Mestrado) — Departamento de Engenharia de Teleinformática, Universidade de Federal do Ceará, 2005.

HAYKIN, S. *Neural Networks and Learning Machines*. 3th. ed. New Jersey, NJ: Prentice-Hall, 2008.

HEBB, D. O. *The Organization of Behavior*. New York: John Wiley, 1949.

HERTZ, J.; KROGH, A.; PALMER, R. G. *Introduction to the theory of neural computation*. Redwood City, CA: Addison-Wesley, 1991.

HINES, W. W. et al. *Probabilidade e Estatística na Engenharia*. 4o. edição. ed. [S.l.]: LTC Editora, 2006.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, v. 2, n. 5, p. 359–366, 1989.

HUANG, G. B.; ZHU, Q. Y.; ZIEW, C. K. Extreme learning machine: Theory and applications. *Neurocomputing*, v. 1–3, n. 70, p. 489–501, 2006.

HUNT, K. J. et al. Neural networks for control systems: a survey. *Automatica*, v. 28, n. 6, p. 1083–1111, 1992.

HUSSAIN, M. A. *Inverse model control strategies using neural networks: Analysis, simulation and on-line implementation*. Tese (Doutorado) — University of London, 1996.

HUSSAIN, M. A.; KERSHENBAUM, L. S. Implementation of an inverse-model-based control strategy using neural networks on a partially simulated exothermic reactor. *Chemical Engineering Research and Design*, v. 78, n. 2, p. 299–311, 2000.

HWANG, J. N. et al. The past, present, and future of neural networks for signal processing. *IEEE Signal Processing Magazine*, v. 14, n. 6, p. 28–48, 1997.

KARAYIANNIS, N. B.; PAI, P.-I. Fuzzy vector quantization algorithms. In: *Proceedings of the 3rd IEEE Conference on Fuzzy systems*. [S.l.: s.n.], 1994. v. 3, p. 1996–2001.

KOHONEN, T. Clustering, taxonomy, and topological maps of patterns. In: *Proceedings of the 6th International Conference on Pattern Recognition (6ICPR)*. Washington, DC: IEEE Computer Soc. Press., 1982. p. 114–128.

KOHONEN, T. K. The self-organizing map. *Neurocomputing*, v. 21, p. 1–6, 1998.

KOHONEN, T. K. *Self-Organizing Maps*. 3rd. ed. Berlin, Heidelberg: Springer-Verlag, 2001.

KOHONEN, T. K. et al. Engineering applications of the self-organizing map. *Proceedings of the IEEE*, v. 84, n. 10, p. 1358–1384, 1996.

KOSKO, B. *Neural Networks and Fuzzy Sytems: A Dynamical Systems Approach to Machine Intelligence*. [S.l.]: Prentice-Hall, 1992.

LAWRENCE, S.; TSOI, A. C.; BACK, A. D. Function approximation with neural networks and local methods: Bias, variance and smoothness. In: *Australian Conference on Neural Networks (ACNN)*. [S.l.: s.n.], 1996. p. 16–21.

LIGHTBODY, G.; IRWIN, G. W. Nonlinear control structures based on embedded neural system models. *IEEE Transactions on Neural Networks*, v. 8, n. 3, p. 553–567, 1997.

LJUNG, L. *System Identification: Theory for the User*. 2nd. ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: Le Cam, L. M.; NEYMAN, J. (Ed.). *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, Califonia. University of California Press: [s.n.], 1967. v. 1, p. 281–297.

MASULLI, F.; ROVETTA, S. Fuzzy concepts in vector quantization training. In: *Fuzzy Logic and Applications*. [S.l.]: Springer, 2006. LNCS-2955/2006, p. 279–288.

MCCULLOCH, W. S.; PITTS, W. H. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, n. 1, p. 115–133, 1943.

NARENDRA, K. S. Neural networks for control theory and practice. *Proceedings of the IEEE*, v. 84, n. 10, p. 1385–1406, 1996.

NARENDRA, K. S.; LEWIS, F. L. Special issue on neural networks feedback control. *Automatica*, v. 37, n. 8, 2001.

NARENDRA, K. S.; PARTHASARATHY, K. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, v. 1, n. 1, p. 4–27, 1990.

NORGAARD, M. et al. *Neural Networks for Modelling and Control of Dynamic Systems*. [S.l.]: Springer-Verlag, 2000.

OH, S.-K. et al. Heterogeneous local model networks for time series prediction. *Applied Mathematics and Computation*, v. 168, n. 1, p. 1993–2005, 2005.

PRINCIPE, J. C.; EULIANO, N. R.; LEFEBVRE, W. C. *Neural Adaptive Systems: Fundamentals Through Simulations*. New York, NY: John Willey & Sons, 2000.

PRINCIPE, J. C.; WANG, L.; MOTTER, M. A. Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control. *Proceedings of the IEEE*, v. 86, n. 11, p. 2240–2258, 1998.

RIPLEY, B. D. *Pattern Recognition and Neural Networks*. [S.l.]: Cambridge University Press, 1996.

RUBIO, J. de J. SOFMLS: Online self-organizing fuzzy modified least-squares network. *IEEE Transactions on Fuzzy Systems*, v. 17, n. 6, p. 1296–1309, 2009.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. Learning representations by back-propagating errors. *Nature*, v. 323, p. 533–536, 1986.

SJÖBERG, J. et al. Nonlinear black-box modeling in system identification: A unified overview. *Automatica*, v. 31, n. 12, p. 1691–1724, 1995.

SOONG, T. T. *Fundamentals of Probability and Statistics for Engineers*. 1st. ed. West Sussex, England: John Wiley & Sons, 2004.

SOUZA, L. G. M.; BARRETO, G. A. Nonlinear system identification using local arx models based on the self-organizing map. *Learning and Nonlinear Models*, v. 4, n. 2, p. 112–123, 2006.

SOUZA, L. G. M.; BARRETO, G. A. On building local models for inverse system identification with vector quantization algorithms. *Neurocomputing*, v. 73, n. 10-12, p. 1993–2005, 2010.

SUGENO, M.; YASUKAWA, M. A fuzzy logic based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems*, v. 1, p. 7–31, 1993.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, v. 15, p. 116–132, 1985.

VASUKI, A.; VANATHI, P. T. A review of vector quantization techniques. *IEEE Potentials*, v. 25, n. 4, p. 39–47, 2006.

WADSWORTTH, G. P.; BRYAN, J. G. *Application of Probality and Random variables*. [S.l.]: Mc Graw-Hill, New York, 1974.

WALTER, J.; RITTER, H.; SCHULTEN, K. Non-linear prediction with self-organizing map. In: *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'90)*. [S.l.: s.n.], 1990. v. 1, p. 587–592.

YAGER, R. R.; FILEV, D. P. *Essentials of Fuzzy Modeling and Control*. New York, NY: Willey, 1994.

ZADEH, L. A. Fuzzy sets. *Information and Control*, n. 8, p. 338–353, 1965.

ZAKNICH, A. *Neural networks for intelligent signal processing*. [S.l.]: World Cientific, 2003.