Universidade Federal de Minas Gerais - Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Alisson Marques da Silva

Sistemas Neuro-Fuzzy Evolutivos: Novos Algoritmos de Aprendizado e Aplicações

Belo Horizonte, MG
2014

Alisson Marques da Silva

Sistemas Neuro-Fuzzy Evolutivos: Novos Algoritmos de Aprendizado e Aplicações

Tese de Doutorado submetida à banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Engenharia Elétrica.

Área de Concentração: Sistemas de Computação e Telecomunicações
Linha de Pesquisa: Inteligência Computacional

Orientador: Prof. Walmir Matos Caminhas
Co-orientador: Prof. André Paim Lemos

Universidade Federal de Minas Gerais - Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Belo Horizonte, MG
2014

"Sistemas Neuro-fuzzy Evolutivos: Novos Algoritmos de Aprendizado e Aplicações"

Alisson Marques da Silva

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 10 de abril de 2014.

Por:

Prof. Dr. Walmir Matos Caminhas
DELT (UFMG) - Orientador

Prof. Dr. André Paim Lemos
DELT (UFMG)

Prof. Dr. Ivan Nunes da Silva
DEE (USP/São Carlos)

Prof. Dr. Fernando Antônio Campos Gomide
DCA (UNICAMP)

Prof. Dr. Marcos Flávio Silveira Vasconcelos DAngelo
DCC (UNIMONTES)

Prof. Dr. Eduardo Mazoni Andrade Marçal Mendes
DELT (UFMG)

Dr. Daniel Furtado Leite
PPGEE (UFMG)

Resumo

Este trabalho tem como objetivo propor novos algoritmos de aprendizado incremental para sistemas *fuzzy* evolutivos visando aplicações em sistemas de tempo real com dinâmica rápida e algoritmos evolutivos com seleção adaptativa de entradas. As técnicas propostas buscam construir algoritmos eficientes e com baixo custo computacional. Com foco nesses objetivos, foram propostos quatro algoritmos de aprendizado para a estrutura da rede *neuro-fuzzy Neo-Fuzzy-Neuron* (NFN). Basicamente, a rede NFN é definida como um conjunto de n modelos desacoplados do tipo Takagi-Sugeno de Ordem Zero, um por variável de entrada, cada um contendo m regras. A estrutura da rede favorece a utilização de algoritmos recursivos eficientes para a atualização dos parâmetros e possibilita inclusão/exclusão de regras *fuzzy* e variáveis de entrada durante o aprendizado. O primeiro algoritmo (eNFN) utiliza uma abordagem de aprendizado incremental que, simultaneamente, granulariza o domínio das variáveis de entrada e atualiza os pesos dos modelos de saída de forma incremental. Inicialmente, são definidas duas regras *fuzzy* para cada entrada e, em função do erro calculado para um fluxo de dados, regras podem ser adicionadas, excluídas e/ou ter seus parâmetros ajustados. O segundo algoritmo (NFN-SAE) consiste num mecanismo incremental de seleção adaptativa de entradas, permitindo a inclusão ou exclusão de variáveis de entrada por meio de um teste estatístico, calculado recursivamente, que leva em consideração a acurácia e a complexidade do modelo. O terceiro (eNFN-SAE) e o quarto (X-eNFN-SAE) algoritmos utilizam a técnica para evolução da estrutura introduzida no primeiro algoritmo e o método de seleção adaptativa de entradas do segundo algoritmo. No eNFN-SAE o processo de granulação do domínio das variáveis de entrada e o cálculo do grau de ativação das funções de pertinência são computados somente uma vez para cada amostra do fluxo de dados. Ao contrário, no X-eNFN-SAE, estes cálculos são realizados uma vez para o modelo atual e uma vez para cada modelo candidato, a cada amostra do fluxo de dados. O desempenho dos algoritmos desenvolvidos é avaliado na identificação de processos não lineares e em problemas de previsão. Resultados computacionais mostram que as abordagens propostas possuem um desempenho comparável ao desempenho de abordagens alternativas tratadas na literatura, e possuem um menor custo computacional - fator importante para modelagem *on-line* de processos. Além disso, o eNFN foi avaliado experimentalmente em tempo real na modelagem de um sistema de levitação magnética (Maglev) e de um sistema MIMO de duplo rotor (TRMS), ambos com alta taxa de amostragem (dinâmica rápida). Os resultados experimentais mostram que o eNFN é capaz de capturar o comportamento do sistema dinâmico com precisão e baixo custo computacional. Os resultados experimentais mostram que o eNFN alcança boa precisão, e sugerem a rede *neuro-fuzzy* como uma abordagem eficaz para estimar em tempo real as variáveis de estado de processos dinâmicos rápidos e não lineares.

Palavras-chave: Sistemas *Neuro-Fuzzy* Evolutivos, *Neo-Fuzzy-Neuron*, Modelagem Adaptativa, Seleção Adaptativa de Entradas.

Abstract

This work introduces incremental learning algorithms for evolving fuzzy models with applications in real-time systems with fast dynamics and algorithms with adaptive feature selection. The focus is on efficient and low computational cost algorithms. Four incremental learning algorithms were developed using the Neo-Fuzzy-Neuron (NFN) neuro-fuzzy network. Basically, the NFN is a set of n decoupled zero-order Takagi-Sugeno models, one for each input variable, each one containing m rules. The network structure favors the use of efficient recursive algorithms for parameter update and the inclusion/exclusion of fuzzy rules and input variables during incremental learning. The first algorithm (eNFN) uses an incremental learning approach that simultaneously granulates the input space and updates the weights of output models. Initially, two fuzzy rules are defined for each input and, depending on an error measure computed based on a data stream, rules can be added, deleted and/or have their parameters adjusted. The second algorithm (NFN-AFS) is an incremental mechanism for adaptive feature selection that allows inclusion/exclusion of input variables using a statistical test that considers the accuracy and complexity of the model, computed recursively. The third (eNFN-AFS) and fourth (X-eNFN-AFS) algorithms use an approach to evolve the network structure introduced in the eNFN and the mechanism to adaptive feature selection of the NFN-AFS. In eNFN-AFS the granulation of the domain of the input variables and the computation of the degree of activation of the membership functions are performed only once at each step, i.e. there is a unique set of membership functions for each input variable. Unlike of the eNFN-AFS, in X-eNFN-AFS granulation the domain of the input variables and computation of the degree of activation of the membership functions are performed for the current model and for each candidate model, i.e. each model (current and candidate) has a set of membership functions for each one of its input variable, at each step. The performance of the algorithms is evaluated considering nonlinear processes identification and times series forecasting problems. Computational experiments and comparisons against alternative evolving models show that the approaches introduced here are accurate and fast, characteristics suitable for adaptive systems modeling, especially in real-time, on-line environments. Furthermore, the eNFN was evaluated experimentally in real-time to model two actual systems (Magnetic Levitation System (MagLev) and Twin Rotor MIMO System (TRMS)) using high sampling rate. Experimental results show that the eNFN is capable to quickly capture the behavior of the system dynamics, and to model the system with precision and low computational complexity. Experimental results also show that the eNFN attains good accuracy, and suggest the neuro-fuzzy network as an effective approach to estimate state variables of fast nonlinear dynamic processes in real-time.

Keywords: Evolving Neuro-Fuzzy Systems, Neo-Fuzzy-Neuron, Adaptive Modeling, Adaptive Feature Selection.

Agradecimentos

Primeiramente, a Deus por me proporcionar mais essa oportunidade de aprendizado e por me guiar pelos caminhos da vida.

Aos meus orientadores, Profs. Walmir Matos Caminhas e André Paim Lemos, agradeço pela oportunidade, incentivo, pelas importantes orientações e contribuições na condução deste trabalho. Pela paciência em entender minhas dificuldades pessoais. Com certeza, o que aprendi com vocês vai muito além da vida acadêmica.

Ao Prof. Fernando Gomide pela disponibilidade em sempre contribuir com minha pesquisa.

Aos Profs. Ivan Nunes, Eduardo Mazoni, Marcos Flávio D' Angelo, Daniel Leite, pelas valiosas contribuições apresentadas na defesa da tese que, com certeza, ajudaram a enriquecer o trabalho.

À minha mãe Madalena pelo apoio de sempre, pelas preciosas palavras de conforto nos momentos difíceis e, principalmente, pela compreensão das ausências mais que constantes.

À minha família, amigos e colegas de trabalho do CEFET-MG - Campus Divinópolis, pelo incentivo e apoio durante esta jornada.

Aos professores e colegas do DIFCOM (Laboratório de Detecção de Falhas, Controle, Otimização e Modelagem) da UFMG. Em especial aos colegas: Alexandre Faria, Guilherme Costa, Maurílio Inácio, por compartilhar as alegrias e também os desesperos no decorrer do doutorado.

A todos os meus professores que contribuirão para a minha formação pessoal e profissional.

À minha namorada Regina pelo apoio nos momentos difíceis, pela compreensão dos momentos distantes, enfim, por me acompanhar durante essa caminhada.

A todos que estiveram ao meu lado e, que mesmo sem saber, de alguma forma me apoiaram e me incentivaram.

Não poderia deixar de agradecer aos revisores anônimos dos meus artigos pela colaboração e sugestões.

A CAPES, pelo apoio financeiro.

Dedico este trabalho a Deus e a minha mãe, pelo exemplo de vida, força e determinação.

”Seja você quem for, seja qual for a posição
social que você tenha na vida, a mais alta ou
a mais baixa, tenha sempre como meta muita força,
muita determinação e sempre faça tudo com muito amor
e com muita fé em Deus, que um dia você chega lá.
De alguma maneira você chega lá”.
Ayrton Senna

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

ANFIS - *Adaptive Neuro-Fuzzy Inference Systems*

AP - Aprendizado Participativo

CMAC - *Cerebellar Model Articulation Controller*

DENFIS - *Dynamic Evolving Neuro-Fuzzy Inference System*

DS - *Datum Significance*

eClass - Classificador *fuzzy* evolutivo

ECM - *Evolving Clustering Method*

eF-OP-ELM - *evolving Fuzzy Optimally Pruned Extreme Learning Machine*

eFPT - *evolving Fuzzy Pattern Tree*

EFS - *Evolving Fuzzy Systems*

eFSM - *evolving Neuro-Fuzzy Semantic Memory*

eFT - *evolving Fuzzy Linear Regression Trees*

eFuMo - *evolving Fuzzy Model*

EFuNNs - *Evolving Fuzzy Neural Networks*

eMG - *Multivariable Gaussian Evolving Fuzzy Modeling System*

eNFN - *evolving Neo-Fuzzy-Neuron*

eNFN-SAE - eNFN com Seleção Adaptativa de Entradas

ePL - *evolving Participatory Learning*

ePL+ - Versão do *evolving Participatory Learning*

ERS - *Extended Rule Significance*

ESAFIS - *Extended Sequencial Adaptive Fuzzy Inference System*

eT2FIS - *evolving Type-2 Neural Fuzzy Inference System*

eTS - *evolving* Takagi-Sugeno

eTS+ - Versão do *evolving* Takagi-Sugeno

eTS-LS-SVM - *evolving Takagi-Sugeno Least Square Support Vector Machine*

eVQ - *evolving Vector Quantization*

FCM - *Fuzzy C-Means*

FDT - *Fuzzy Decision Trees*

FLEXFIS - *Flexible Fuzzy Inference System*

FLEXFIS++ - *Flexible Evolving Fuzzy Inference System*

FLEXFIS-Class - Classificador FLEXFIS

FLNN - *Functional Link Neural Network*

GART+ - *Generalized Adaptive Resonance Theory+*

GENEFIS - *Generic Evolving Neuro-Fuzzy Inference System*

IRSFNN - *Interactively Recurrent Self-Evolving Fuzzy Neural Network*

LARS - *Least Angle Regression*

Log - Logaritmo Natural

LS-SVM - *Local Least Squares Support Vector Machine Models*

MagLev - *Magnetic Levitation System*

MIMO - *Multiple Input / Multiple Output*

MLP - *Multi-Layer Perceptron*

Mod_eTS - Versão do *evolving* Takagi-Sugeno

MRIT2NFS - *Mutually Recurrent Interval Type-2 Neural Fuzzy System*

NFN - *Neo-Fuzzy-Neuron*

NFN-SAE - *Neo-Fuzzy-Neuron* com Seleção Adaptativa de Entradas

PRESS - *PREdiction Sum of Squares*

RBF - *Radial Basis Function*

rFCM - *recursive Fuzzy C-Means*

RGK - *Recursive Gustafson-Kessel*

RMSE - *Root Mean Squared Error*

SAFIS - *Sequential Adaptive Fuzzy Inference System*

SELM - *Structure Evolved Learning Method*

Simpl_eClass - *Simple eClass*

Simpl_eTS - *Simple evolving* Takagi-Sugeno

Simpl_eTS+ - Versão do *Simple evolving* Takagi-Sugeno

SOFMLS - *Self-Organizing Fuzzy Modified Least-Squares Network*

SQR - Soma dos Quadrados Residuais

SSEM - *Simplified Structure Evolving Method*

TS - Takagi-Sugeno

TSCIT2FNN - *TSK-Type-Based Self-Evolving Compensatory Interval Type-2 Fuzzy Neural Network*

TRMS - *Twin Rotor MIMO System*

X-eNFN-SAE - *eXtended* eNFN-SAE

xTS - *eXtended* Takagi-Sugeno

Lista de Símbolos

A - Função de pertinência e/ou conjunto *fuzzy*

a - Limite inferior de uma função de pertinência triangular

$ativa$ - Tempo t em que uma função de pertinência foi ativada

b - Valor modal (centro) de uma função de pertinência triangular

b^* - Índice da função de pertinência mais ativa

b^- - Índice da função de pertinência menos ativa

c - Limite superior de uma função de pertinência triangular

$dist$ - Distância entre o valor modal da função criada e o valor modal das funções adjacentes

E - Erro quadrático

e - Erro residual

F - Teste F

F_{exc} - Teste F para modelos candidatos à exclusão de variáveis de entrada

F_{inc} - Teste F para modelos candidatos à inclusão de variáveis de entrada

i - Índice para as variáveis de entrada

$idade$ - Tempo de inatividade de uma função de pertinência

j - Índice para as funções de pertinência

k_i - Índice para as funções de pertinência ativas

l - Índice para as saídas

m - Número de funções de pertinência

max - Valor máximo

min - Valor mínimo

ms - Milissegundo

n - Número de variáveis de entrada

n_c - Número de modelos candidatos

n_m - Número total de modelos (soma do número de modelos candidatos com o modelo atual)

N - Número de amostras para estimar os parâmetros dos modelos

N_a - Número de amostras para estimar os parâmetros do modelo atual

N_c - Número de amostras para estimar os parâmetros do modelo candidato

p_a - Número de parâmetros do modelo atual

p_c - Número de parâmetros do modelo candidato

q - Parâmetros (pesos) das redes NFN (NFN, eNFN, NFN-SAE, eNFN-SAE, X-eNFN-SAE)

R - Regra *fuzzy*

r - Número de regras

s - Número de saídas

std - Desvio padrão

SQR - Soma dos Quadrados Residuais

t - Instância de tempo atual - amostra de dados atual

x - Padrão (dado) de entrada

y - Saída estimada

$\hat{y}$ - Saída desejada, saída medida

z - Índice para o melhor modelo candidato

α - Taxa de aprendizado (tamanho do passo no algoritmo do gradiente)

β - Taxa de aprendizado para redes eNFN

γ - Soma da multiplicação do grau de ativação das funções de pertinência pelos seus respectivos pesos

Δ - Distância entre os centros (valores modais) das funções de pertinência

η - Parâmetro auxiliar na criação de funções de pertinência

κ - Nível de significância

λ - Número de vezes que uma mesma hipótese é testada

μ_A - Grau de ativação da função de pertinência

$\hat{\mu}_{b^*}$ - Média do erro local correspondente a função de pertinência mais ativa

$\hat{\mu}_g$ - Média do erro global

$\mu\hat{\mu}_g$ - Média do erro local correspondente a função de pertinência mais ativa para rede com múltiplas saídas

ν - Soma do grau de ativação das funções de pertinência

$\hat{\sigma}_g^2$ - Variância do erro global da rede

$\sigma\hat{\sigma}_g^2$ - Média da variância do erro global para rede com múltiplas saídas

τ - Limitador da menor distância permitida na criação de funções de pertinência

ω - Limiar para exclusão de funções de pertinência por tempo de inatividade

Capítulo 1

Introdução

Neste capítulo inicial, apresentam-se, na próxima seção, a evolução dos sistemas *fuzzy* e os principais desafios no desenvolvimento destes sistemas. Esses desafios norteiam as motivações para o desenvolvimento deste trabalho. A Seção 1.2 mostra o objetivo geral e os objetivos específicos a serem alcançados. As principais contribuições desta tese são descritas na Seção 1.3. Em seguida, na Seção 1.4, são listados os artigos frutos do desenvolvimento deste trabalho. Por fim, na Seção 1.5, é descrito sucintamente o conteúdo de cada um dos capítulos.

1.1 Motivação e Relevância

Os sistemas *fuzzy* e suas variações têm sido amplamente utilizados em atividades nas quais são necessárias capacidade de raciocínio semelhantes à dos seres humanos. Nos primeiros sistemas *fuzzy*, o conhecimento para modelagem do sistema era adquirido por meio de especialistas (Mamdani and Assilian, 1999). A partir da década de 1990, a modelagem desses sistemas passou a ser realizada com base em informações extraídas de um conjunto de dados, sendo o conhecimento do especialista, se necessário, utilizado como informação complementar (Mallinson and Bentley, 1999; Kasabov and Filev, 2006; Angelov et al., 2008a). Em ambas abordagens os sistemas são estáticos, apenas os parâmetros dos sistemas são atualizados durante o treinamento (Tung et al., 2013).

Atualmente, existe um aumento da necessidade de processar uma quantidade crescente de informações de forma eficiente para a extração do conhecimento para modelagem de sistemas complexos (Maciel et al., 2012c). Geralmente, as informações são disponibilizadas na forma de fluxos de dados, vêm em grandes quantidades, e são altamente dinâmicas e não estacionárias. Nessas condições a modelagem baseada em conhecimento especialista ou baseada em dados, onde o sistema possui uma estrutura fixa, pode não ser adequada. O re-treinamento dos modelos normalmente não é uma opção viável, pois os fluxos de dados chegam continuamente (Bouchachia et al., 2014). Assim, recentemente, foi criado um campo de pesquisa emergente que aborda esse problema usando metodologias que possuem mecanismos para treinar e atualizar continuamente os modelos de maneira *on-line*, para que estes se adaptem a dinâmica das mudanças presentes no fluxo de dados. Os sistemas *fuzzy* com essa habilidade são chamados sistemas *fuzzy* evolutivos (*Evolving Fuzzy Systems* - EFS).

Essencialmente, os EFS possuem a capacidade de desenvolver gradualmente sua estrutura (aumentar e/ou diminuir) simultaneamente ao ajuste dos parâmetros a partir de um fluxo de dados. Isto proporciona um alto nível de adaptação com um aprendizado incremental e contínuo (Maciel et al., 2012a,d). O aprendizado incremental permite um processamento rápido e possui um baixo consumo de memória, uma vez que as amostras do fluxo de dados são processadas uma única vez e, então, são descartadas (Lughofer, 2013), isto é, o mecanismo de aprendizagem utiliza apenas a amostra de dados atual (Tung et al., 2013). Esse processo de aprendizagem permite a evolução da estrutura do sistema para incorporar novos conhecimentos e possibilita aprendizagem ao "longo da vida". Enquanto o aprendizado permite a aquisição contínua de novos conhecimentos modificando a estrutura e os parâmetros do sistema, este deve ser capaz de manter/armazenar o conhecimento relevante aprendido previamente (Tung et al., 2013). O aprendizado incremental é aplicável, principalmente, no contexto em que existe fluxos de dados e em ambiente não estacionário evolvendo condições ambientais voláteis (Lughofer, 2013).

Um princípio fundamental dos EFS é a estrutura flexível, onde a base de regras é adaptável e evolui de acordo com as características do fluxo de dados de entrada. Ele pode iniciar seu processo de aprendizagem a partir de uma base de regras vazia e novas regras podem ser criadas e/ou eliminadas a partir de uma determinada medida de qualidade dos dados. O processo de aprendizagem é *on-line*, incremental e com uma única passagem do fluxo de dados (Pratama et al., 2013a).

Os algoritmos associados a sistemas *fuzzy* evolutivos definem uma abordagem promissora para a construção de modelos não lineares e adaptativos. Segundo Angelov et al. (2008a), existe uma crescente demanda para implementação dos sistemas *fuzzy* evolutivos aplicados a diversos domínios. Sensores inteligentes, sistemas autônomos, controle adaptativo, detecção e diagnóstico de falhas são exemplos de aplicações industriais (Angelov et al., 2008b).

As principais tendências em sistemas *fuzzy* evolutivos estão na criação de técnicas e metodologias para construção de sistemas que tenham alto grau de adaptabilidade e autonomia, desenvolvidos a partir de um fluxo de dados coletado *on-line*, eventualmente em tempo real (Angelov et al., 2008b, 2010; Pratama et al., 2013a). A adaptabilidade e a autonomia são obtidas pela modificação dos parâmetros e da estrutura do sistema à medida que ocorrem alterações nos dados de entrada. Outra tendência é a incorporação de métodos de seleção de variáveis de entrada do modelo e sua inserção na metodologia de aprendizagem evolutiva (Zhu et al., 2010) para capturar variações no processo, seus diferentes modos de operação, dentre outros (Lughofer, 2011). Em particular, uma das questões é integrar métodos incrementais de seleção de variáveis (Li, 2004) no mecanismo de adaptação sem causar descontinuidades no aprendizado (Lughofer, 2011).

Os modelos evolutivos presentes na literatura podem apresentar problemas em aplicações de tempo real que necessitam de uma resposta rápida do sistema (sistemas de tempo real com dinâmica rápida), devido ao seu custo computacional. Para esse tipo de sistema são necessários modelos que tenham uma boa precisão e, principalmente, que possuam uma estrutura compacta, com equações simples e baixo custo computacional. Partindo dessa necessidade, este trabalho busca desenvolver algoritmos de aprendizado evolutivo tendo como foco a sua aplicação em sistemas de tempo real com dinâmica rápida e algoritmos evolutivos com métodos de seleção adaptativa de entradas.

1.2 Objetivos: geral e específicos

1.2.1 Objetivo Geral

O objetivo deste trabalho é propor novos algoritmos evolutivos de aprendizado para aplicações em sistemas de tempo real com dinâmica rápida e algoritmos evolutivos com seleção adaptativa de entradas. Deseja-se algoritmos que tenham:

- baixo custo computacional;
- estrutura parcimoniosa, i. e., baixo número de parâmetros livres;
- alta precisão.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Desenvolver algoritmo de aprendizado incremental com baixo custo computacional para a rede *neuro-fuzzy Neo-Fuzzy-Neuron* (NFN) (Yamakawa et al., 1992).
- Desenvolver algoritmo de aprendizado incremental incorporado a um método de seleção adaptativa de entradas para a estrutura da NFN.

1.3 Contribuições da Tese

As principais contribuições desta tese são descritas a seguir:

- Proposição de um algoritmo evolutivo com baixo custo computacional para a rede *neuro-fuzzy Neo-Fuzzy-Neuron* (NFN). No algoritmo proposto a estrutura da rede NFN evolui simultaneamente com os ajustes dos pesos por meio da inclusão e/ou exclusão de funções de pertinência.
- Realização de simulações computacionais e testes experimentais em tempo real para avaliar a rede NFN evolutiva. Geralmente, em modelagem/identificação de sistemas, os testes são realizados com dados reais, mas não em tempo real (Rahideh and Shaheed, 2008; Subudhi and Jena, 2009; Nejjari et al., 2012).
- Proposição de um método de seleção adaptativa de entradas para a rede *Neo-Fuzzy-Neuron*. O método de seleção adaptativa de entradas foi empregado no desenvolvimento de três algoritmos, sendo um para a rede NFN original e dois para a rede NFN evolutiva.
- Avaliação da complexidade espacial, temporal e do tempo de processamento dos algoritmos propostos e a comparação com modelos alternativos.
- Proposição de uma nova abordagem para estimar os parâmetros e avaliar o desempenho dos modelos evolutivos.

- Proposição do uso da Soma dos Quadrados Residuais (SQR) para analisar o comportamento do erro dos modelos evolutivos ao longo do tempo.

1.4 Publicações

Esta seção apresenta as publicações produzidas no decorrer desta pesquisa. Em resumo, foram publicados sete artigos, sendo um em periódico e seis em congressos. Além disso, foram submetidos três artigos para congressos e dois artigos estão em processo de finalização e serão submetidos para periódicos.

1.4.1 Periódicos

1. SILVA, A. M.; CAMINHAS, W.; LEMOS, A.; GOMIDE, F.. **A Fast Learning Algorithm for Evolving Neo Fuzzy Neuron**. Applied Soft Computing, 2014. (Publicado). (Silva et al., 2014c).

2. SILVA, A. M.; CAMINHAS, W.; LEMOS, A.; GOMIDE, F.. **Real-Time Non-Linear Modelling of a Twin Rotor MIMO System and of a Magnetic Levitation System Using Evolving Neuro-Fuzzy Network**. (Em desenvolvimento).

3. SILVA, A. M.; INACIO, M. J.; CAMINHAS, W.; LEMOS, A.; GOMIDE, F.. **Evolving Fuzzy Systems: A Review and Perspectives**. (Em desenvolvimento).

O primeiro artigo descreve a rede *Neo-Fuzzy-Neuron* evolutiva (eNFN) proposta no Capítulo 4. A estrutura da eNFN evolui pela alteração dos parâmetros das funções de pertinência e pela inclusão e/ou exclusão de funções de pertinência. O segundo artigo ilustra a aplicação da rede eNFN na identificação/modelagem em tempo real de um sistema de levitação magnética e de um sistema MIMO de duplo rotor. O terceiro artigo é uma revisão da literatura sobre sistemas *fuzzy* evolutivos. Apresenta-se o estado da arte, os desafios da pesquisa, e as perspectivas no desenvolvimento destes sistemas.

1.4.2 Congressos Internacionais

1. SILVA, A.; CAMINHAS, W.; LEMOS, A.; GOMIDE, F.. **Evolving Neural Fuzzy Network with Adaptive Feature Selection**. In: ICMLA - 11th International Conference on Machine Learning and Applications, 2012. (Publicado). (Silva et al., 2012a).

2. SILVA, A. M.; CAMINHAS, W.; LEMOS, A.; GOMIDE, F.. **Evolving Neo-Fuzzy Neural Network with Adaptive Feature Selection**. In: BRICS-CCI - 1st BRICS Countries, 2013. (Publicado). (Silva et al., 2013a).

3. SILVA, A. M.; CAMINHAS, W.; LEMOS, A.; GOMIDE, F.. **Extended Approach for Evolving Neo-Fuzzy Neural with Adaptive Feature Selection**. In: FLINS - 11th International FLINS Conference on Decision Making and Soft Computing, 2014. (Aceito para publicação). (Silva et al., 2014b).

Estes artigos apresentam as três redes *Neo-Fuzzy-Neuron* com seleção adaptativa de entradas propostas no Capítulo 6. No primeiro artigo é proposta uma nova metodologia para alteração do valor modal das funções de pertinência e a evolução da estrutura da rede (Seção 6.3) se dá pela inclusão e/ou exclusão de variáveis de entrada. No segundo a rede (Seção 6.4 - Algoritmo 6) evolui não só pela inclusão/exclusão de variáveis de entrada, mas também pela inclusão/exclusão de funções de pertinência. No terceiro artigo apresenta-se a rede X-eNFN-SAE (Seção 6.4 - Algoritmo 7). Esta rede estende a apresentada no segundo artigo. A rede X-eNFN-SAE provê uma maior independência na evolução da estrutura dos modelos.

1.4.3 Congressos Nacionais

1. SILVA, A. M. ; CAMINHAS, W. M. ; LEMOS, A. P. . **Uma Breve Revisão de Sistemas Nebulosos Evolutivos**. In: I CBSF - I Congresso Brasileiro de Sistemas Fuzzy, 2010. (Publicado). (Silva et al., 2010).

2. SILVA, A. M. ; CAMINHAS, W. M. ; LEMOS, A. P. ; GOMIDE, F. C. . **Modelo Nebuloso Evolutivo para Sistemas de Tempo Real e Dinâmica Rápida**. In: XIX CBA - XIX Congresso Brasileiro de Automática, 2012. (Publicado). (Silva et al., 2012c).

3. SILVA, A. M. ; CAMINHAS, W. M. ; LEMOS, A. P. ; GOMIDE, F. C. . **Modelo Nebuloso Evolutivo com Seleção Adaptativa de Entradas**. In: II CBSF - II Congresso Brasileiro de Sistemas Fuzzy, 2012. (Publicado). (Silva et al., 2012b).

4. SILVA, A. M.; CAMINHAS, W.; LEMOS, A.; GOMIDE, F.. **Rede Neuro-Fuzzy Evolutiva para Estimação em Tempo Real de Posição de um Sistema de Levitação Magnética**. In: XI SBAI - XI Simpósio Brasileiro de Automação Inteligente, 2013. (Publicado). (Silva et al., 2013b).

5. SILVA, A. M.; CAMINHAS, W.; LEMOS, A.; GOMIDE, F.. **Estimação em Tempo Real de Variáveis de Estado do TRMS com Rede Neuro-Fuzzy Evolutiva**. In: XX CBA - XX Congresso Brasileiro de Automática, 2014. (Submetido). (Silva et al., 2014a).

6. SILVA, A. M.; ROSA, R.; CAMINHAS, W.; LEMOS, A.; GOMIDE, F.. **Rede Neuro-Fuzzy Evolutiva com Seleção de Entradas Aplicada na Modelagem de Sistemas**. In: II CBSF - II Congresso Brasileiro de Sistemas Fuzzy, 2014. (Submetido). (Silva et al., 2014d).

O primeiro artigo contempla uma revisão das recentes pesquisas e desenvolvimentos em sistemas *fuzzy* evolutivos. O segundo artigo descreve uma versão inicial da rede eNFN introduzida no Capítulo 4. Nesta versão a estrutura da eNFN evolui pela alteração dos parâmetros das funções de pertinência e pela inclusão de novas funções de pertinência. O terceiro artigo apresenta a rede *Neo-Fuzzy-Neuron* com seleção adaptativa de entradas proposta na Seção 6.3. No quarto artigo apresenta-se uma aplicação em tempo real da rede eNFN (Capítulo 4) na estimação da posição de uma esfera em um sistema de levitação magnética. O quinto artigo ilustra

a aplicação da rede eNFN na estimação em tempo real de variáveis de estado de um sistema de duplo rotor. O sexto artigo descreve a rede com seleção adaptativa de entradas X-eNFN-SAE (Seção 6.4 - Algoritmo 7) e sua aplicação em problemas de previsão e identificação de sistemas.

1.4.4 Prêmios e Títulos

O artigo *Evolving Neural Fuzzy Network with Adaptive Feature Selection* (Primeiro artigo da Seção 1.4.2) recebeu o prêmio *Best Special Session Paper Award - Adaptive and Dynamic Modeling in Non stationary Environments, ICMLA 2012 - 11th International Conference on Machine Learning and Applications.*

1.5 Organização do Texto

O texto está dividido em 8 capítulos como sumarizado a seguir:

- Este capítulo apresenta a motivação para realização desta pesquisa, os objetivos, as contribuições e as publicações decorrentes deste trabalho.
- O Capítulo 2 discorre sobre os sistemas evolutivos. Nesse capítulo realiza-se uma revisão do estado da arte sobre sistemas *fuzzy* evolutivos, redes *neuro-fuzzy* evolutivas e modelos evolutivos incorporados a métodos de seleção adaptativa de entradas.
- O Capítulo 3 revisa a estrutura da rede *Neo-Fuzzy-Neuron* que será utilizada como base para os algoritmos evolutivos propostos nos capítulos 4 e 6.
- O Capítulo 4 introduz o primeiro algoritmo de aprendizado proposto neste trabalho. Inicialmente, realiza-se uma breve descrição do algoritmo, seguida pelo processo de inicialização das funções de pertinência, o método de adaptação de contexto, a metodologia de atualização do valor modal das funções de pertinência, e a abordagem proposta para a inserção e exclusão de funções de pertinência. Por fim, tem-se a análise da complexidade computacional dos algoritmos.
- O Capítulo 5 ilustra resultados de simulação e experimentais para a rede eNFN proposta no Capítulo 4. O desempenho da rede *neuro-fuzzy* proposta (eNFN) é avaliado e comparado com outros modelos evolutivos na identificação de processo não linear e em problemas de previsão. Nesse capítulo também realizam-se simulações para analisar o tempo de processamento dos modelos. Após as simulações são apresentados experimentos em tempo real utilizando um módulo de levitação magnética e um de duplo rotor.
- O Capítulo 6 apresenta as redes *neuro-fuzzy* incorporadas a um método de seleção adaptativa de entradas. Este começa com uma breve descrição do método proposto, apresenta a abordagem para a seleção de adaptativa de entradas, a definição das funções de pertinência e a adaptação de contexto. São propostos três algoritmos com seleção adaptativa de entradas. Concluindo o capítulo, tem-se uma análise da complexidade computacional dos algoritmos propostos.

- O Capítulo 7 mostra os resultados de simulação para as redes com seleção adaptativa de entradas. As redes são avaliadas por meio de sua acurácia e pelo tempo de processamento na identificação de sistema não linear e em problemas de previsão.
- O Capítulo 8 resume as contribuições desta tese e sugere tópicos para pesquisa futura.

Capítulo 2

Sistemas Evolutivos

Este capítulo apresenta uma revisão da literatura tendo como foco os temas de interesse deste trabalho: redes neurais *fuzzy* e sistemas evolutivos. Uma introdução aos sistemas evolutivos é a ênfase da Seção 2.1. A revisão da literatura, Seção 2.2, está dividida em: Sistemas *Fuzzy* Evolutivos (Subseção 2.2.1); Sistemas *Neuro-Fuzzy* Evolutivos (Subseção 2.2.2) e; Sistemas Evolutivos com Seleção de Entradas (Subseção 2.2.3). Concluindo, a Seção 2.3 faz as considerações finais.

2.1 Introdução

Sistemas *fuzzy* evolutivos (*Evolving Fuzzy Systems* - EFS) foram tratados inicialmente em Kasabov and Song (1999); Angelov and Buswell (2001); Kasabov (2001); Kasabov and Song (2002); Angelov and Filev (2004) para cobrir uma lacuna metodológica existente no contexto de modelagem adaptativa, tendo em vista o processamento *on-line* de dados. Em particular, sistemas *fuzzy* evolutivos têm sido empregados com sucesso para tratar problemas de modelagem (Angelov and Buswell, 2002; Cernuda et al., 2011, 2012), controle (Smith and Tighe, 2005; Barros and Dexter, 2007), classificação (Xydeas et al., 2006; Angelov et al., 2007b,a; Leite et al., 2009, 2010; Lughofer, 2011), identificação, detecção e diagnóstico de falhas (Gomez and Dasgupta, 2002; Gomez and Leon, 2006; Angelov et al., 2008c; Lughofer and Guardiola, 2008; Iglesias et al., 2009; Lemos et al., 2010), previsão e predição (Nguyen et al., 2006; Nayak and Sudheer, 2008; Wang and Vrbanek, 2008; Chang et al., 2009; Lughofer and Kindermann, 2010; Lemos et al., 2011c; Lughofer et al., 2011b,a; Leite et al., 2012).

Os sistemas *fuzzy* evolutivos podem ser considerados como uma sinergia entre sistemas *fuzzy* e métodos recursivos de aprendizado de máquina. Esses sistemas são caracterizados por sua capacidade de extrair conhecimento a partir de dados e adaptar sua estrutura e parâmetros em tempo real, para acompanhar as mudanças do ambiente (Kasabov and Filev, 2006). Segundo Lemos et al. (2009), a modelagem baseada em dados vem sendo amplamente utilizada na construção de modelos capazes de reproduzir comportamentos típicos de sistemas complexos, por meio de um conjunto de amostras dos mesmos.

Sistemas evolutivos são fundamentalmente diferentes de sistemas adaptativos e sistemas evolucionários. Os sistemas adaptativos são capazes de ajustar apenas seus parâmetros e pos-

suem uma estrutura fixa, enquanto os sistemas evolucionários (e. g. algoritmos genéticos (Gomez and Dasgupta, 2002), programação genética (Linden and Bhaya, 2007)) baseiam-se no processo de evolução que ocorre em populações de seres vivos e utilizam, como mecanismo de adaptação, operadores de seleção, cruzamento e mutação de cromossomos. Já os sistemas evolutivos são capazes de ajustar sua estrutura e seus parâmetros em tempo real. Tais sistemas se inspiram no processo de aprendizado humano, mais especificamente, na geração e adaptação do conhecimento a partir de experiências, ou seja, tem como base o processo de evolução de indivíduos ao "longo da vida" (Angelov and Zhou, 2006). Por exemplo, em sistemas baseados em regras, esse processo pode iniciar-se a partir de um conjunto vazio de regras, e novas regras aprendidas à medida que o indivíduo se depara com novas experiências não explicadas pelas regras existentes. As regras não são fixas ou pré-determinadas. A geração das regras é incremental, permitindo que elas possam ser adicionadas, excluídas ou modificadas, para melhor se adaptarem às experiências.

A maioria das técnicas de modelagem *fuzzy* evolutiva utiliza informações sobre a organização espacial das variáveis de entrada e/ou saída para a construção de um conjunto de regras de forma adaptativa (Lemos, 2011). A organização espacial dessas variáveis é feita recursivamente por um algoritmo de agrupamento evolutivo. Este algoritmo processa os dados de entrada recursivamente, adaptando os grupos existentes (Pedrycz, 2005) e eventualmente reestruturando a base de regras. Geralmente, a estrutura do modelo é obtida durante a etapa de agrupamento, pois os componentes principais do modelo (por exemplo, neurônios ou regras *fuzzy*) estão diretamente associados aos grupos (*clusters*). O processo de agrupamento evolutivo pode adicionar, excluir ou modificar os componentes do modelo de forma dinâmica. Um mecanismo típico para atualizar a estrutura do modelo é ilustrado na Figura 2.1.

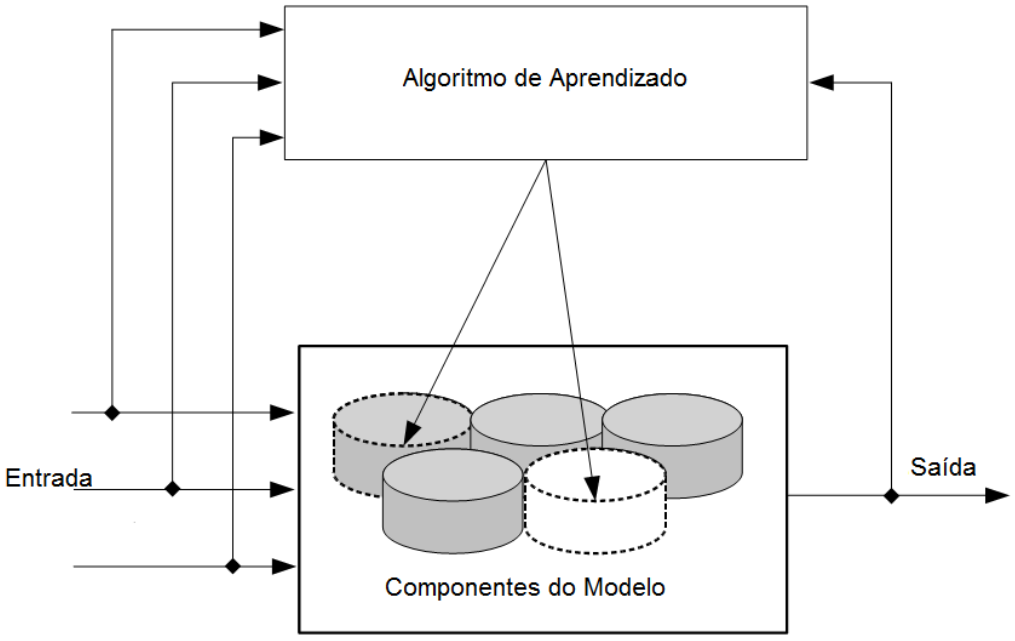

Fig. 2.1: Mecanismo de atualização da estrutura do modelo.

2.2 Revisão da Literatura

Os sistemas evolutivos e os métodos de aprendizagem *on-line* têm atraído crescente atenção dos pesquisadores ao longo dos últimos anos. Isso resultou na organização de diversos simpósios, congressos, seções especiais em congressos e periódicos, e livros, viz.:

- Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS).
- Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT).
- Conference on Adaptive and Intelligent Systems (ICAIS).
- Conference on Evolving and Adaptive Intelligent Systems (EAIS).
- Conference on Fuzzy Systems (FUZZ-IEEE).
- Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU).
- Conference on Machine Learning and Applications (ICMLA).
- Conference on Systems Man and Cybernetics (IEEE-SMC).
- Symposium on Evolving and Autonomous Learning System (EALS).
- Symposium on Evolving Fuzzy Systems (EFS).
- Symposium on Evolving Intelligent Systems (EIS).
- Workshop on Evolving and Self-Developing Intelligent Systems (ESDIS).
- Workshop on Evolving Systems at the IEEE World Congress on Computational Intelligence (WCCI).
- World Congress of the International Fuzzy Systems Association (IFSA).
- Periódico - Applied Soft Computing: Special issue on Evolving Soft Computing Techniques and Applications.
- Periódico - Evolving Systems.
- Periódico - Information Sciences: Special section on Online fuzzy machine learning and data mining.
- Livro - Evolving Connectionist Systems.
- Livro - Evolving Fuzzy Systems: Methodologies, Advanced Concepts and Applications.
- Livro - Evolving Intelligent Systems: Methodology and Applications.
- Livro - Learning in Non-Stationary Environments: Methods and Applications.

Os exemplos citados mostram que existe uma abundante literatura descrevendo modelos evolutivos. Esta seção apresenta o estado da arte.

2.2.1 Sistemas *Fuzzy* Evolutivos

Muitos modelos *fuzzy* evolutivos foram desenvolvidos considerando um conjunto de regras funcionais do tipo Takagi-Sugeno (TS) (Takagi and Sugeno, 1985). Estes trabalhos propõem modelos funcionais em que a estrutura do modelo (número de regras e parâmetros dos antecedentes) é flexível, sofrendo alterações baseadas em grupos criados, eliminados ou atualizados por algoritmos de agrupamento recursivo não-supervisionado. Os parâmetros dos consequentes são atualizados utilizando-se mínimos quadrados recursivos ou suas variações. As principais abordagens incluem aquelas reportadas em Angelov and Buswell (2001); Angelov et al. (2004b,a); Angelov and Filev (2004, 2005); Lughofer and Guardiola (2008); Lughofer (2008b); Wang and Vrbanek (2008); Angelov and Zhou (2008a,b); Angelov et al. (2008c); McDonald and Angelov (2010); Cernuda et al. (2011, 2012).

Uma abordagem para identificação de modelos *on-line* foi proposta em Angelov and Filev (2004). Esta se baseia em um algoritmo incremental de agrupamento subtrativo que atualiza recursivamente a estrutura da base de regras do modelo TS e seus parâmetros através de uma combinação de aprendizado supervisionado e não-supervisionado. A base de regras evolui continuamente adicionando novas regras com maior poder de sumarização e modificando as regras e parâmetros existentes. A criação de novas regras ou a modificação das regras existentes é realizada pela avaliação recursiva do potencial das novas amostras. Os parâmetros do consequente das regras são atualizados com o algoritmo recursivo de mínimos quadrados. Esse modelo evolutivo é chamado de eTS (*evolving Takagi-Sugeno* - Takagi-Sugeno Evolutivo). A natureza adaptativa do eTS em combinação com a forma transparente e extremamente compacta de suas regras torna este modelo promissor para a modelagem *on-line* de processos complexos. Diferentes versões do algoritmo e várias aplicações têm sido propostas.

O algoritmo eTS (original) não permite simplificação da base de regras (exclusão de regras) durante a aprendizagem, ignorando que as regras podem se tornar irrelevantes com a chegada de novas informações. Uma versão do eTS, Simpl_eTS, que simplifica a base de regras foi sugerida em Angelov and Filev (2005). O objetivo do Simpl_eTS é reduzir a complexidade dos cálculos do potencial. O modelo combina o conceito de dispersão com uma medida de potencial baseada na densidade dos dados, o que torna o algoritmo computacionalmente mais eficiente. O Simpl_eTS utiliza função de pertinência do tipo Cauchy no antecedente, ao invés de uma função Gaussiana e o algoritmo recursivo de mínimos quadrados para o aprendizado dinâmico dos parâmetros e estrutura do modelo eTS. A simplificação da base de regras do Simpl_eTS é baseada na representatividade das regras. Se o número de amostras associadas a uma regra for menor que um determinado percentual dos dados, a regra é ignorada e seu grau de disparo é definido em zero.

Uma versão estendida do eTS foi introduzida em Angelov and Zhou (2006) e denominada xTS (*eXtended Takagi-Sugeno*). Neste modelo o raio de influência de cada grupo (zona de influência) é estimado recursivamente. Outra funcionalidade introduzida neste modelo é a idade, que é utilizado para medir a relevância dos grupos. Em Maciel et al. (2012b) versões do eTS e xTS com múltiplas entradas e múltiplas saídas (MIMO - *Multiple Input / Multiple Output*) são utilizadas para a previsão em único modelo de todos os fatores dinâmicos latentes da curva de juros. Uma modificação do algoritmo eTS (Angelov and Filev, 2004) é proposta em Birek et al. (2014) e chamada de Mod_eTS. O Mod_eTS incorpora uma modificação no

mecanismo de atualização dinâmica dos grupos. Os grupos criados servem como base para regras *fuzzy* com funções de pertinência Gaussianas que são definidas pelos centros dos grupos e tem funções lineares no consequente. Os parâmetros das funções lineares são calculados usando um algoritmo recursivo de mínimos quadrados.

No trabalho de Angelov et al. (2010) foi proposto o eTS+ como uma nova abordagem para o eTS. No eTS+, os parâmetros do antecedente e a estrutura da base de regras são atualizados usando critérios como idade, utilidade, densidade local e zona de influência, o que compreende uma técnica robusta, flexível e autônoma. O modelo pode começar com um conjunto vazio de regras e não necessita de limites ou parâmetros especificados pelo usuário. Baseado na abordagem do eTS+ foi proposto o Simpl_eTS+ (Angelov, 2011).

O FLEXFIS (*Flexible Fuzzy Inference System* - Sistema de Inferência *Fuzzy* Flexível), detalhado em Lughofer (2008b), utiliza um algoritmo de agrupamento recursivo derivado de uma modificação da quantização vetorial (eVQ - (Lughofer, 2008a)). A estimação de parâmetros do consequente é realizada pelo algoritmo recursivo de mínimos quadrados ponderados. Entre as aplicações do FLEXFIS inclui-se o controle e previsão de emissões de poluentes (NOx) de motores de automóveis de combustão interna (Lughofer et al., 2011a). Uma nova versão do FLEXFIS foi introduzida em Lughofer (2012) e denominada FLEXFIS++ (*Flexible Evolving Fuzzy Inference System*). Os principais benefícios do FLEXFIS++ em relação ao FLEXFIS é a redução da complexidade pela união e exclusão de regras realizadas por um mecanismo que reduz a dimensionalidade de forma suave e gradual, e a maior interpretabilidade.

Um importante desenvolvimento é o SAFIS (*Sequencial Adaptive Fuzzy Inference System*) sugerido por Rong et al. (2006). O SAFIS emprega um método baseado na rede RBF (*Radial Basis Function*) para construção da base de regras de um sistema Takagi-Sugeno de ordem zero e pode ser representado por uma rede *feed-forward* com cinco camadas. Esse sistema de inferência *fuzzy* utiliza um critério baseado na distância Euclidiana entre as amostras e os centros dos grupos, em conjunto com uma medida de influência para adicionar regras de forma incremental. A remoção de regras é realizada utilizando o critério de influência. Um algoritmo baseado em uma modificação do Filtro de Kalman é utilizado para atualizar os parâmetros das regras. Uma versão estendida do SAFIS chamada de ESAFIS foi proposta em Rong et al. (2011, 2014) e provê melhorias na acurácia da previsão e na velocidade de processamento em relação a versão original. No ESAFIS, uma modificação no conceito da influência é introduzida para inclusão ou exclusão de regras. Quando não há alteração no número de regras, os parâmetros do consequente são atualizados por um algoritmo recursivo de mínimos quadrados. Isto reduz o tamanho da matriz de covariância e a complexidade computacional.

No trabalho de Angelov et al. (2007a) é proposto um classificador *fuzzy* evolutivo denominado eClass. Uma característica importante dessa abordagem é a possibilidade do eClass iniciar-se com um conjunto vazio de regras. No referido trabalho, propuseram-se quatro diferentes arquiteturas eClass que se diferenciam na sua estrutura interna e pela maneira como seus parâmetros e estruturas são atualizados. Outras propostas para o eClass podem ser encontradas em Angelov and Zhou (2008a,b). Um classificador evolutivo simplificado (Simpl_eClass) é introduzido em Baruah et al. (2011) como uma evolução do eClass (Angelov et al., 2007a). Assim como no eClass, no Simpl_eClass são propostos classificadores de ordem zero (Simpl_eClass0) e de primeira ordem (Simpl_eClass1). Os dois se diferem pela parte do consequente das regras e pela estratégia de classificação utilizada. No Simpl_eClass a aprendizagem não envolve cál-

culo de potencial, e sim o conceito de incremento da densidade, introduzido no Simpl_eTS+ (Angelov, 2011). Isto permite reduzir o custo computacional do classificador.

Em Lima (2008) e Lima et al. (2010) foi proposto o uso do Aprendizado Participativo (AP), um método robusto de aprendizado (Yager, 1990), para realizar a granularização do espaço de entrada de um sistema *fuzzy* evolutivo (ePL). Esse paradigma se baseia no mecanismo de aprendizagem humana, no qual a ideia principal é que o conhecimento já adquirido pelo sistema é utilizado durante o processo de aprendizado. No AP, uma conexão deve ser estabelecida entre as informações coletadas a cada instante e o conhecimento atual, as informações coerentes devem atualizá-lo. As informações pouco coerentes precisam ser detectadas e suavizadas a fim de rever o que se aprendeu até então ou proteger o sistema de possíveis erros e divergências (Yager, 2004; Lima et al., 2010). Nesse algoritmo, o conceito de aprendizagem participativa é utilizado na atualização dos antecedentes e o estimador de mínimos quadrados recursivos na atualização dos consequentes. O ePL foi generalizado para um sistema MIMO em Maciel et al. (2012c). Ao contrário da versão original do ePL em que o número inicial de grupos e respectivos centros são estimados usando o algoritmo de agrupamento *fuzzy* C-*means* (quando o conhecimento *a priori* sobre o problema está disponível), no ePL MIMO utilizou-se agrupamento subtrativo. Segundo os autores, o principal benefício no uso do agrupamento subtrativo é que não existe a necessidade do usuário estimar o número inicial de grupos, aumentando a autonomia do modelo. Uma versão melhorada do ePL foi proposta em Maciel et al. (2012a, 2013a) e é chamada de ePL+. As principais diferenças entre o ePL e o ePL+ são descritas a seguir. No ePL os raios dos grupos que representam a zona de influência das funções de pertinência são fixos e definidos por conhecimento especializado. As funções de pertinência das regras *fuzzy* têm a mesma dispersão, e, portanto, a mesma forma. Em contraste, o ePL+ adota um mecanismo adaptativo para calcular a zona de influência de cada regra. Assim, os raios dos grupos são diferentes e seus valores podem ser ajustados durante a aprendizagem. Outra diferença está no algoritmo de agrupamento. O ePL elimina grupos redundantes sempre que uma medida de compatibilidade entre os centros dos grupos é maior que um limiar definido pelo usuário, não utilizando nenhum mecanismo adaptativo para avaliar a qualidade da base de regras. O ePL+ inclui uma medida de utilidade para avaliar, de forma recursiva, a qualidade da estrutura dos grupos, de maneira similar ao eTS+ (Angelov et al., 2010).

Um sistema *fuzzy* evolutivo com uma nova abordagem do aprendizado participativo chamado de eMG (*Multivariable Gaussian Evolving Fuzzy System* - Sistema *Fuzzy* Evolutivo Multivariável Gaussiano) foi proposto em Lemos et al. (2011c). O eMG é um modelo funcional do tipo Takagi-Sugeno que usa um algoritmo de agrupamento evolutivo Gaussiano baseado no conceito de aprendizado participativo (Yager, 1990) para atualizar continuamente a base de regras. O agrupamento evolutivo pode ser visto como uma extensão da abordagem proposta em Silva et al. (2005). Diferentemente do algoritmo original, no eMG cada grupo é representado por uma função Gaussiana multivariável. A estrutura do agrupamento (número, centro e forma dos grupos) é atualizada recursivamente a cada passo do algoritmo e os limiares definidos automaticamente.

Maciel et al. (2013b) sugere uma abordagem não linear para problemas de previsão empregando um sistema *fuzzy* evolutivo simplificado baseado no conceito de nuvens de dados (Angelov and Yager, 2012) para definir o antecedente das regras *fuzzy*. Esse método utiliza uma medida de potencial da densidade dos dados e oferece uma alternativa não paramétrica para definição

do antecedente das regras. A medida de potencial reflete a real distribuição dos dados, sem a necessidade de operações de agregação ou funções de pertinência escalares. Segundo os autores, esta abordagem proporciona um algoritmo mais autônomo e eficiente.

Lemos et al. (2011b) propuseram uma nova abordagem para modelagem *fuzzy* evolutiva utilizando árvore de regressão linear recursiva (*evolving Fuzzy Linear Regression Trees* - eFT). A topologia da eFT pode ser atualizada continuamente por meio de testes estatísticos. A árvore de regressão linear *fuzzy* é gerada a partir de um fluxo de dados, empregando métodos estatísticos de seleção de modelos. Isto é feito de forma recursiva, substituindo as folhas por sub-árvores que melhoram a qualidade do modelo. Para avaliar a qualidade, o teste de seleção utilizado considera a precisão e o número de parâmetros do modelo resultante. Isto gera modelos altamente eficientes e evita o *overfitting*. O algoritmo de aprendizagem incremental começa com uma árvore contendo uma única folha associada a um modelo de regressão global inicial. O algoritmo de aprendizado avalia se a combinação ponderada de alguns modelos de regressão linear locais possui um desempenho superior ao do modelo linear global inicial. Essa abordagem captura mudanças nas informações de entrada e fornece modelos linguisticamente interpretáveis. O gradiente descendente é utilizado para ajuste dos nós internos da árvore. Em Lemos et al. (2012), a eFT é utilizada na previsão do volume de vendas de produtos de petróleo, e em Lemos et al. (2013) na previsão de vazão semanal.

Em Marsala (2013), é proposta uma nova abordagem com aprendizado incremental para construção de árvores de decisão *fuzzy* (*Fuzzy Decision Trees* - FDT). Na abordagem proposta utiliza-se uma medida de discriminação que considerada a idade das informações para classificar os atributos durante o processo de construção da FDT por meio de um fluxo de dados. Uma versão evolutiva dessa abordagem é proposta em Shaker et al. (2013) e denominada eFPT (*evolving Fuzzy Pattern Tree*). A idéia chave da eFPT é manter, além do modelo atual, um conjunto de árvores vizinhas que podem substituir o modelo atual se seu desempenho for inferior ao de uma das árvores alternativas. O desempenho dos modelos é monitorado continuamente por um mecanismo de janela deslizante de comprimento fixo.

Um método de aprendizagem para evolução de estruturas (*Structure Evolved Learning Method* - SELM) foi proposto em Wang et al. (2010). A estrutura do SELM evolui com o objetivo de reduzir o erro de modelagem. Isto ocorre com a inclusão de uma nova partição para a sub-região com o maior erro médio. Para a inclusão de uma nova partição, divide-se inicialmente a região de domínio do problema em duas sub-regiões. Em seguida, seleciona-se para divisão a sub-região com o maior erro médio. O processo incremental de divisão continua até que se obtenha uma precisão satisfatória. Essa abordagem de aprendizado foi empregada em modelos linguísticos (Mamdani) e funcionais (Takagi-Sugeno). Em Wang et al. (2013) é proposta uma simplificação do método para evolução de estruturas (*Simplified Structure Evolving Method* - SSEM) e empregado no modelo de Mamdani. O SSEM inicia-se com um modelo global com uma única regra *fuzzy* (o SELM inicia-se com 2^n regras, onde n é o número de variáveis de entrada). Esta simplificação reduz significativamente o número de parâmetros e de regras *fuzzy*, o que diminui o custo computacional do algoritmo e permite sua aplicação a problemas com qualquer dimensão do espaço de entrada.

Em Komijani et al. (2012) é introduzido o eTS-LS-SVM (*evolving Takagi-Sugeno Least Square Support Vector Machine*) para previsão de séries temporais. Como diferencial, este trabalho utilizou modelos locais não lineares do tipo máquinas de vetores suporte por mínimos

quadrados (LS-SVM - *Local Least Squares Support Vector Machine Models*) como consequente das regras *fuzzy*, em vez dos modelos lineares utilizados nos modelos evolutivos de Takagi-Sugeno convencionais. O aprendizado é realizado em duas etapas. Primeiro os antecedentes das regras são criados e atualizados adaptativamente por uma técnica de agrupamento sequencial para obter a estrutura do modelo Takagi-Sugeno. Em seguida, os parâmetros de cada modelo local LS-SVM (consequente da regra) são atualizados recursivamente por um novo algoritmo de aprendizado baseado em gradiente.

Um modelo *fuzzy* evolutivo (*evolving Fuzzy Model* - eFuMo) foi proposto em Dovzan et al. (2012). O modelo baseia-se em métodos recursivos de agrupamento derivados dos métodos *fuzzy* C-*means* (rFCM) e Gustafson-Kessel (RGK). A evolução da estrutura é obtida pela inclusão, união, divisão e exclusão de grupos. Os parâmetros dos modelos locais lineares são atualizados por um algoritmo recursivo de mínimos quadrados.

Uma abordagem para identificação de sistemas evolutivos Takagi-Sugeno aplicáveis a problemas de regressão foi proposta em Pouzols and Lendasse (2010), e chamada de eF-OP-ELM (*evolving Fuzzy Optimally Pruned Extreme Learning Machine*). O eF-OP-ELM gera aleatoriamente um conjunto de regras *fuzzy* com antecedentes simples, estrutura aleatória para os antecedentes e valores aleatórios para os parâmetros de funções de pertinência Gaussianas. O método LARS (*Least Angle Regression*) generalizado é empregado para classificar as funções *fuzzy* de acordo com sua acurácia. O número de regras *fuzzy* é selecionado pela estatística PRESS (*PREdiction Sum of Squares*). Os parâmetros dos consequentes são determinados analiticamente.

Uma outra abordagem utilizada em sistemas *fuzzy* evolutivos é baseada na computação granular, que emprega grânulos de informação tipo *fuzzy* para construir mapas granulares entre os dados granulares de entrada-saída. Um modelo *fuzzy* evolutivo que utiliza o conceito de grânulos de informação foi proposto em Leite and Gomide (2012) para aproximação de função e controle robusto. O modelo chamado de FBeM utiliza funções de pertinência trapezoidais e regras que combinam consequente linguístico (Tipo Mamdani) e funcional (Tipo TS). Em Leite et al. (2011) o algoritmo de aprendizagem do FBeM foi modificado para tratar problemas de previsão de séries temporais. O FBeM descrito em Leite et al. (2011) utiliza funções de pertinência Gaussianas e regras que combinam consequentes linguísticos e funcionais. A estrutura do FBeM evolui incrementalmente por meio da inclusão de um novo grânulo (regra) quando as regras existentes não são suficientemente ativadas por uma nova amostra. A adaptação das regras consiste em contrair ou expandir os grânulos representados pelos conjuntos *fuzzy* e é realizada quando a amostra de dados ativa suficientemente uma regra. Os grânulos podem ser unidos para formar um grânulo maior e remover a redundância.

2.2.2 Sistemas *Neuro-Fuzzy* Evolutivos

Sistemas *neuro-fuzzy* são redes neurais formadas por neurônios *fuzzy*. Essas redes combinam elementos das teorias de conjuntos *fuzzy* e de neurais, gerando modelos que integram o tratamento da incerteza e a interpretabilidade provida por sistemas *fuzzy* e a habilidade de aprendizado proporcionada por redes neurais (Pedrycz, 1991). Modelos *fuzzy* evolutivos híbridos são propostos em Kasabov and Song (1999); Kasabov (2001); Chang et al. (2009), e definidos como redes *neuro-fuzzy* evolutivas (Pedrycz and Gomide, 2007). Esses modelos

possuem estrutura multicamada sendo possível extrair regras *fuzzy* a partir de sua estrutura. Algoritmos de treinamento permitem adaptar os pesos da rede, assim como o número de nós em determinadas camadas dessas redes.

Duas abordagens para EFuNN (*Evolving Fuzzy Neural Network* - Rede *Neuro-Fuzzy* Evolutiva (Kasabov and Filev, 1998)) propostas em Kasabov and Song (1999) foram denominadas mEFuNN e dmEFuNN. Estas possuem uma estrutura que evolui por meio de aprendizado híbrido (supervisionado / não-supervisionado). A mEFuNN utiliza regras do tipo Mamdani (Mamdani and Assilian, 1999), enquanto a dmEFuNN utiliza regras do tipo Takagi-Sugeno (Takagi and Sugeno, 1985). O antecedente das regras é obtido por meio de agrupamento *fuzzy* C-*means* e os parâmetros das funções de saída são avaliados dinamicamente para cada um dos nós da regra.

Uma rede *neuro-fuzzy* evolutiva auto-organizável foi desenvolvida em Rubio (2009) e denominada SOFMLS (*On-Line Self-Organizing Fuzzy Modified Least-Squares Network*). Esta rede emprega um algoritmo de agrupamento evolutivo do vizinho mais próximo (*Evolving Nearest Neighborhood Clustering Algorithm*) para definir a estrutura (criação da base de regras) de um modelo TS de ordem zero. Os parâmetros do consequente da regras são atualizados por um algoritmo de mínimos quadrados.

Um novo sistema de inferência *fuzzy* denominado DENFIS (*Dynamic Evolving Neuro-Fuzzy Inference System* - Sistema de Inferência *Neuro-Fuzzy* Dinâmico Evolutivo) é proposto em Kasabov and Song (2002). Este sistema evolui incrementalmente por meio de um método de aprendizagem híbrido. Nesse modelo é empregado o ECM (*Evolving Clustering Method* - Método de Agrupamento Evolutivo (Song and Kasabov, 2001)) para adaptar a estrutura da base de regras e um algoritmo recursivo de mínimos quadrados ponderados para atualização dos parâmetros do consequente das regras. O DENFIS utiliza uma generalização local, como as EFuNNs (Kasabov and Filev, 1998) e redes as CMAC (Brown and Harris, 1994). Por isso necessita de mais dados de treinamento que os usados pelos modelos de generalização global, como ANFIS (Jang, 1993) e MLP (Haykin, 1999).

Uma rede evolutiva *neuro-fuzzy* com memória semântica (*evolving Neuro-Fuzzy Semantic Memory* - eFSM) é introduzida em Tung and Quek (2010). A rede é do tipo Mamdani e adapta sua estrutura de forma incremental pelo fluxo de dados. O algoritmo de aprendizado possibilita a inclusão de novas regras a partir do surgimento de novas informações nos dados de entrada. As regras obsoletas, que já não descrevem a tendência dos dados, são eliminadas. O conceito de potencial é utilizado para atualizar a base de regras.

Uma rede *neuro-fuzzy* recorrente chamada de IRSFNN (*Interactively Recurrent Self-Evolving Fuzzy Neural Network*), é proposta em Lin et al. (2013a). A estrutura recorrente da IRSFNN é formada por laços de repetição externos e por uma realimentação interna. Estes alimentam a força de disparo das regras. A rede emprega uma FLNN (*Functional Link Neural Network*) no consequente das regras promovendo a capacidade de mapeamento. Na FLNN, os consequentes das regras são definidos como funções não lineares. O algoritmo de aprendizagem inicia com uma base de regras vazia e as regras são geradas simultaneamente com a evolução da estrutura e ajuste dos parâmetros. As regras são criadas por um algoritmo de agrupamento e os parâmetros do consequente são atualizados por um algoritmo baseado em uma modificação do Filtro de Kalman. Os antecedentes das regras e os parâmetros da recorrência da rede são ajustados pelo algoritmo do gradiente descendente.

Um sistema de inferência genérico *neuro-fuzzy* evolutivo (*Generic Evolving Neuro-Fuzzy Inference System* - GENEFIS) é introduzido em Pratama et al. (2013b) e empregado em problemas de classificação em Pratama et al. (2013a). O GENEFIS adota uma forma generalizada de Takagi-Sugeno onde o antecedente é composto por função Gaussiana multivariada e o seu processo de aprendizado se inicia a partir de uma base de regras vazia e a evolução da estrutura se dá com base em novas informações provenientes de um fluxo de dados. O método DS (*Datum Significance*) e GART+ (*Generalized Adaptive Resonance Theory+*) são usados para inclusão de novas regras. Regras obsoletas são excluídas pela técnica ERS (*Extended Rule Significance*) e uma técnica baseada em Kernel é utilizada para mesclar regras redundantes.

Em Lin et al. (2013b) e Lin et al. (2014) foram propostas duas redes *neuro-fuzzy* evolutivas similares que utilizam modelos *fuzzy* do Tipo-2. As duas redes iniciam-se com uma base de regras vazias e o aprendizado se dá pela evolução da estrutura e ajuste dos parâmetros. O antecedente das regras *fuzzy* são definidos usando conjuntos *fuzzy* do Tipo-2 e o consequente é do tipo Takagi-Sugeno. A estrutura das redes é obtida por um agrupamento *on-line fuzzy* do Tipo-2, os parâmetros do consequente são ajustados via Filtro de Kalman e os conjuntos *fuzzy* do Tipo-2 do antecedente são atualizados pelo algoritmo do gradiente descendente. A diferença entre as duas redes é que a MRIT2NFS (*Mutually Recurrent Interval Type-2 Neural Fuzzy System*) (Lin et al., 2013b) utiliza uma estrutura recorrente com realimentação mútua, enquanto a rede TSCIT2FNN (*TSK-Type-Based Self-Evolving Compensatory Interval Type-2 Fuzzy Neural Network)* (Lin et al., 2014) utiliza intervalos compensatórios. Os pesos da realimentação mútua e os pesos compensatórios são ajustados pelo algoritmo do gradiente descendente.

Uma nova abordagem para sistemas evolutivos que utiliza um sistema *neuro-fuzzy* de Mamdani Tipo-2 é proposta em Tung et al. (2013). O sistema proposto é chamado eT2FIS (*evolving Type-2 Neural Fuzzy Inference System*) e a aprendizagem é realizada de forma incremental, onde o sistema evolui e aprende com a chegada de novas informações. O algoritmo de aprendizagem usa uma medida de similaridade para evoluir sua estrutura por meio da inclusão de novas regras, exclusão de regras obsoletas, e/ou pela união de regras redundantes. A abordagem é aplicada na identificação *on-line* de um sistema variante no tempo, no monitoramento *on-line* dos preços de ações financeiras e na modelagem da densidade do fluxo de tráfego de uma estrada.

2.2.3 Sistemas Evolutivos com Seleção de Entradas

A característica importante dos sistemas *fuzzy* evolutivos é a sua capacidade de adaptar, simultaneamente, a estrutura e os parâmetros do modelo a partir dos dados de entrada. Uma limitação das atuais abordagens evolutivas diz respeito à utilização de mecanismos de seleção adaptativa de entradas em conjunto com a adaptação da estrutura e dos parâmetros dos modelos usando os dados de entrada. Em geral, as variáveis de entrada são escolhidas utilizando-se, por exemplo, conhecimento *a priori* e, depois, são mantidas fixas (Lemos et al., 2011c). Procedimentos de seleção adaptativa de entradas para capturar variações nos modos de operação e variações na estrutura dos sistemas continuam a ser um desafio na modelagem *fuzzy* (Zhu et al., 2010; Lughofer, 2011). Uma questão chave é como incorporar a seleção de variáveis no processo de aprendizagem incremental de uma maneira que não cause descontinuidade ou danos

no processo de aprendizagem. Em outras palavras, o mecanismo de seleção de variáveis deve ser parte do próprio processo de aprendizagem.

A seleção adaptativa de entradas pode ser realizada como se segue (Guyon and Elisseeff, 2003):

- ***Forward Selection* (Seleção para frente):** o modelo inicia-se considerando uma única variável e são acrescentadas novas variáveis ao modelo durante a aprendizagem;

- ***Backward Selection* (Eliminação para trás):** o modelo inicia-se com todas as variáveis e descarta-se as variáveis irrelevantes durante a aprendizagem.

Para gerar subconjuntos de variáveis aninhadas, a seleção para frente é computacionalmente mais eficiente do que a eliminação para trás (Guyon and Elisseeff, 2003). Os métodos de seleção incremental de entradas iniciam-se com uma ou mais variáveis, e, de acordo com o fluxo de dados, novas variáveis podem ser adicionadas ou eliminadas do modelo. O conjunto inicial de variáveis pode ser escolhido a partir de conhecimento prévio sobre o sistema a ser modelado ou por meio de um método de ranqueamento (Guyon and Elisseeff, 2003).

Como mencionado anteriormente, a incorporação de métodos de seleção adaptativa de entradas aos modelos evolutivos é um recente desafio para o desenvolvimento destes modelos. Na realidade, métodos de seleção de variáveis de entrada já foram incorporados a modelos evolutivos para resolver problemas de classificação (Katakis et al., 2005, 2006; Lughofer, 2011) e identificação de sistemas (Lemos et al., 2011a), por exemplo.

Mais especificamente, o classificador com método de seleção de variáveis de entrada proposto em Katakis et al. (2006) cria uma lista de variáveis candidatas e atribui pesos para indicar a relevância de cada uma das variáveis. As variáveis de entrada mais relevantes são selecionadas para compor o modelo. A relevância das variáveis de entrada é atualizada constantemente, possibilitando, assim, enfatizar a importância de cada uma delas, mas o número de variáveis de entrada do modelo permanece fixo.

Recentemente, foi proposto, em Lughofer (2011), um classificador *fuzzy* evolutivo com seleção de variáveis de entrada. A metodologia de seleção de variáveis foi implementada no algoritmo de aprendizado do classificador *fuzzy* evolutivo FLEXFIS-Class (Lughofer et al., 2007). Nesta metodologia, pesos com valores no intervalo $[0, 1]$ são atribuídos a cada variável de entrada proporcionalmente à sua relevância. Variáveis importantes para discriminação entre duas ou mais classes têm seus valores próximos de 1, enquanto que variáveis menos relevantes têm o valor de seus pesos próximos de 0. Quanto maior o peso da variável, maior o seu impacto na aprendizagem. Os pesos são atualizados continuamente durante o processo de aprendizagem.

Uma árvore de regressão linear *fuzzy* evolutiva com seleção adaptativa de entradas foi introduzida em Lemos et al. (2011a), sendo esta uma evolução da árvore apresentada em Lemos et al. (2011b). A topologia da árvore é atualizada incrementalmente utilizando testes estatísticos, possibilitando a alteração do número de nós da árvore e do número de variáveis de entrada em cada folha a cada nova entrada de dados. A árvore de regressão possibilita a redução do número de parâmetros livres, resultando em modelos mais compactos.

2.3 Resumo

Este capítulo apresentou uma revisão das abordagens no desenvolvimento de sistemas evolutivos, incluindo sistemas *fuzzy* evolutivos, redes *neuro-fuzzy* evolutivas e sistemas evolutivos com seleção adaptativa de entradas. Diferentes abordagens e metodologias empregadas nos sistemas evolutivos foram discutidas. Estas servem como base para o desenvolvimento dos algoritmos evolutivos propostos neste trabalho.

Verifica-se que a maioria das abordagens utiliza algoritmos recursivos de agrupamento para construção e atualização da estrutura da base de regras. Entre os principais algoritmos de agrupamento utilizados estão: Subtrativo, Participativo, *Fuzzy C-means*, eVQ, ECM, Gustafson-Kessel, *Fuzzy* Tipo-2, e variações. Estes algoritmos de agrupamento empregam um ou vários critérios para evoluir a base de regras. Os principais critérios adotados são medidas de densidade, dispersão, distância, idade, zona de influência, potencial e utilidade. Algumas metodologias utilizam informações sobre o erro de modelagem para construir incrementalmente a base de regras. A atualização dos parâmetros do antecedente e/ou consequente das regras, geralmente, é realizada recursivamente por algoritmos de mínimos quadrados, gradiente descendente, Filtro de Kalman, ou por versões modificadas desses algoritmos.

As principais tendências e perspectivas nas abordagens evolutivas estão no desenvolvimento de algoritmos com maior grau de adaptabilidade que permitam a modificação/inclusão/exclusão de regras, a atualização dos parâmetros e o cálculo das saídas com menor custo computacional, maior autonomia e menor número de parâmetros definidos pelos usuários. Outra questão chave é o desenvolvimento de métodos incrementais de seleção adaptativa de entradas incorporados ao algoritmo de aprendizagem evolutivo sem causar descontinuidade no processo de aprendizagem. Essas tendências e perspectivas são corroboradas pelas recentes abordagens que propõem melhorias em importantes algoritmos evolutivos, como, por exemplo, nos algoritmos do eTS, ePL, FLEXFIS e SAFIS, para citar somente alguns.

Apesar das relevantes abordagens desenvolvidas e dos importantes avanços alcançados no desenvolvimento de sistemas evolutivos nos últimos anos, vários aspectos ainda precisam ser melhores explorados, especialmente em aplicações reais. Hoje existe uma crescente demanda para implementação dos sistemas evolutivos, aplicados a diversos domínios, tais como sensores inteligentes, sistemas autônomos, controle adaptativo não linear, detecção e diagnóstico de falhas, previsão, extração de conhecimento, biomedicina, bioinformática. Por estas razões, é provável que esta área de pesquisa e desenvolvimento se mantenha ativa e em crescimento nos próximos anos.

Capítulo 3

Revisitando a Rede *Neo-Fuzzy-Neuron* (NFN)

Este capítulo revisa a estrutura *neuro-fuzzy Neo-Fuzzy-Neuron* (NFN) (Yamakawa et al., 1992) que servirá como base para os algoritmos evolutivos propostos nos próximos capítulos. A próxima seção apresenta os fatores que motivaram o uso da estrutura NFN no desenvolvimento dos algoritmos propostos. Na Seção 3.2, é descrita a estrutura da rede *Neo-Fuzzy-Neuron* (NFN). Depois, na Seção 3.3, são apresentadas as funções de pertinência empregadas na rede *neuro-fuzzy*. Um método de aprendizado incremental com taxa de aprendizado ótima é descrito na Seção 3.4. A estrutura da NFN com múltiplas saídas é o assunto da Seção 3.5. Por fim, a Seção 3.6 discorre sobre as considerações finais.

3.1 Introdução

A estrutura, batizada de *Neo-Fuzzy-Neuron*, foi proposta por Yamakawa et al. (1992). A escolha desta topologia de rede *neuro-fuzzy* se dá por ela favorecer o uso de algoritmos de aprendizado eficientes para atualização dos pesos da rede e possibilitar a evolução de sua estrutura por meio da inclusão/exclusão de funções de pertinência e variáveis de entrada durante o aprendizado incremental. Outro fator relevante para sua escolha é o seu baixo custo computacional, fator extremamente importante para implementação em sistemas de tempo real com dinâmica rápida, aliado a seu bom desempenho. A complexidade computacional dessa topologia foi avaliada em Caminhas and Gomide (2000). Na análise realizada, foram consideradas as expressões que permitem calcular as saídas em função das entradas, isto é, o número de operações básicas e os cálculos de funções. Os resultados apresentados mostram a simplicidade das operações executadas pela rede NFN e, consequentemente, o seu baixo custo computacional.

Na rede NFN cada função de pertinência representa uma regra *fuzzy* tipo Takagi-Sugeno (Takagi and Sugeno, 1985). Os parâmetros do consequente das regras (pesos da rede) são atualizados por um método baseado no gradiente com taxa de aprendizagem ótima.

3.2 Estrutura da Rede *Neo-Fuzzy-Neuron*

Esta seção apresenta a rede *neuro-fuzzy* construída com *Neo-Fuzzy-Neurons* (NFN). Basicamente, a estrutura da rede NFN (Figura 3.1) corresponde a n estruturas tipo Takagi-Sugeno (Takagi and Sugeno, 1985) de ordem zero, uma para cada entrada. Os modelos são desacoplados, e a saída é a soma dos modelos individuais. A saída da NFN no instante t (y_t) é dada por:

$$y_t = f_1(x_{t1}) + ... + f_n(x_{tn}) = \sum_{i=1}^{n} f_i(x_{ti}) = \sum_{i=1}^{n} y_{ti}. \tag{3.1}$$

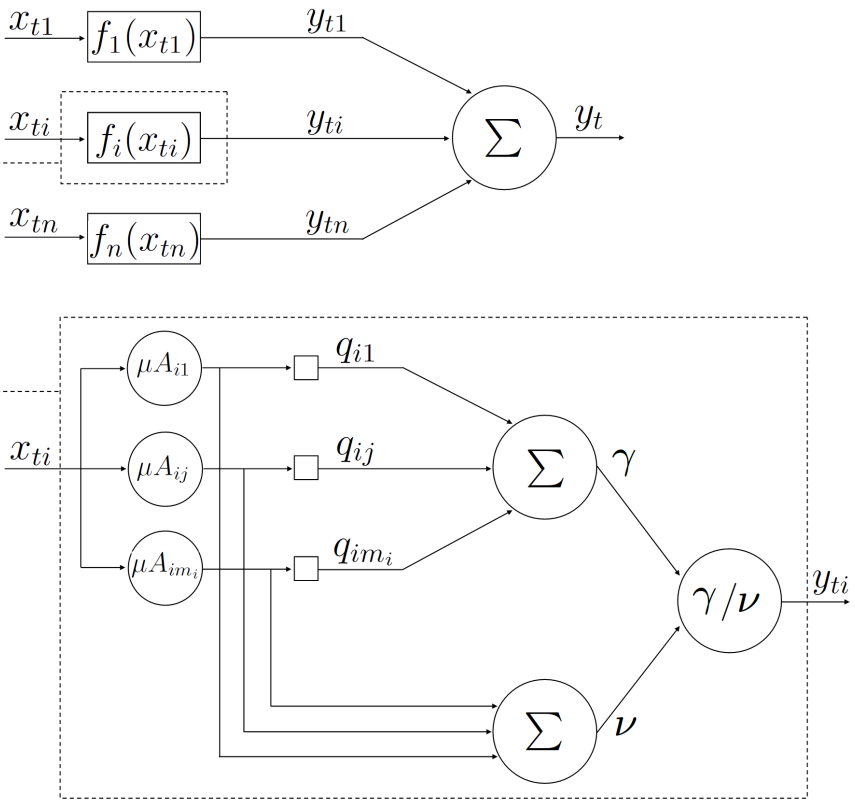

Fig. 3.1: Estrutura da NFN.

Cada saída y_{ti} é definida por um conjunto de m_i regras *fuzzy* de Takagi-Sugeno. Aqui, assume-se que o domínio de cada variável de entrada x_{ti} é particionado em m_i funções de pertinência triangulares e complementares, como ilustrado na Figura 3.2. Com essa partição do domínio de entrada, têm-se as seguintes m_i regras para cada entrada x_{ti}:

$$\begin{array}{llll} R_i^1 & Se & x_{ti}\,\acute{e}\,A_{i1} & Então \quad y_{ti}\,\acute{e}\,q_{i1}. \\ R_i^j & Se & x_{ti}\,\acute{e}\,A_{ij} & Então \quad y_{ti}\,\acute{e}\,q_{ij}. \\ R_i^{m_i} & Se & x_{ti}\,\acute{e}\,A_{im_i} & Então \quad y_{ti}\,\acute{e}\,q_{im_i}. \end{array}$$

Os valores de $y_{ti} = f_i(x_{ti})$ são calculados usando a seguinte expressão:

$$y_{ti} = f_i(x_{ti}) = \frac{\sum_{j=1}^{m_i} \mu_{A_{ij}}(x_{ti}) q_{ij}}{\sum_{j=1}^{m_i} \mu_{A_{ij}}(x_{ti})} = \frac{\gamma}{\nu}. \tag{3.2}$$

Como as funções de pertinência são complementares, no máximo duas entre as m_i regras são ativadas por uma entrada x_{ti}. Essas duas regras são indexadas por k_i e $k_i + 1$. Uma regra está ativa quando o seu valor de pertinência é maior que zero. A Figura 3.2 mostra um exemplo. Neste exemplo, somente as regras R_i^2 e R_i^3 estão ativas, pois o grau de pertinência é diferente de zero para os conjuntos *fuzzy* A_{i2} e A_{i3}. Alternativamente, pode-se dizer que as funções de pertinência dos conjuntos A_{i2} e A_{i3} estão ativas. O valor de k_i corresponde ao índice da primeira função de pertinência ativa, isto é, aquela com valor não nulo para uma dada entrada. Esse valor é obtido pela menor diferença positiva entre a variável de entrada x_{ti} e o valor modal das funções de pertinência. Como as funções de pertinência são complementares temos que, por definição:

$$\nu = \sum_{j=1}^{m_i} \mu_{A_{ij}}(x_{ti}) = \mu_{A_{ik_i}}(x_{ti}) + \mu_{A_{ik_i+1}}(x_{ti}) = 1. \tag{3.3}$$

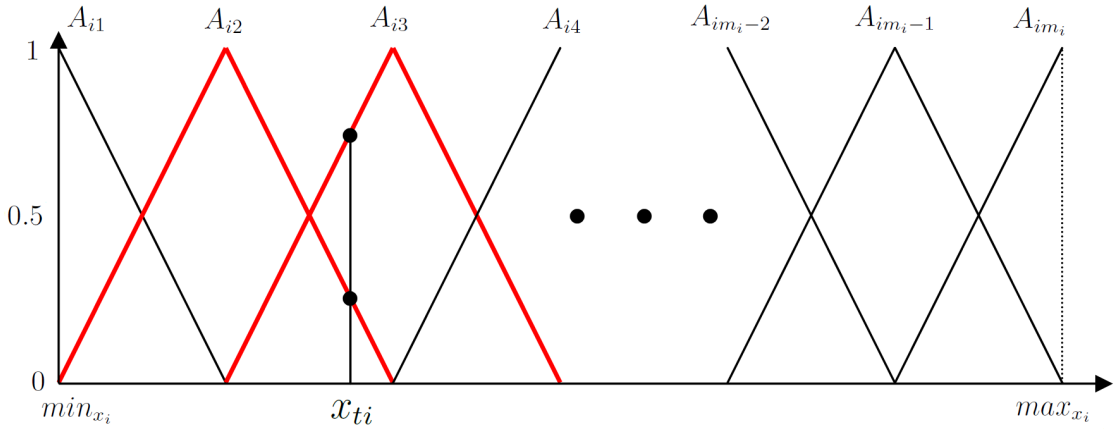

Fig. 3.2: Funções de pertinência triangulares e complementares.

Portanto, a expressão final para y_{ti} é dada por (3.4), e a estrutura da rede *Neo-Fuzzy-Neuron* mostrada inicialmente na Figura 3.1 pode ser ilustrada de forma mais simples pela Figura 3.3.

$$y_{ti} = \gamma = \mu_{A_{ik_i}}(x_{ti}) q_{ik_i} + \mu_{A_{ik_i+1}}(x_{ti}) q_{ik_i+1}. \tag{3.4}$$

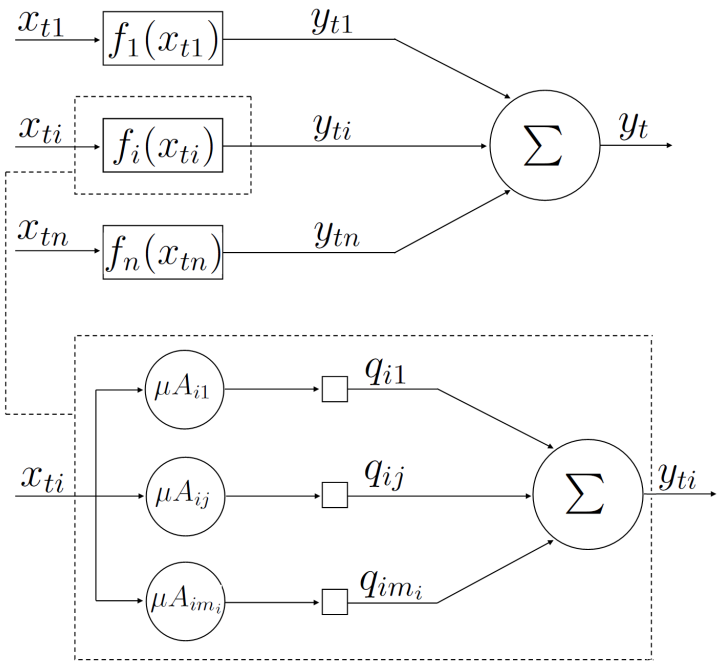

Fig. 3.3: Estrutura simplificada da NFN.

3.3 Funções de Pertinência

As funções de pertinência utilizadas na NFN são triangulares e complementares. Outros tipos de funções de pertinência também podem ser utilizados, como funções Gaussianas ou Trapezoidais (Bodyanskiy et al., 2003; Zaychenko and Gasanov, 2012). A função triangular é a mais utilizada na NFN. Optou-se pela sua utilização neste trabalho devido à simplicidade e baixo custo computacional envolvidos nos seus cálculos. Uma função de pertinência triangular pode ser representada por três parâmetros: o limite inferior (a), o valor modal (b) e o limite superior (c). Como as funções de pertinência são complementares, a j-ésima função é definida pelo seu valor modal (b_j), o limite inferior é dado pelo valor modal da função adjacente inferior ($a_j = b_{j-1}$) e o limite superior pelo valor modal da função adjacente superior ($c_j = b_{j+1}$), como ilustra a Figura 3.4.

Considerando uma entrada x_{ti}, o valor modal das funções de pertinência é computado a partir de:

$$b_{ij} = min_{x_i} + (j-1)\Delta_i, \tag{3.5}$$

onde i indexa a variável de entrada, j indexa a função de pertinência, min_{x_i} é o limite inferior do domínio da i-ésima variável de entrada, e Δ_i é calculado usando:

$$\Delta_i = \frac{(max_{x_i} - min_{x_i})}{m_i - 1}, \tag{3.6}$$

onde max_{x_i} é o limite superior para o domínio da i-ésima variável de entrada e m_i é o número

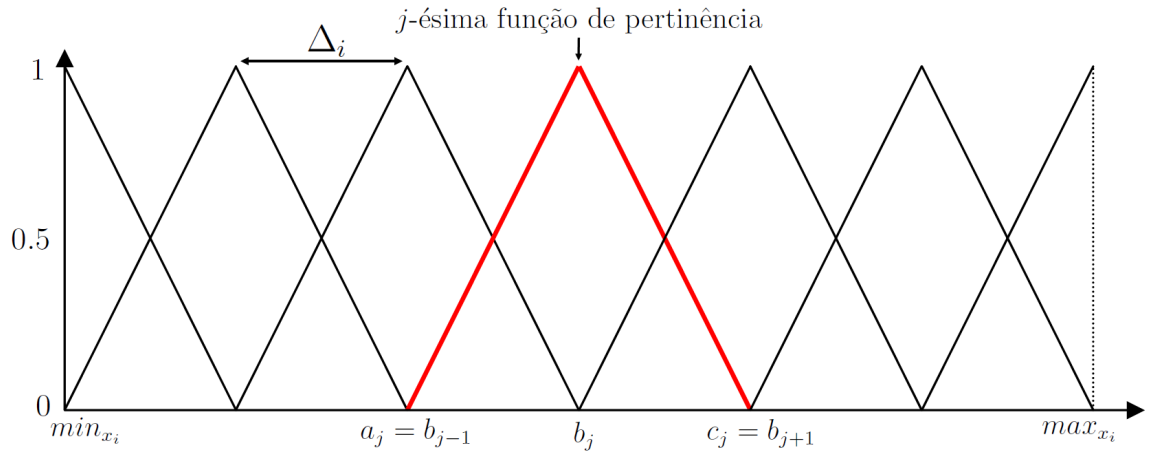

Fig. 3.4: Parâmetros das funções de pertinência.

de funções de pertinência que granulariza o domínio da i-ésima variável.

3.4 Aprendizado da Rede NFN

Esta seção detalha o procedimento de adaptação dos pesos da rede NFN usando um método baseado no gradiente com taxa de aprendizado ótima. O aprendizado é supervisionado e tem o propósito de ajustar os parâmetros q_{ij}. O procedimento de adaptação dos pesos descrito a seguir é um mecanismo incremental de atualização, portanto, apropriado para modelos evolutivos. Procedimentos alternativos como atualização em lote e incremental com termo de *Momentum* são tratados em Yamakawa et al. (1992).

Dado um conjunto de entrada $X = [x_1, x_2, ..., x_t, ..., x_N]$, sendo N o número de amostras e $x_t = (x_{t1}, x_{t2}, ..., x_{ti}, ..., x_{tn})$ o t-ésimo padrão de entrada. Se y_t é a saída estimada pela NFN e $\hat{y}_t$ a saída desejada em t, n é o número de variáveis de entrada. Calcula-se o erro quadrático, E_t a partir de:

$$E_t = \frac{1}{2}(y_t - \hat{y}_t)^2 = E_t(q_{ij}). \tag{3.7}$$

Fixadas a granularização dos domínios de entrada e as funções de pertinência, o aprendizado da NFN é realizado escolhendo-se os valores de pesos que minimizam o erro E_t. A minimização do erro pode ser formulada como:

$$min\, E_t = E_t(q_{ij}), \tag{3.8}$$

com $q_{ij} \in \Re$, $\forall\, i = 1, ..., n$; $j = 1, ..., m_i$. Uma vez que para cada entrada somente as funções de pertinência A_{ik_i} e A_{ik_i+1} possuem graus de ativação diferentes de zero, a minimização do erro pode reescrita por:

$$min\, E_t = E_t(q_{ij}) = E_t(\tilde{q}), \tag{3.9}$$

com $q_{ij} \in \Re,\ \forall\, i = 1,\ ...,\ n;\ j = k_i,\ k_i + 1$, e $\tilde{q}$ é calculado por:

$$\tilde{q} = [q_{1k_1}\ q_{1k_1+1}\ ...\ q_{ik_i}\ q_{ik_i+1}\ ...\ q_{nk_n}\ q_{nk_n+1}]. \tag{3.10}$$

Independente do número de funções de pertinência, a minimização de $E_t(\tilde{q})$ envolve $2n$ variáveis e $E_t(\tilde{q})$ é uma função quadrática e convexa com relação aos parâmetros do modelo de saída (pesos) (Caminhas, 1997; Caminhas et al., 1998; Caminhas and Gomide, 2000). Portanto, o mínimo local de $E_t(\tilde{q})$ é mínimo global (Uchino and Yamakawa, 1994).

Como $E_t(\tilde{q})$ é uma função não linear, a convergência para a solução ótima depende do ponto inicial, da direção de busca e do tamanho do passo na direção de busca. A condição inicial para a aprendizagem da NFN pode ser nula, sem comprometer a convergência do algoritmo (Caminhas, 1997). Como direção de busca adota-se a do gradiente, dada a simplicidade no seu cálculo. Portanto, q_{ik_i} e q_{ik_i+1} são atualizados utilizando:

$$\begin{aligned} q_{ik_i} &= q_{ik_i} - \alpha(y_t - \hat{y}_t)\mu_{A_{ik_i}}(x_{ti}), \\ q_{ik_i+1} &= q_{ik_i+1} - \alpha(y_t - \hat{y}_t)\mu_{A_{ik_i+1}}(x_{ti}). \end{aligned} \tag{3.11}$$

A taxa de aprendizado α corresponde ao tamanho do passo no algoritmo do gradiente, e, em geral, é determinada por meio de um método de busca unidirecional (Bazaraa et al., 1993). Neste trabalho, utiliza-se uma fórmula fechada (Caminhas et al., 1998; Caminhas and Gomide, 2000) para calcular α, pois a função a ser minimizada ($E_t(\tilde{q})$) é quadrática e, dada a forma de $E_t(\tilde{q})$ pode-se calcular α analiticamente para se obter erro de aproximação nulo a cada passo do algoritmo. A fórmula para se calcular o passo ótimo α é:

$$\alpha = \frac{1}{\sum_{i=1}^{n} \mu_{A_{ik_i}}(x_{ti})^2 + \mu_{A_{ik_i+1}}(x_{ti})^2}. \tag{3.12}$$

Apenas as funções de pertinência ativas são relevantes durante a adaptação dos parâmetros da rede NFN. Isto é, somente os pesos das funções ativas são atualizados a cada passo. Isso significa que a NFN tem um tempo de processamento e adaptação muito menor que aqueles demandados por redes neurais clássicas. Isto é especialmente importante na modelagem de processos complexos e de grande porte, uma vez que o número de operações é pequeno e envolve a computação de funções simples (Bacelar et al., 2003, 2004).

O Algoritmo 1 sumariza a estimação da saída e o processo de atualização dos parâmetros da NFN com uma única saída.

3.5 NFN com Múltiplas Saídas

Na estrutura da *Neo-Fuzzy-Neuron*, introduzida por Yamakawa et al. (1992) e descrita nas seções anteriores, o número de saídas é igual a um. Em Caminhas and Gomide (2000) foi proposta uma modificação na estrutura da rede NFN que permite o cálculo de múltiplas saídas. A estrutura da rede NFN com múltiplas saídas é mostrada na Figura 3.5.

Similarmente a NFN com uma única saída, na rede NFN com múltiplas saídas cada uma das saídas y_t^l é calculada pela soma das saídas individuais y_{ti}^l e podem ser obtidas por:

Algorithm 1: Rede NFN com uma única saída.

Entrada: x_t, $\hat{y}_t$, n;
Saída: y_t;
Inicializar b_{ij};
for $t = 1, 2, ...$ **do**
  Ler x_t, $\hat{y}_t$;
  for $i = 1 : n$ **do**
    Calcular $\mu_{A_{ik_i}}(x_{ti})$, $\mu_{A_{ik_i+1}}(x_{ti})$;
    Calcular y_{ti};
  end for
  Calcular y_t;
  Calcular α;
  Atualizar q_{ik_i}, q_{ik_i+1};
end for

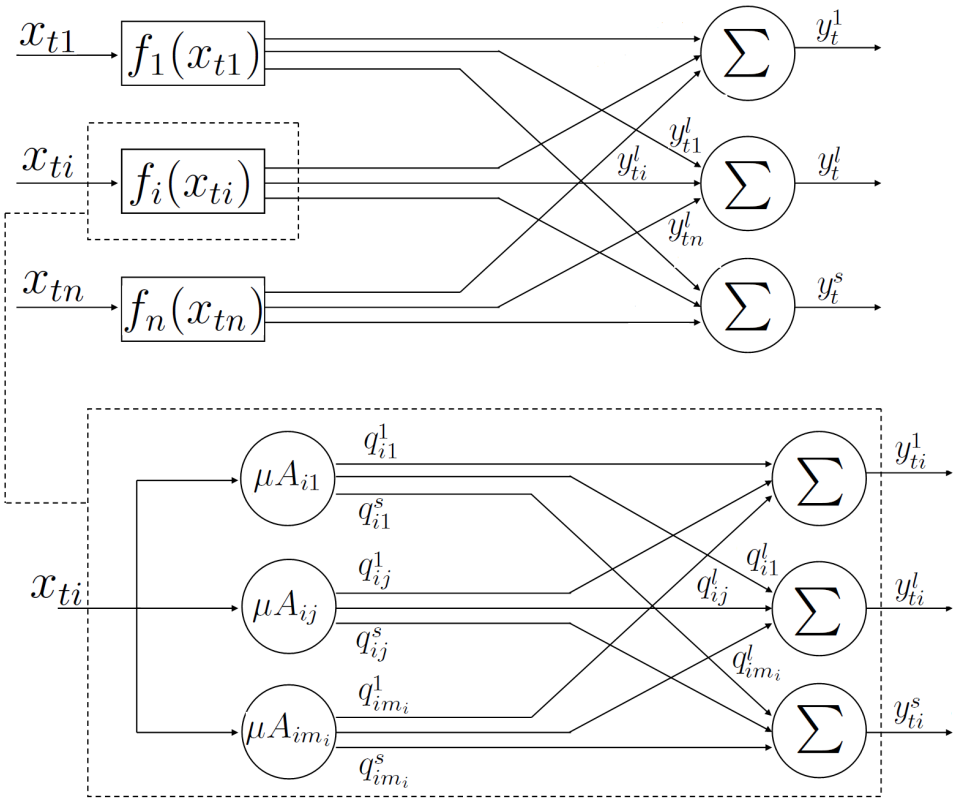

Fig. 3.5: Estrutura da NFN com múltiplas saídas.

$$y_t^l = \sum_{i=1}^{n} y_{ti}^l = \sum_{i=1}^{n} (\mu_{A_{ik_i}}(x_{ti}) q_{ik_i}^l + \mu_{A_{ik_i+1}}(x_{ti}) q_{ik_i+1}^l), \tag{3.13}$$

onde l é o índice de cada saída da NFN, t o passo atual, e n número de variáveis de entrada. Os pesos $q_{ik_i}^l$ e $q_{ik_i+1}^l$ da rede NFN com múltiplas saídas são ajustados de forma análoga ao descrito na Seção 3.4. Note que, existe um conjunto de pesos para cada saída da NFN.

Em um sistema com múltiplas saídas, pode-se utilizar uma NFN com múltiplas saídas ou várias redes NFN com uma única saída. A vantagem do uso da NFN com múltiplas saídas está no menor custo computacional, pois o grau de ativação das funções de pertinência é calculado uma única vez para cada entrada.

O Algoritmo 2 detalha o processo de estimação da saída e atualização dos parâmetros da rede NFN com múltiplas saídas.

Algorithm 2: Rede NFN com múltiplas saídas.

Entrada: x_t, $\hat{y}_t$, n, s;
Saída: y_t^l;
Inicializar b_{ij};
for $t = 1, 2, ...$ **do**
  Ler x_t, $\hat{y}_t$;
  for $i = 1 : n$ **do**
    Calcular $\mu_{A_{ik_i}}(x_{ti})$, $\mu_{A_{ik_i+1}}(x_{ti})$;
  end for
  for $l = 1 : s$ **do**
    for $i = 1 : n$ **do**
      Calcular y_{ti}^l;
    end for
    Calcular y_t^l;
    Calcular α;
    Atualizar $q_{ik_i}^l$, $q_{ik_i+1}^l$;
  end for
end for

3.6 Resumo

Este capítulo apresentou a rede base para os algoritmos de aprendizado evolutivo propostos nos próximos capítulos. A rede apresentada possui uma estrutura compacta composta por equações simples e com uma baixa complexidade computacional, mesmo para um grande número de variáveis de entrada. Como um dos objetivos do trabalho é desenvolver algoritmos evolutivos eficientes e com baixo custo computacional, a rede apresentada neste capítulo mostra-se com uma abordagem promissora para alcançar esse objetivo.

Capítulo 4

Rede *Neo-Fuzzy-Neuron* Evolutiva (eNFN)

Neste capítulo, propõe-se um algoritmo de aprendizado evolutivo para a rede *Neo-Fuzzy-Neuron* (NFN) apresentada no Capítulo 3. A *Neo-Fuzzy-Neuron* possui uma estrutura fixa. A estrutura do algoritmo de aprendizado proposto neste capítulo é flexível, isto é, evolui de acordo com um fluxo de dados. A evolução da estrutura se dá pela alteração do valor modal, pela inclusão e também pela exclusão de funções de pertinência. Espera-se que esta rede evolutiva tenha um baixo custo computacional, uma boa precisão e consiga lidar com as mudanças ocorridas no fluxo de dados de entrada.

Uma breve descrição do algoritmo proposto é apresentada na Seção 4.1. A Seção 4.2 descreve o processo de inicialização das funções de pertinência, enquanto o método para adaptação de contexto do domínio das variáveis de entrada é apresentado na Seção 4.3. A abordagem empregada na atualização do valor modal das funções de pertinência é descrita na Seção 4.4. A metodologia para criação e inserção de funções de pertinência é apresentada na Seção 4.5, e a para exclusão na Seção 4.6. A rede eNFN com múltiplas saídas é o assunto da Seção 4.7, enquanto a Seção 4.8 detalha e exemplifica os parâmetros das redes eNFN. A Seção 4.9 apresenta a complexidade computacional das redes eNFN e, por fim, a Seção 4.10 contém as considerações.

4.1 Introdução

A rede *Neo-Fuzzy-Neuron* evolutiva (eNFN) atualiza seus pesos usando o método do gradiente com taxa de aprendizado ótima, similar ao descrito na Seção 3.4. No entanto, diferente da abordagem original, a abordagem evolutiva atualiza a estrutura da rede usando os dados de entrada. Ao contrário da rede original, na eNFN os valores modais das funções de pertinência, o número de funções de pertinência e o número de neurônios podem ser modificados simultaneamente com a atualização dos pesos. As funcionalidades propostas para a rede evolutiva têm por objetivo manter um erro mínimo e uniforme para todas as regiões do espaço de entrada e permitir a adaptação dos parâmetros e estrutura da rede a fim de lidar com as mudanças no fluxo de dados.

O aprendizado da eNFN pode ser sumarizado em cinco etapas:

1. Definir os valores iniciais dos parâmetros das funções de pertinência. Esse passo é computado apenas uma vez, utilizando os limites inferior e superior para realizar a granularização inicial do domínio das variáveis de entrada.

2. Verificar se o valor da entrada atual é maior que max_{x_i} ou menor que min_{x_i}, e decidir se o valor de max_{x_i} ou min_{x_i} deve ser atualizado.

3. Calcular os graus de pertinência da entrada x_{ti}, encontrar a função de pertinência mais ativa e atualizar o valor modal da função de pertinência mais ativa.

4. Verificar se a função de pertinência mais ativa representa bem a vizinhança de x_{ti}, e decidir se uma função de pertinência deve ser criada e inserida para cobrir a vizinhança de x_{ti}. Em outras palavras, este passo atualiza (refina) a granularização do espaço de entrada.

5. Encontrar a função de pertinência que está a mais tempo inativa, e decidir se essa função deve ser excluída por inatividade. Uma função será eliminada se o tempo de inatividade for maior que um limiar determinado *a priori*.

A seguir, apresentar-se-á uma descrição detalhada dessas cinco etapas.

4.2 Inicialização das Funções de Pertinência

As funções de pertinência utilizadas na eNFN são triangulares e complementares. O número inicial de funções de pertinência m_i pode ser definido empiricamente, ou com base no conhecimento *a priori* do sistema, ou baseado em alguma técnica de agrupamento (Caminhas, 1997). Neste trabalho, optou-se por iniciar o algoritmo com a granularização padrão de duas funções de pertinência para cada uma das n variáveis ($m_i = 2, \;\; i = 1, ..., n$). O valor modal inicial das funções de pertinência é encontrado como se segue:

$$\begin{aligned} b_{i1} &= min_{x_i}, \\ b_{i2} &= max_{x_i}, \end{aligned} \tag{4.1}$$

onde min_{x_i} é o limite inferior e max_{x_i} o limite superior do domínio da i-ésima variável de entrada.

A granularização inicial do espaço de entrada é refinada por meio da inserção e/ou da exclusão de funções de pertinência de acordo com o fluxo de dados. O processo de criação e inserção de funções de pertinência é detalhado na Seção 4.5 e o de exclusão na Seção 4.6.

4.3 Adaptação de Contexto

Na modelagem de sistemas, podem ocorrer mudanças no ambiente que permitam o surgimento de dados cujos valores estão fora do domínio definido pelos valores máximo (max_{x_i})

e mínimo (min_{x_i}). Portanto, nesse caso, é importante a atualização dos valores de max_{x_i} e min_{x_i} para se adaptarem às mudanças nas condições ambientais. Este procedimento é chamado *adaptação de contexto* e é definido como se segue:

$$\begin{array}{l} se\ x_{t_i} > max_{x_i}\ então\ max_{x_i} = x_{t_i}\ e\ b_{im_i} = max_{x_i} \\ se\ x_{t_i} < min_{x_i}\ então\ min_{x_i} = x_{t_i}\ e\ b_{i1} = min_{x_i}, \end{array} \tag{4.2}$$

ou seja, os valores máximo e mínimo de cada variável de entrada são atualizados a medida que o algoritmo recebe novas amostras que ultrapassam esses limiares.

4.4 Atualização do Valor Modal

Nesta etapa, encontra-se b_i^*, índice da função de pertinência mais ativada por $x_{ti},\ i = 1, ..., n$. Se $b_i^* > 1$ e $b_i^* \neq m_i$, então, a b_i^*-ésima função de pertinência terá seu valor modal atualizado por:

$$b_{b_i^*}^{novo} = b_{b_i^*}^{atual} + \beta(x_{ti} - b_{b_i^*}^{atual}), \tag{4.3}$$

onde β é a taxa de aprendizado. Caso contrário, se $b_i^* = 1$ ou $b_i^* = m_i$, o valor modal não será atualizado, pois neste caso os valores modais correspondem a min_{x_i} e max_{x_i}, respectivamente.

4.5 Criação e Inserção de Funções de Pertinência

A criação e inserção de funções de pertinência refina a granularização do espaço de entrada visando reduzir o erro de saída de maneira uniforme. Ao contrário do método proposto em Wang et al. (2010), no qual esse refinamento é realizado para a região com o maior erro médio, neste trabalho, o refinamento é realizado com base no erro da função de pertinência mais ativa.

O valor médio do erro local das regras correspondentes às funções de pertinência mais ativas é comparado com o valor médio do erro global de modelagem. Se o valor médio do erro local for maior que o erro médio global, então, essa região é refinada pela adição de funções de pertinência como se segue.

O valor médio $\hat{\mu}_{g_t}$ e a variância $\hat{\sigma}_{g_t}^2$ do erro global é recursivamente calculado para a entrada x_t por:

$$\hat{\mu}_{g_t} = \hat{\mu}_{g_{t-1}} - \beta(\hat{\mu}_{g_{t-1}} - e_t), \tag{4.4}$$

$$\hat{\sigma}_{g_t}^2 = (1 - \beta)(\hat{\sigma}_{g_{t-1}}^2 + \beta(\hat{\mu}_{g_t} - e_t)^2), \tag{4.5}$$

onde $e_t = y_t - \hat{y}_t$.

Similarmente, o valor médio $\hat{\mu}_{b_{ti}^*}$ do erro local correspondente à função de pertinência mais ativa (b_i^*) é recursivamente computado para a entrada x_{ti} usando:

$$\hat{\mu}_{b_{ti}^*} = \hat{\mu}_{b_{ti-1}^*} - \beta(\hat{\mu}_{b_{ti-1}^*} - e_t). \tag{4.6}$$

A fim de se evitar uma granularidade muito fina de uma determinada região do domínio de entrada, insere-se um limitador (τ). Este é utilizado para limitar a menor distância ($dist$) permitida entre o valor modal da função criada e o valor modal das funções adjacentes. Se $b_i^* > 1$ e $b_i^* \neq m_i$, então, a distância entre os valores modais das funções de pertinência é dada por (4.8). Caso contrário, se $b_i^* = 1$ ou $b_i^* = m_i$, então, a distância é calculada por (4.11) e (4.13), respectivamente.

O objetivo não é definir o número de regras *a priori*, pois este é um resultado do algoritmo. O que se faz é limitar a granularização excessiva de uma determinada região do domínio de entrada. Indiretamente, esta restrição permite manter o controle sobre o número de regras, evitando modelos complexos e *overfitting*. Este limite só é válido na criação de funções de pertinência. É possível, por meio da atualização do valor modal, que duas funções adjacentes possuam uma distância inferior a este limite.

Mais precisamente, se $\hat{\mu}_{b_{ti}^*} > \hat{\mu}_{g_t} + \hat{\sigma}_{g_t}^2$ e $dist > \tau$, então, uma nova função de pertinência é criada e inserida. O limiar τ é calculado iterativamente usando:

$$\tau = \frac{max_{x_i} - min_{x_i}}{\eta}, \tag{4.7}$$

onde η é um parâmetro definido empiricamente.

A criação e inserção de funções de pertinência atualiza a granularização da i-ésima variável do domínio de entrada como se segue:

- **Caso 1:** Se $b_i^* > 1$ e $b_i^* \neq m_i$, então, a função mais ativa será substituída por duas funções de pertinência cujos valores modais são calculados por (4.9) e (4.10). Esse procedimento é apresentado na Figura 4.1a que mostra a granularização do domínio de entrada antes e na Figura 4.1b, após a criação e inserção das novas funções de pertinência.

$$dist = \frac{(b_{i,b_i^*+1} - b_{i,b_i^*-1})}{3}. \tag{4.8}$$

$$b_{i,novo_1} = b_{i,b_i^*-1} + \frac{(b_{i,b_i^*+1} - b_{i,b_i^*-1})}{3}. \tag{4.9}$$

$$b_{i,novo_2} = b_{i,b_i^*-1} + 2 * \frac{(b_{i,b_i^*+1} - b_{i,b_i^*-1})}{3}. \tag{4.10}$$

- **Caso 2:** Se a função mais ativa for a primeira ($b_i^* = 1$), então, uma nova função de pertinência será criada e inserida entre a primeira e a segunda. O valor modal da função criada é dado por (4.12). A Figura 4.2a mostra o espaço de entrada e a função mais ativa antes, e a Figura 4.2b apresenta a granularização do espaço de entrada após a criação e inserção de uma nova função de pertinência.

$$dist = \frac{(b_{i,b_i^*+1} - b_{i,b_i^*})}{2}. \tag{4.11}$$

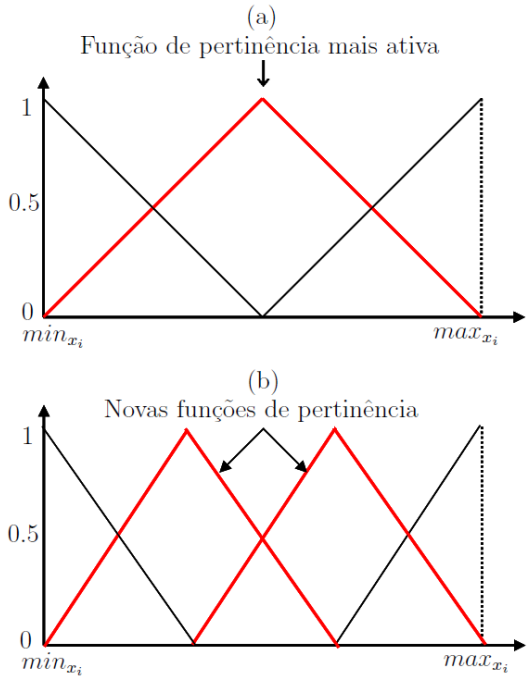

Fig. 4.1: Criação e inserção de função de pertinência - Caso 1.

$$b_{i,novo} = b_{i,b_i^*} + \frac{(b_{i,b_i^*+1} - b_{i,b_i^*})}{2}. \tag{4.12}$$

- **Caso 3:** Se a função mais ativa for a última ($b_i^* = m_i$), então, uma nova função de pertinência será criada entre a penúltima e a última e terá seu valor modal conforme descrito por (4.14). A Figura 4.3a mostra o espaço de entrada antes da criação e inserção da função de pertinência, enquanto a Figura 4.3b após.

$$dist = \frac{(b_{i,b_i^*} - b_{i,b_i^*-1})}{2}. \tag{4.13}$$

$$b_{i,novo} = b_{i,b_i^*} - \frac{(b_{i,b_i^*} - b_{i,b_i^*-1})}{2}. \tag{4.14}$$

Nos três casos apresentados, as funções criadas recebem o valor do erro médio estimado para a função mais ativa e o número de funções para aquela variável é atualizado. A criação e inserção de funções de pertinência modifica os limites inferior e/ou superior das funções de pertinência adjacentes, mas seus valores modais permanecem inalterados. A alteração nos limites das funções adjacentes ocorre para manter as funções de pertinência complementares.

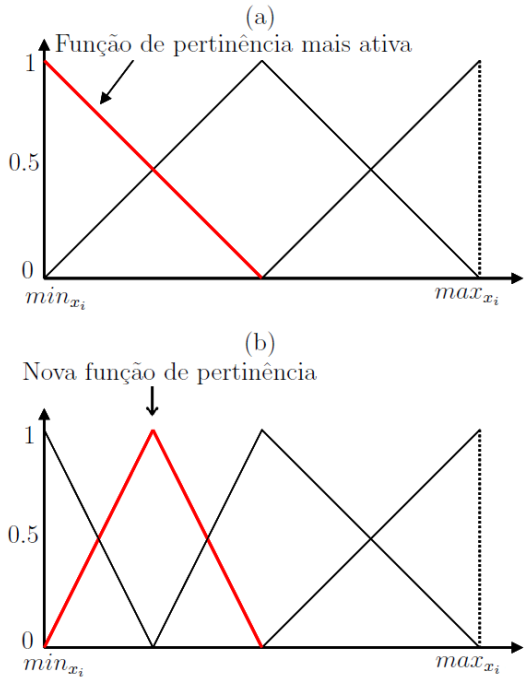

Fig. 4.2: Criação e inserção de função de pertinência - Caso 2.

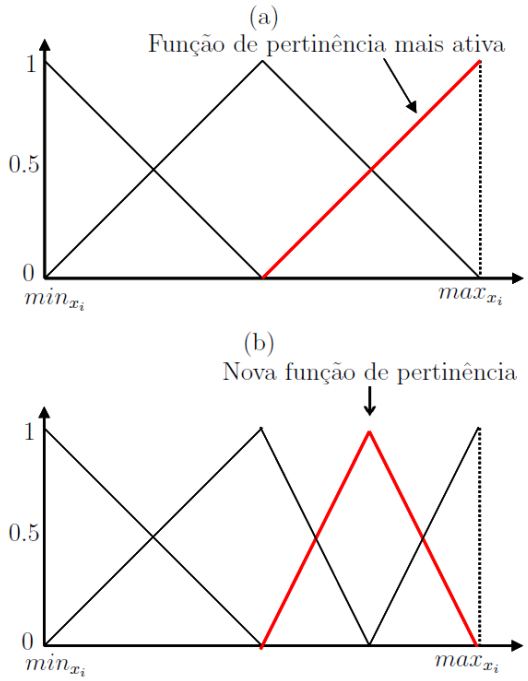

Fig. 4.3: Criação e inserção de função de pertinência - Caso 3.

4.6 Exclusão de Funções de Pertinência

Esta seção descreve o método para exclusão de funções de pertinência (regras *fuzzy*) com uma abordagem baseada no conceito de idade da regra. Em Angelov and Filev (2005), o conceito de idade da regra foi introduzido, e, em Lughofer and Angelov (2011), ele foi estendido. Neste trabalho, o conceito de idade da regra é empregado para determinar o tempo de inatividade de uma função de pertinência, isto é, o intervalo temporal que uma função de pertinência fica sem ser ativada. A idade da função de pertinência é dada por:

$$idade_j = t - ativa_j, \tag{4.15}$$

onde j é o índice da função de pertinência, *ativa* denota a instância de tempo que a j-ésima função de pertinência foi ativada, e t é a instância de tempo atual.

A abordagem proposta permite a exclusão de funções de pertinência que estão inativas por uma instância de tempo determinada *a priori*. Para cada variável de entrada i esse procedimento encontra b_i^-, índice da função de pertinência com maior tempo de inatividade. A função de pertinência indexada por b_i^- será excluída se:

$$idade_{b_i^-} > \omega \; e \; m_i > 2, \tag{4.16}$$

onde $idade_{b_i^-}$ computa a instância de tempo que a função indexada por b_i^- esta inativa, e ω é um parâmetro que indica o limiar de tempo de inatividade das funções de pertinência.

Após a exclusão de uma função de pertinência, a granularização do domínio da variável de entrada é realizada como se segue:

- **Caso 1:** Se $b_i^- > 1$ e $b_i^- \neq m_i$, então, a função indexada por b_i^- será excluída e as funções adjacentes terão seus limites superior e inferior modificados. Esse procedimento é ilustrado na Figura 4.4a que mostra a granularização do domínio de entrada antes e na Figura 4.4b após a exclusão da função de pertinência. No exemplo ilustrado na Figura 4.4, A_{i4} é a função de pertinência indexada por b_i^-. Na exclusão de A_{i4}, o limite superior de A_{i3} e o limite inferior de A_{i5} são modificados.

- **Caso 2:** Se $b_i^- = 1$, então, a função indexada por b_i^- será excluída e o valor modal da função adjacente será definido pelo valor de min_{x_i}. A Figura 4.5a mostra o espaço de entrada e a função indexada por b_i^- (A_{i1}) antes e a Figura 4.5b a granularização do espaço de entrada após a exclusão da função de pertinência. A exclusão da função de pertinência modifica o valor modal de A_{i2} e o limite inferior de A_{i3}, como pode ser visto na Figura 4.5b.

- **Caso 3:** Se $b_i^- = m_i$, então, a função indexada por b_i^- será excluída e o valor modal da função adjacente será definido pelo valor de max_{x_i}. A exclusão da função de pertinência modifica o limite superior da função adjacente a que teve seu valor modal modificado, como mostra as figuras 4.6a e 4.6b.

O Algoritmo 3 sumariza a abordagem proposta para estimação da saída, atualização dos parâmetros e evolução da estrutura da rede eNFN com uma única saída.

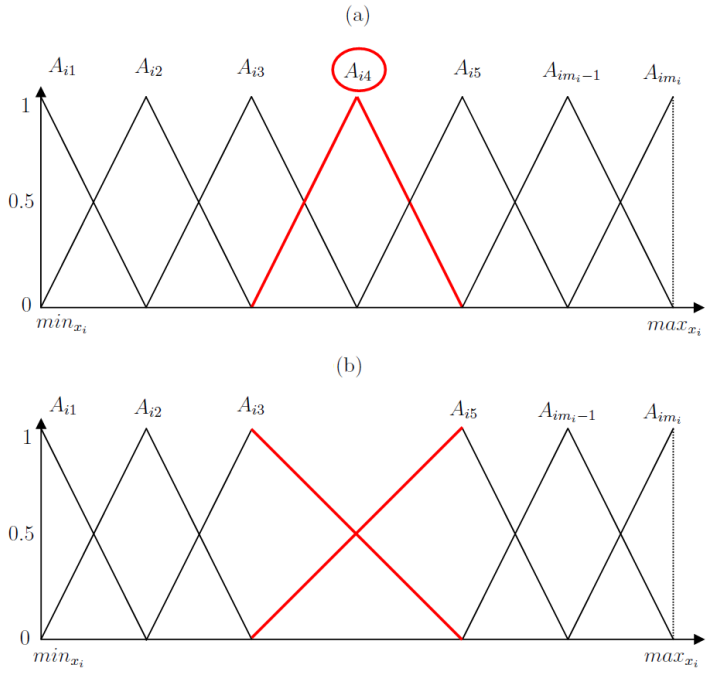

Fig. 4.4: Exclusão de função de pertinência - Caso 1.

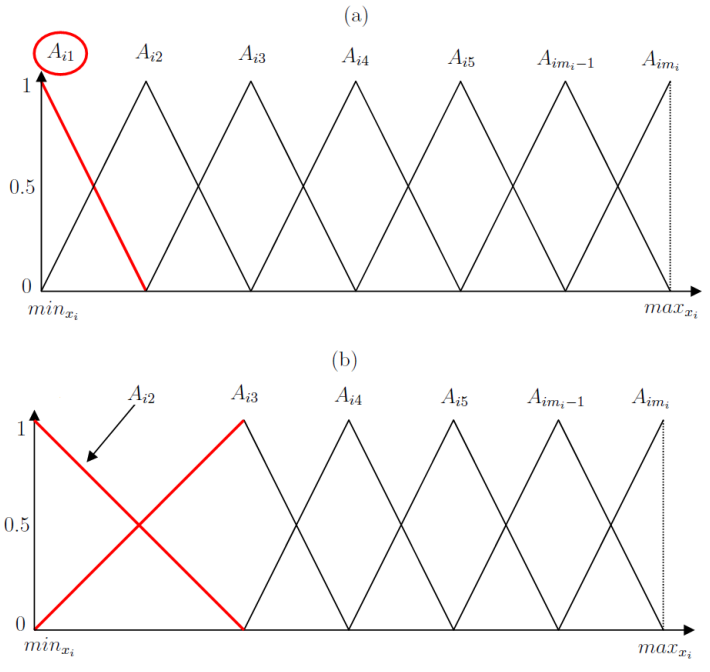

Fig. 4.5: Exclusão de função de pertinência - Caso 2.

Algorithm 3: Rede eNFN com uma única saída.

Entrada: x_t, $\hat{y}_t$, β, n, η, w;
Saída: y_t;
Inicializar b_{ij};
for $t = 1, 2, ...$ **do**
  Ler x_t, $\hat{y}_t$;
  Verificar a adaptação de contexto;
  //**Cálculo da Saída**;
  for $i = 1 : n$ **do**
    Calcular $\mu_{A_{ik_i}}(x_{ti})$, $\mu_{A_{ik_i+1}}(x_{ti})$;
    Calcular y_{ti};
  end for
  Calcular y_t;
  //**Atualização dos Parâmetros**;
  Calcular α;
  Calcular $\hat{\mu}_{g_t}$;
  Calcular $\hat{\sigma}^2_{g_t}$;
  Atualizar b_{ij};
  Atualizar q_{ik_i}, q_{ik_i+1};
  for $i = 1 : n$ **do**
    // **Criação e inserção de funções de pertinência**;
    Encontrar b_i^*;
    Calcular $\hat{\mu}_{b^*_{ti}}$;
    Calcular τ;
    Calcular $dist$;
    if $(\hat{\mu}_{b^*_{ti}} > \hat{\mu}_{g_t} + \hat{\sigma}^2_{g_t})$ *e* $(dist > \tau)$ **then**
      Criar e inserir funções de pertinência;
      Atualizar os parâmetros;
    end if
    // **Exclusão de funções de pertinência**;
    Atualizar $idade_i$;
    Encontrar b_i^-;
    if $(idade_{b_i^-} > \omega)$ *e* $(m_i > 2)$ **then**
      Excluir a função de pertinência indexada por b_i^-;
      Atualizar os parâmetros;
    end if
  end for
end for

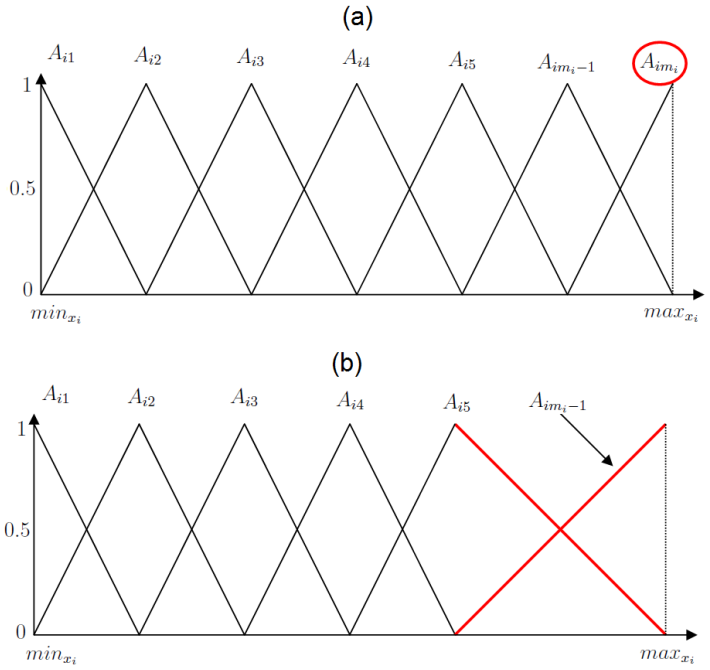

Fig. 4.6: Exclusão de função de pertinência - Caso 3.

4.7 eNFN com Múltiplas Saídas

Basicamente, aqui é introduzida uma generalização da rede eNFN para permitir o cálculo de múltiplas saídas. Os procedimentos para a eNFN com múltiplas saídas são similares aos descritos para a rede eNFN com uma única saída. Portanto, essa seção apresenta somente as modificações realizadas na estrutura da eNFN para permitir o cálculo de múltiplas saídas.

Conforme descrito no Capítulo 3, a principal vantagem da rede com múltiplas saídas está no seu menor custo computacional quando comparado com várias redes de única saída. A diferença entre o custo computacional da rede eNFN com múltiplas saídas e o de várias redes eNFN com uma única saída é maior que no caso da rede NFN (Capítulo 3). Isto ocorre porque na eNFN além do cálculo do grau de ativação das funções de pertinência, o procedimento de evolução da estrutura da rede também é realizado somente uma vez para cada entrada. Ao contrário do que em várias redes de única saída, onde estes cálculos devem ser realizados para cada uma das redes.

Na eNFN com múltiplas saídas, cada uma das saídas da rede y_t^l é obtida pela soma das saídas individuais y_{ti}^l como descrito por:

$$y_t^l = \sum_{i=1}^{n} y_{ti}^l = \sum_{i=1}^{n} (\mu_{A_{ik_i}}(x_{ti}) q_{ik_i}^l + \mu_{A_{ik_i+1}}(x_{ti}) q_{ik_i+1}^l), \tag{4.17}$$

onde l é o índice de cada saída da eNFN, t o passo atual, n número de variáveis de entrada, k_i and $k_i + 1$ os índices das duas funções de pertinência ativadas para i-ésima variável de entrada.

Os pesos da rede eNFN com múltiplas saídas são ajustados de forma análoga ao descrito

na Seção 3.4. Aqui, vale ressaltar que, para cada uma das saídas da rede existe um conjunto de pesos, isto é, o conjunto de pesos é específico para cada saída.

Na criação e inserção de funções de pertinência são calculadas, recursivamente, a soma da média do erro e a soma da variância do erro de modelagem global de todas as saídas, e estas são divididas pelo número de saídas, como se segue:

$$\mu\hat{\mu}_{g_t} = \frac{\sum_{l=1}^{s} \hat{\mu}_{g_t^l}}{s} = \frac{\sum_{l=1}^{s} \hat{\mu}_{g_{t-1}^l} - \beta(\hat{\mu}_{g_{t-1}^l} - e_t)}{s}, \tag{4.18}$$

$$\sigma\hat{\sigma}^2_{g_t} = \frac{\sum_{l=1}^{s} \hat{\sigma}^2_{g_t^l}}{s} = \frac{\sum_{l=1}^{s} (1-\beta)(\hat{\sigma}^2_{g_{t-1}^l} + \beta(\hat{\mu}_{g_t^l} - e_t)^2)}{s}, \tag{4.19}$$

onde β é a taxa de aprendizado, e s o número de saídas.

O valor médio $\hat{\mu}_{b^*_{ti}}$ (4.6) do erro local correspondente à função mais ativa, o limitador τ (4.7) e a distância $dist$ (4.8, 4.11, 4.13) são calculados de forma idêntica aos da Seção 4.5. Assim, uma nova função de pertinência é criada e inserida se $(\hat{\mu}_{b^*_{ti}} > \mu\hat{\mu}_{g_t} + \sigma\hat{\sigma}^2_{g_t})$ e $(dist > \tau)$.

O procedimento para exclusão de função de pertinência é idêntico ao introduzido na Seção 4.6, portanto, não será descrito novamente. O Algoritmo 4 detalha a rede eNFN com múltiplas saídas.

4.8 Parâmetros da Rede eNFN

O algoritmo de aprendizado evolutivo da eNFN (com uma única ou múltiplas saídas) possui quatro parâmetros:

- m_i: número de funções de pertinência que granulariza o domínio da i-ésima variável de entrada, $i = 1 \ldots n$. São necessárias ao menos duas funções de pertinência triangulares e complementares para cobrir o domínio de cada variável de entrada. Portanto, a granulação deve iniciar com duas ou mais funções de pertinência, isto é, $m_i \geq 2$. Em todos os experimentos computacionais relatados neste trabalho, o domínio de cada variável de entrada é inicialmente granulado por duas funções de pertinência ($m_i = 2$) e, se necessário, novas funções de pertinência podem ser adicionadas em função dos dados de entrada e do erro de modelagem.

- β: taxa de aprendizado utilizada para atualizar os valores modais das funções de pertinência, para calcular o valor da média e da variância do erro de modelagem global, e para calcular o erro médio da função de pertinência mais ativa. A taxa de aprendizado β é usualmente definida com um valor pequeno, por exemplo, $\beta \in [10^{-3}\,,\, 10^{-1}]$.

- η: parâmetro auxiliar para cálculo da menor distância permitida entre o valor modal da função de pertinência a ser criada e os valores modais de suas funções adjacentes. Note que, indiretamente, este parâmetro limita o número de regras. Valores de η são escolhidos considerando o seguinte: valores altos de η podem não ser suficientes para capturar as informações presentes nos dados de entrada, com isso, podem diminuir a acurácia da rede; valores baixos de η aumentam a complexidade da rede e podem causar *overfitting*. A prática sugere $\eta \in [5\,,\, 25]$.

Algorithm 4: Rede eNFN com múltiplas saídas.

Entrada: x_t, $\hat{y}_t$, β, n, s, η, w;
Saída: y_t;
Inicializar b_{ij};
for $t = 1, 2, ...$ **do**
  Ler x_t, $\hat{y}_t$;
  Verificar a adaptação de contexto;
  for $i = 1 : n$ **do**
    Calcular $\mu_{A_{ik_i}}(x_{ti})$, $\mu_{A_{ik_i+1}}(x_{ti})$;
  end for
  for $l = 1 : s$ **do**
    for $i = 1 : n$ **do**
      Calcular y_{ti}^l;
    end for
    Calcular y_t^l;
    Calcular α;
    Calcular $\hat{\mu}_{g_t^l}$;
    Calcular $\hat{\sigma}^2_{g_t^l}$;
    Atualizar $q_{ik_i}^l$, $q_{ik_i+1}^l$;
  end for
  Atualizar b_{ij};
  Calcular $\mu\hat{\mu}_{g_t}$;
  Calcular $\sigma\hat{\sigma}^2_{g_t}$;
  for $i = 1 : n$ **do**
    // **Criação e inserção de funções de pertinência**;
    Encontrar b_i^*;
    Calcular $\hat{\mu}_{b_{ti}^*}$;
    Calcular τ;
    Calcular $dist$;
    if $(\hat{\mu}_{b_{ti}^*} > \mu\hat{\mu}_{g_t} + \sigma\hat{\sigma}^2_{g_t})$ *e* $(dist > \tau)$ **then**
      Criar e inserir funções de pertinência;
      Atualizar os parâmetros;
    end if
    // **Exclusão de funções de pertinência**;
    Atualizar $idade_i$;
    Encontrar b_i^-;
    if $(idade_{b_i^-} > \omega)$ *e* $(m_i > 2)$ **then**
      Excluir a função de pertinência indexada por b_i^-;
      Atualizar os parâmetros;
    end if
  end for
end for

- ω: limiar para exclusão de funções de pertinência pela idade. Este limiar empregado para excluir funções de pertinência está associado com a habilidade de a rede esquecer conhecimento antigo sempre que este se torna inútil. O valor de ω geralmente é definido entre 50 e 250.

4.9 Complexidade Computacional

Existem diversas maneiras de avaliar o desempenho de um algoritmo, como, por exemplo, a acurácia, a simplicidade, o tempo de processamento, a quantidade de memória requerida e a otimalidade (Toscani and Veloso, 2012). Nesta seção, realiza-se a análise da complexidade computacional dos algoritmos propostos, isto é, a análise da complexidade temporal e espacial.

Análise de complexidade computacional de algoritmos é a determinação da quantidade de recursos necessários para executá-los, em outras palavras, ela fornece estimativas teóricas/práticas dos recursos necessários para um algoritmo resolver determinado problema computacional (Cormen et al., 2002). A complexidade de um algoritmo relata o custo em termos de processamento e armazenamento requerido para executá-lo, e pode ser expressa por uma notação matemática ou ser obtida experimentalmente pela execução do algoritmo em um computador real.

Nesta seção, a complexidade dos algoritmos é obtida por meio da expressão matemática, isto é, pela sua notação assintótica. Avalia-se tanto a complexidade temporal, que relaciona o tamanho da entrada com o tempo de execução do algoritmo, quanto a espacial, que relaciona o tamanho da entrada com o espaço de armazenamento requerido. No próximo capítulo, os algoritmos são avaliados experimentalmente pela acurácia e pelo tempo de execução.

Geralmente, a maioria dos algoritmos possui um baixo custo computacional quando o número de entradas n é pequeno. Na análise do custo computacional pela notação assintótica considera-se o comportamento do algoritmo para valores muito grandes de n, isto é, o comportamento assintótico de um algoritmo representa o limite de custo quando o número de entradas n cresce (Cormen et al., 2002; Ziviani, 2006).

Pela notação assintótica, um algoritmo pode ser avaliado pelo melhor caso (Ordem Ω), pelo caso médio (Ordem Θ) e pelo pior caso (Ordem O). Neste trabalho o algoritmo será avaliado pelo pior caso. A complexidade de um algoritmo pode ser classificada como se segue: $O(1)$ custo constante; $O(\log n)$ complexidade logarítmica; $O(n)$ complexidade linear; $O(n^2)$ complexidade quadrática ; $O(n^3)$ complexidade cúbica; $O(n^c)$ complexidade polinomial (onde c é uma constante); $O(c^n)$ complexidade exponencial.

4.9.1 Complexidade Temporal

Na análise da complexidade temporal da rede eNFN com uma única saída (eNFN_1), considera-se apenas o número de variáveis de entrada. O número de funções de pertinência e/ou número de regras não interfere no custo computacional do algoritmo, pois independente do número de funções de pertinência somente duas funções são ativadas para cada variável de entrada. Assim, a eNFN_1 é $O(n)$, onde n é o número de variáveis de entrada. Para a rede eNFN com múltiplas saídas (eNFN_S), a complexidade temporal envolve o número de variáveis de entrada n e o número de saídas s. Portanto, a rede eNFN_S é $O(ns)$.

A Tabela 4.1 apresenta a complexidade temporal das redes propostas e a compara com quatro modelos evolutivos alternativos (DENFIS (Kasabov and Song, 2002), eMG (Lemos et al., 2011c), eTS (Angelov and Filev, 2004) e xTS(Angelov and Zhou, 2006)), onde n é o número de variáveis de entrada, r o número de regras, e s o número de saídas. Estes modelos alternativos representam o atual estado da arte na modelagem adaptativa de sistemas e processos.

Tab. 4.1: Complexidade temporal dos algoritmos evolutivos.

Modelo	Custo Temporal
DENFIS	$O(rn^2)$
eMG	$O((rn)^3)$
eNFN_1	$O(n)$
eNFN_S	$O(ns)$
eTS	$O(rn^3)$
xTS	$O(rn^3)$

Como era de se esperar, as redes eNFN possuem a menor complexidade temporal entre os algoritmos avaliados. Um fator importante que deve ser considerado, além da complexidade temporal, é a complexidade das operações executadas pelos algoritmos. A complexidade dos cálculos computacionais pode ser um fator de grande influência no custo temporal dos algoritmos. Analisando esse fator, percebe-se que os cálculos e equações para avaliar, atualizar os parâmetros e estimar a saída das redes propostas são muito mais simples do que de outros modelos evolutivos, pois as redes eNFN realizam somente operações de multiplicação, adição e comparação (Silva et al., 2014c).

Note que, o custo temporal das redes eNFN não se altera com a inclusão de novas regras ao modelo, i. e., as redes eNFN possuem um custo temporal constante durante a sua execução, devido ao seu custo temporal ser determinado pelo número de variáveis de entrada e pelo número de saídas (no eNFN com múltiplas saídas). Alternativamente, pode-se dizer que o custo temporal das redes eNFN é invariante no tempo. Nos modelos evolutivos, como pode ser visto na Tabela 4.1, geralmente a complexidade temporal é definida pela relação entre o número de variáveis de entrada e o número de regras. Em outras palavras, isto quer dizer que o custo temporal desses algoritmos aumenta à medida que novas regras são inseridas no modelo. Sumarizando, nos modelos evolutivos o custo temporal pode aumentar durante sua execução em função da granularidade (número de regras), o que não acontece com as redes eNFN que possuem um custo temporal constante.

4.9.2 Complexidade Espacial

Nesta seção objetiva-se avaliar a complexidade espacial das abordagens propostas. Para tal, assume-se o custo espacial para cada tipo de constante, variável ou parâmetro, como se segue: 2 bytes para inteiro (*short int*); 4 bytes para número real (*float* - número em ponto flutuante de precisão simples) (Jamsa and Klander, 1999).

A complexidade espacial das redes propostas, isto é, o custo espacial para armazenar todas as constantes, variáveis e parâmetros são mostrados na Tabela 4.2, onde n é o número de variáveis de entrada, m o número de funções de pertinência, e s o número de saídas.

Tab. 4.2: Complexidade espacial dos algoritmos propostos.

Modelo	Custo Espacial
eNFN_1	$62 + 32n + 20nm$
eNFN_S	$56 + 32n + 16nm + 8s + 48snm$

Para ilustrar os resultados ilustrados na Tabela 4.2, apresenta-se a seguir a curva de crescimento do custo de armazenamento. A Figura 6.3 mostra o crescimento do custo espacial para as redes eNFN (eNFN_1 com uma única saída, eNFN_{S2} com duas saídas, eNFN_{S3} com três saídas, e eNFN_{S4} com quatro saídas). O gráfico da Figura 6.3 foi gerado considerando o custo espacial ilustrado na Tabela 4.2, o número de funções de pertinência $m = 10$, e o número de entradas de $n = 1, \ldots, 20$, e para a rede a eNFN_S o número de saídas $s = 2, \ldots, 4$.

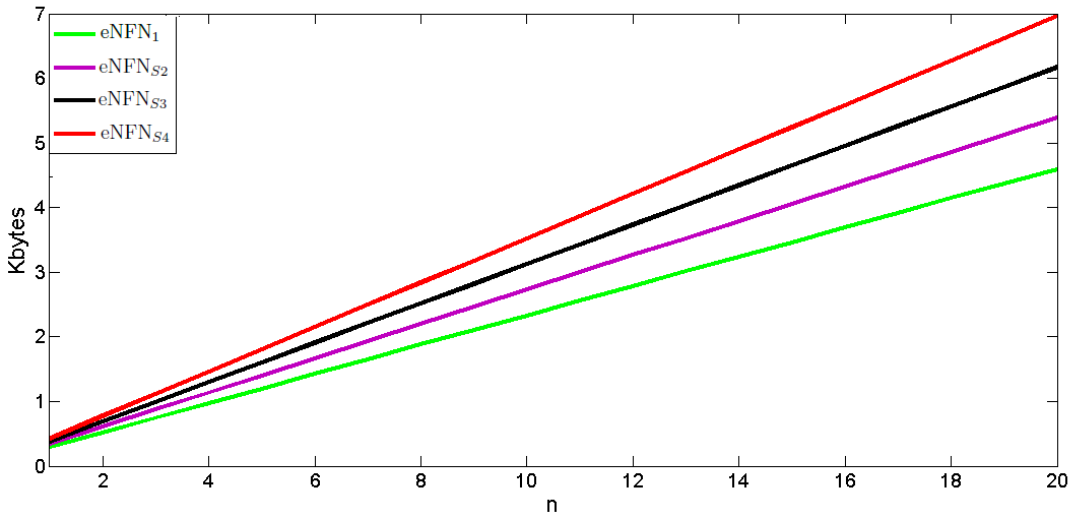

Fig. 4.7: Custo espacial para as redes eNFN.

Conforme esperado, as redes eNFN possuem um crescimento linear no custo espacial com o aumento no número de variáveis de entrada, as redes eNFN_S possuem um crescimento linear em relação ao aumento no número de saídas, e o custo espacial da rede eNFN_S com s saídas é menor do que o de s redes eNFN_1 (com uma única saída).

É importante salientar que, apesar do aumento da disponibilidade do espaço de armazenamento em sistemas computacionais, em diversas aplicações práticas o espaço de armazenamento é um fator crítico.

4.10 Resumo

Este capítulo introduziu um novo algoritmo de aprendizado evolutivo para a rede *neuro-fuzzy* descrita no Capítulo 3. Inicialmente é descrito uma rede evolutiva (eNFN) com uma única saída e em seguida, esta rede é generalizada para múltiplas saídas. A eNFN utiliza funções de pertinência triangulares cujos parâmetros e número de funções evoluem de acordo com um fluxo de dados. Na análise da complexidade, as redes eNFN apresentaram uma baixa complexidade computacional (temporal e espacial), e se mostraram como uma alternativa viável e promissora para aplicações em sistemas dinâmicos de tempo real com alta taxa de amostragem. Evidentemente, as redes eNFN também podem ser utilizadas para sistemas lentos.

Capítulo 5

Resultados de Simulações e Experimentais - Redes eNFN

Neste capítulo, o desempenho da rede eNFN é avaliado em simulações computacionais e em experimentos em tempo real. São avaliados a acurácia e o tempo de processamento. A Seção 5.2 apresenta os resultados da rede eNFN na identificação de sistema não linear e em problemas de previsão. A análise do tempo de processamento é descrita na Seção 5.3 e a relação entre a dimensão do espaço de entrada/saída e o tempo de processamento é o assunto da Seção 5.4. A Seção 5.5 discorre sobre os experimentos computacionais em tempo real utilizando o módulo de levitação magnética (*Magnetic Levitation System* - MagLev) e o sistema MIMO de duplo rotor (*Twin Rotor MIMO System* - TRMS), ambos produzidos pela Feedback Instruments Limited. Concluindo o capítulo, a Seção 5.6 apresenta uma discussão final sobre o desempenho das redes eNFN.

5.1 Introdução

O eNFN é o algoritmo evolutivo para a rede NFN e foi desenvolvido com foco na sua aplicação sistemas com alta taxa de amostragem, ou seja, com dinâmica rápida. Neste tipo de sistema o tempo de processamento e de atualização dos parâmetros é uma restrição importante, que pode inviabilizar o uso de modelos com alta complexidade computacional e dificultar o uso de modelos que possuem muitos parâmetros. Isso porque, o modelo deve estimar a saída e atualizar os parâmetros a cada nova amostra de dados em tempo hábil para evitar o acúmulo de amostras. Assim, busca-se algoritmos eficientes, com baixo custo computacional e com boa precisão e que estes gerem modelos com estrutura compacta e parcimoniosa.

5.2 Resultados de Simulações Computacionais

Esta seção descreve simulações computacionais realizadas para avaliar a rede eNFN[1]. A rede foi avaliada na identificação de processo não linear e em problemas de previsão. Os resultados

[1]A rede eNFN foi desenvolvida em Matlab.

obtidos são comparados com algoritmos de modelagem evolutivos alternativos como DENFIS[2] (Kasabov and Song, 2002), eMG[3] (Lemos et al., 2011c), eTS[4] (Angelov and Filev, 2004) e xTS[4] (Angelov and Zhou, 2006). Estes algoritmos são representativos do estado da arte corrente em modelagem adaptativa de processos e sistemas.

A Seção 5.2.1 apresenta a metodologia empregada nas simulações e as medidas de desempenho utilizadas na avaliação. Os resultados obtidos na identificação de processo não linear são apresentados na Seção 5.2.2 e os resultados das simulações em problemas de previsão nas seções 5.2.4 e 5.2.3.

5.2.1 Metodologia

Os sistemas evolutivos são desenvolvidos, principalmente, para aplicações dinâmicas não estacionárias e variantes no tempo. Nestas circunstâncias, as técnicas tradicionais utilizadas para estimar os parâmetros, tais como validação cruzada, não são úteis. Nos experimentos relatados neste trabalho optou-se por utilizar a abordagem proposta em Silva et al. (2014c) para estimar os parâmetros e avaliar o desempenho dos modelos. Nesta abordagem, o conjunto de dados é dividido em dois subconjuntos com 50% das amostras cada um. O primeiro subconjunto é utilizado para estimar os parâmetros e o segundo para avaliar o desempenho. Os melhores parâmetros são obtidos através de busca exaustiva. Valores de parâmetros que alcançam o menor erro são utilizados para avaliar a modelagem e o desempenho. A Tabela 5.1 mostra os parâmetros e a faixa de variação destes na busca exaustiva pelo melhor conjunto de parâmetros para cada um dos modelos evolutivos avaliados.

Tab. 5.1: Faixa de variação dos parâmetros dos modelos evolutivos.

Modelo	Parâmetro	*Step*
DENFIS	$dthr = 0.01, ..., 0.10$	0.01
	$mofn = 1, ..., 10$	1.00
eMG	$\lambda = 0.05$	fixo
	$w = 5, ..., 40$	5.00
	$\sum_{init} = 10^{-1}, ..., 10^{-4}$	1.00
	$\alpha = 0.01$	fixo
eNFN	$\beta = 0.01$	fixo
	$\omega = 50, ..., 250$	50.00
	$\eta = 5, ..., 20$	5.00
eTS	$r = 0.02, ..., 0.10$	0.02
	$\Omega = 100, ..., 900$	50.00
xTS	$\Omega = 100, ..., 900$	50.00

[2]A implementação em Matlab utilizada para o DENFIS está disponível no *Knowledge Engineering and Discovery Research Institute* (KEDRI) - http://www.aut.ac.nz/research/research-institutes/kedri/books.

[3]A implementação em Matlab utilizada para o eMG foi cedida pelos autores.

[4]A implementação em Java utilizada para o eTS e o xTS foi fornecida pelos respectivos autores.

Os modelos são avaliados pelo *RMSE - Root Mean Squared Error*:

$$RMSE = \frac{1}{N}(\sum_{t=1}^{N}(y_t - \hat{y}_t))^{\frac{1}{2}}, \tag{5.1}$$

onde N é o número de dados do conjunto de teste, $\hat{y}_t$ é a saída desejada, y_t é a saída estimada. As simulações computacionais foram executadas de forma *on-line*, isto é, durante a estimação dos parâmetros e avaliação do desempenho os modelos evoluem sua estrutura para todas as amostras do conjunto de dados.

Os modelos evolutivos adaptam seus parâmetros e sua estrutura continuamente para capturar as mudanças observadas no fluxo de dados, isto é, o mecanismo de aprendizagem é contínuo. Os modelos iniciam-se com uma amostra de dados de entrada e estimam a saída, ajustam seus parâmetros e evoluem sua estrutura a cada nova amostra apresentada. Para analisar o comportamento do erro dos modelos ao longo do tempo, utiliza-se a Soma dos Quadrados Residuais (SQR), que é calculada para todas as amostras processadas até o passo k, isto é:

$$SQR^k = \sum_{t=1}^{k}(y_t - \hat{y}_t)^2. \tag{5.2}$$

As medidas de erros são boas para se medir a precisão do modelo. Porém, elas não revelam se um modelo é estatisticamente superior a outro (Lemos et al., 2011b). Para isso, será utilizado um teste estatístico para se comparar a precisão dos modelos. O teste MGN (Diebold and Mariano, 1995) é um teste paramétrico empregado para comparar a acurácia de dois modelos e é definido por:

$$MGN = \frac{\hat{\rho}_{sd}}{\sqrt{\frac{1-\rho_{sd}^2}{n-1}}}, \tag{5.3}$$

onde $\hat{\rho}_{sd}$ é o coeficiente de correlação estimado entre $s = r_1 + r_2$, e $d = r_1 - r_2$, sendo r_1 e r_2 os resíduos dos dois modelos. A estatística do teste é dada por uma distribuição t de *Student* com $n - 1$ graus de liberdade. No teste MGN, se os modelos são igualmente precisos, a correlação entre s e d será zero.

5.2.2 Identificação de Processo Não Linear com Alta Dimensionalidade

Esta seção visa analisar o comportamento da rede eNFN em problemas com alta dimensionalidade do espaço de entrada. O processo não linear a ser identificado é descrito por:

$$y_t = \frac{\sum_{i=1}^{m} y_{t-i}}{1 + \sum_{i=1}^{m}(y_{t-i})^2} + u_{t-1}, \tag{5.4}$$

onde $u_t = sin(2\pi t/20)$, e $y_j = 0$ para $j = 1, ..., m$ e $m = 10$ (Lemos et al., 2011c). O objetivo é prever a saída corrente utilizando-se a entrada com atraso e as saídas. O modelo para este conjunto de dados é definido por:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, ..., y_{t-10}, u_{t-1}), \tag{5.5}$$

onde $\hat{y}_t$ é a saída. Para o experimento foram criadas 3300 amostras, sendo 1650 utilizadas para estimação dos parâmetros e 1650 para avaliação do desempenho.

O valor desejado e o estimado pela eNFN pode ser visto na Figura 5.1. Avaliou-se o desempenho dos modelos pelo RMSE, SQR e MGN considerando todas as amostras do conjunto de teste e definiram-se os parâmetros dos modelos como se segue: O DENFIS (limiar de distância - $dthr = 0.01$; número de regras no sistema dinâmico de inferência *fuzzy* - $mofn = 4$); eMG ($\lambda = 0.05$, $w = 10$, $\sum_{init} = 10^{-1}.I_{11}$, $\alpha = 0.01$); eNFN ($\beta = 0.01$, $\omega = 100$ e $\eta = 15$); eTS ($r = 0.04$ e $\Omega = 750$); e xTS ($\Omega = 750$).

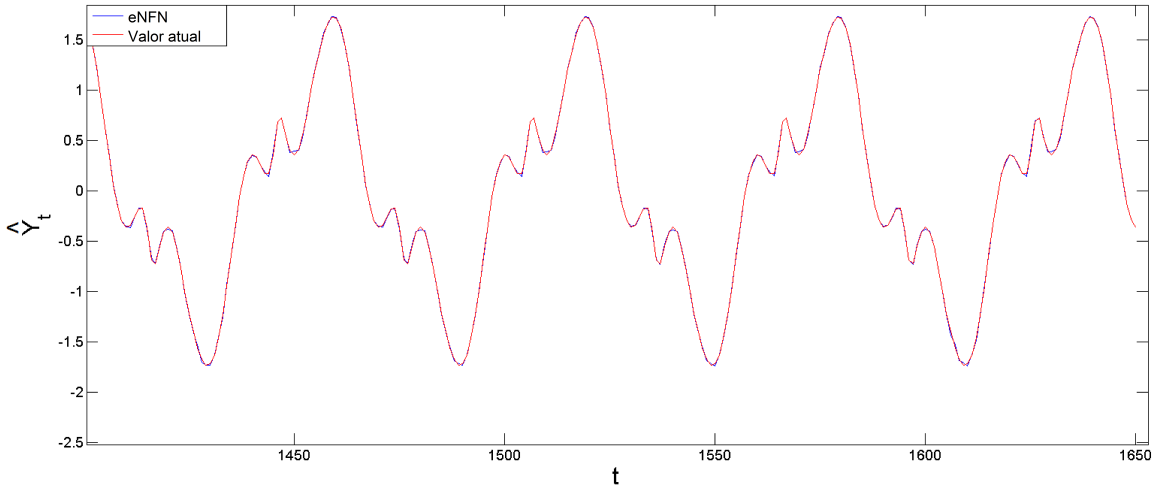

Fig. 5.1: Identificação de processo não linear com alta dimensionalidade.

Os resultados das simulações são apresentados na Tabela 5.2. Eles indicam que o melhor desempenho foi obtido pela eNFN, seguida pelo eMG. Os resultados obtidos pela eNFN e eMG superam os do DENFIS, eTS, e xTS em uma ordem de grandeza.

Tab. 5.2: Desempenho na identificação de processo não linear com alta dimensionalidade.

Modelo	N. Regras	RMSE
DENFIS	13	0.2080
eMG	10	0.1244
eNFN	107	0.1210
eTS	09	0.8303
xTS	03	0.8316

O logaritmo da soma dos quadrados residuais (Log (SQR)) dos modelos é apresentado na Figura 5.2. Os resultados visualizados na Figura 5.2 sugerem que a eNFN possui um desempenho superior ao DENFIS, eTS e xTS, e comparável ao eMG.

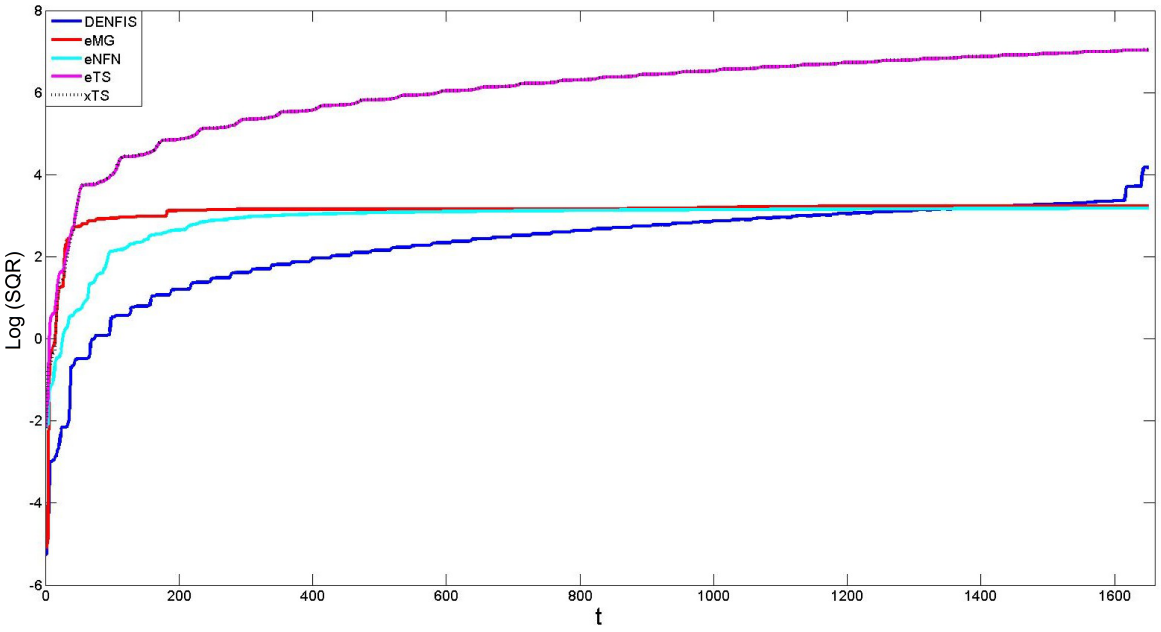

Fig. 5.2: Logaritmo da soma dos quadrados residuais na identificação de processo não linear com alta dimensionalidade.

A Tabela 5.3 mostra os resultados para as comparações entre a eNFN e os modelos alternativos utilizando o teste MGN. Dado um nível de significância de 0,05, os resultados apresentados mostram que a eNFN é estatisticamente superior ao DENFIS, eTS, xTS, e similar ao eMG.

Tab. 5.3: Avaliação da eNFN pelo teste MGN na identificação de processo não linear com alta dimensionalidade.

Modelo	MGN	p-valor
eNFN vs DENFIS	20.8169	0.0000
eNFN vs eMG	1.1195	0.1316
eNFN vs eTS	109.5962	0.0000
eNFN vs xTS	109.5019	0.0000

5.2.3 Previsão da Série Temporal de Mackey-Glass

Esta seção considera o problema de previsão da série temporal clássica de Mackey-Glass (Mackey and Glass, 1977). A série é criada pela seguinte equação diferencial com atraso:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x(t-\tau)^{10}} - 0.1x(t) \tag{5.6}$$

onde, $x(0) = 1.2$ e $\tau = 17$. O objetivo é fazer a previsão de x_{t+6} para qualquer valor de t considerando os valores de $[x_{t-18}\ \ x_{t-12}\ \ x_{t-6}\ \ x_t]$ como entrada. O conjunto de dados é composto por 3500 amostras, sendo que 3000 foram coletadas com $t \in [201, 3200]$ e 500 com $t \in [5001, 5500]$ (Kasabov and Song, 2002). Neste experimento, 1750 amostras foram utilizadas para estimar os parâmetros e 1750 para avaliar o desempenho.

A evolução da estrutura da eNFN pode ser vista na Figura 5.3. Nesta percebe-se que a estrutura final da eNFN é composta por cinco funções de pertinência para as variáveis x_{t-18}, x_{t-6}, x_t e seis para x_{t-12}. A Figura 5.4 mostra a saída desejada e a saída estimada pela eNFN.

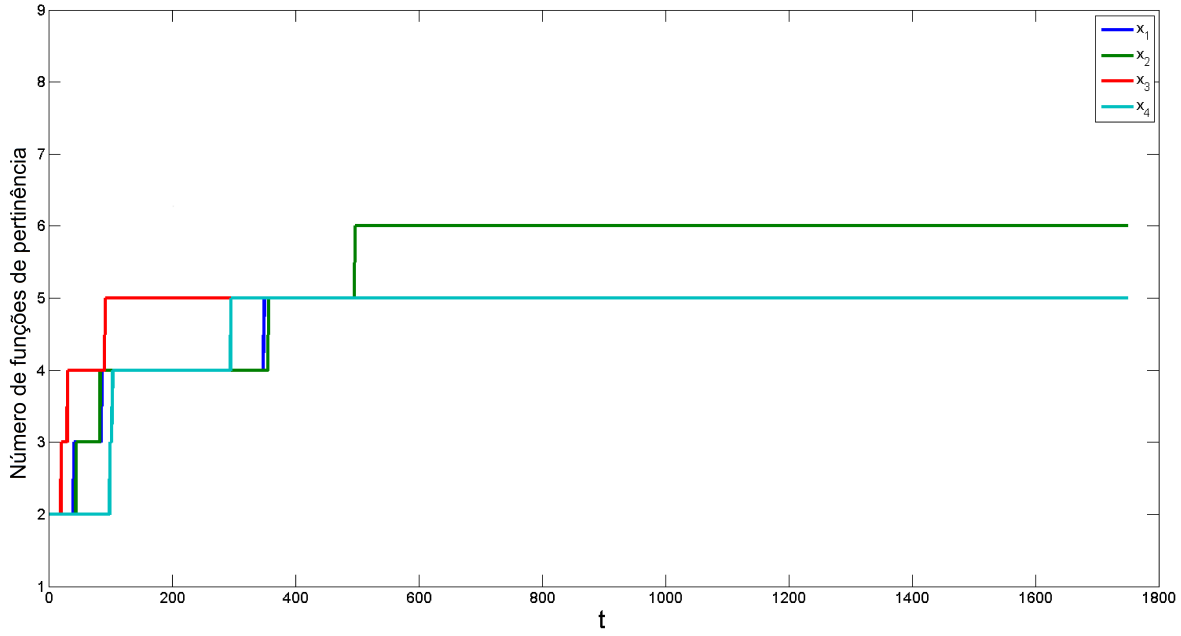

Fig. 5.3: Número de funções de pertinência da eNFN na previsão da série temporal de Mackey-Glass.

O desempenho dos modelos foi avaliado pelo RMSE, SQR e MGN calculados para todas as amostras do conjunto de teste. Os parâmetros dos modelos foram definidos como se segue: DENFIS ($dthr = 0.1$, $mofn = 3$); eMG ($\lambda = 0.05$, $w = 25$, $\sum_{init} = 10^{-2}.I_4$, $\alpha = 0.01$); eNFN ($\beta = 0.01$, $\omega = 100$ e $\eta = 05$); eTS ($r = 0.06$ e $\Omega = 500$); e xTS ($\Omega = 500$).

A Tabela 5.4 ilustra os resultados obtidos pelos modelos. Estes sugerem que o melhor desempenho foi obtido pelo DENFIS, seguido pela eNFN e eMG. Os resultados alcançados pelo DENFIS, eNFN e eMG superam os obtidos pelo eTS e xTS em uma ordem de grandeza.

O logaritmo da soma dos quadrados residuais - $Log\ (SQR)$ - dos modelos para cada iteração t é apresentado na Figura 5.5. Analisando-se a Figura 5.5, pode-se observar que a curva de erro da eNFN é menor que a do eMG, eTS e xTS, e maior que a do DENFIS.

O teste estatístico MGN é ilustrado na Tabela 5.5. Os resultados do teste MGN sugerem, com um nível de significância de 0,05, que a eNFN é estatisticamente superior ao eMG, eTS e xTS. A comparação também mostra que o desempenho do DENFIS é superior ao da eNFN.

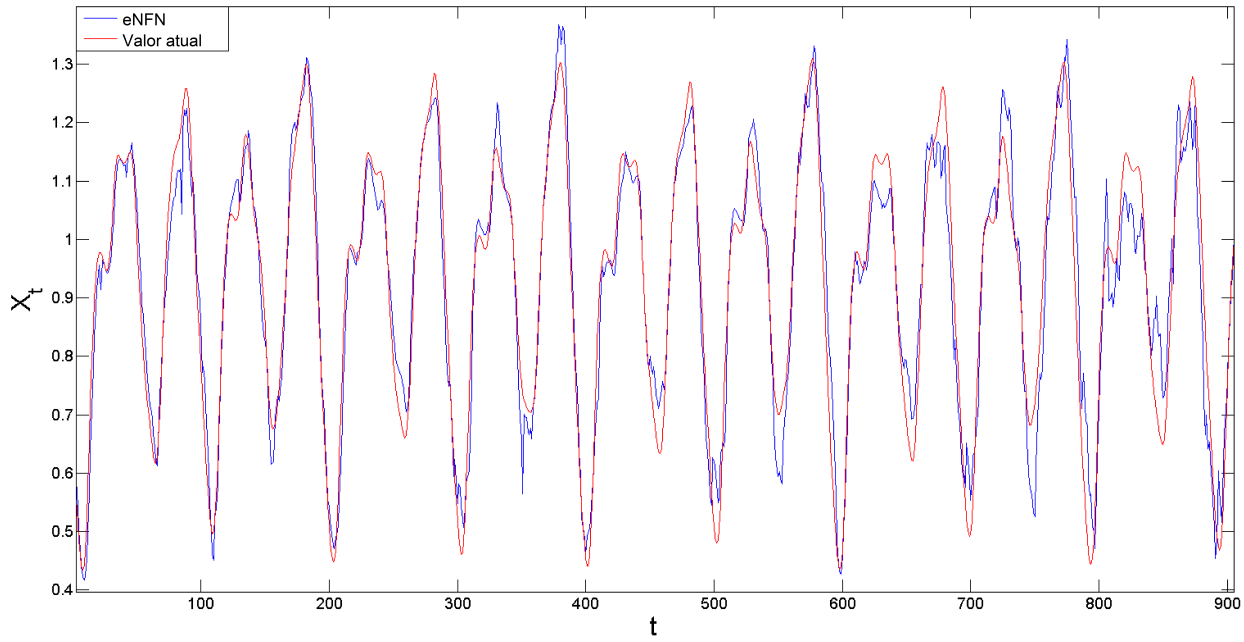

Fig. 5.4: Previsão da série temporal de Mackey-Glass.

Tab. 5.4: Desempenho na previsão da série temporal de Mackey-Glass.

Modelo	N. Regras	RMSE
DENFIS	23	0.0425
eMG	07	0.0871
eNFN	21	0.0745
eTS	05	0.3739
xTS	04	0.3750

Tab. 5.5: Avaliação da eNFN pelo teste MGN na previsão da série temporal de Mackey-Glass.

Modelo	MGN	p-valor
DENFIS vs eNFN	27.8106	0.0000
eNFN vs eMG	6.9862	0.0000
eNFN vs eTS	5.7841	0.0000
eNFN vs xTS	8.6803	0.0000

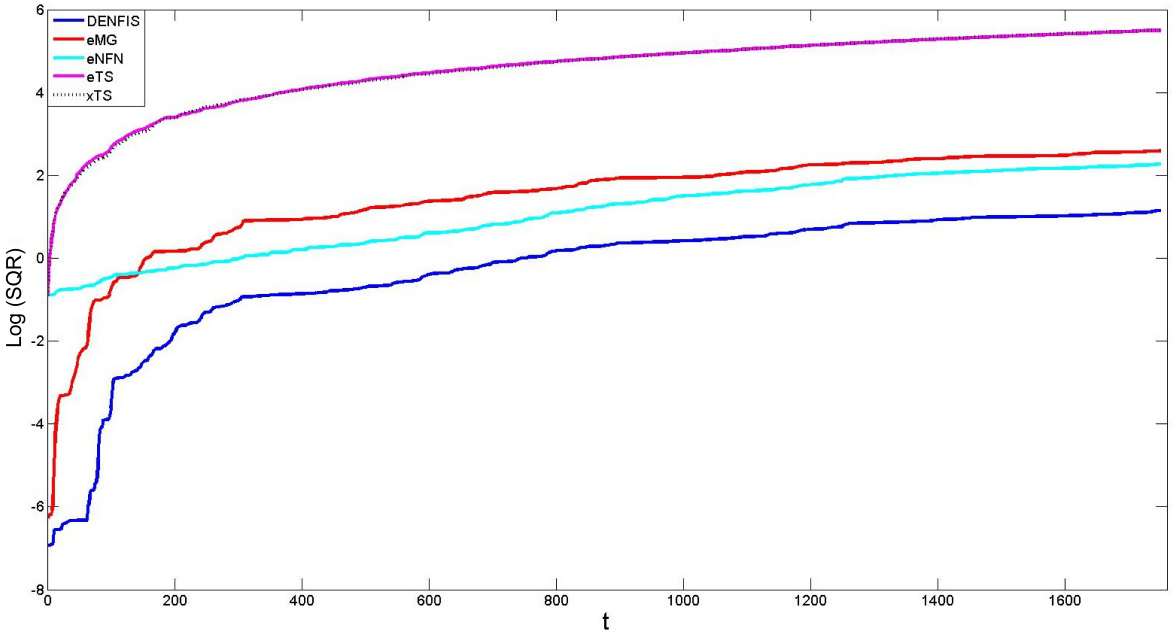

Fig. 5.5: Logaritmo da soma dos quadrados residuais na previsão da série temporal de Mackey-Glass.

5.2.4 Previsão de Vazão Semanal

Nesta seção, avaliam-se os modelos na previsão da vazão média semanal para uma usina hidrelétrica de grande porte situada no Nordeste do Brasil (Usina de Sobradinho). Os dados obtidos compreendem o período entre 1931 e 2000. Neste trabalho, como em Lemos et al. (2011b), os dados foram normalizados entre 0 e 1 para preservar a privacidade.

A análise e previsão de séries de vazões sazonais são extremamente importantes na gestão dos recursos hídricos. Uma das maiores dificuldades neste tipo de previsão é sua natureza não estacionária, devido a períodos de cheia e seca durante o ano, por exemplo. O objetivo do modelo é prever a vazão da semana seguinte em função da vazão de semanas anteriores. O modelo empregado é descrito por:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, y_{t-3}). \tag{5.7}$$

O conjunto de dados utilizado no experimento é composto por 3707 observações, destas 1854 são utilizadas para estimar os parâmetros e 1853 para avaliar o desempenho. A Figura 5.6 ilustra as funções de pertinência iniciais e as finais da eNFN para o problema de previsão de vazão. A evolução da estrutura (número de funções de pertinência) ao longo do tempo pode ser vista na Figura 5.7. Os valores desejados e os estimados pela eNFN são ilustrados na Figura 5.8.

Todas as 1853 amostras do conjunto de teste foram utilizadas para avaliar o desempenho empregando o RMSE, SQR e MGN. O DENFIS foi executado com os seguintes parâmetros: limiar de distância - $dthr = 0.07$; número de regras no sistema dinâmico de inferência *fuzzy*

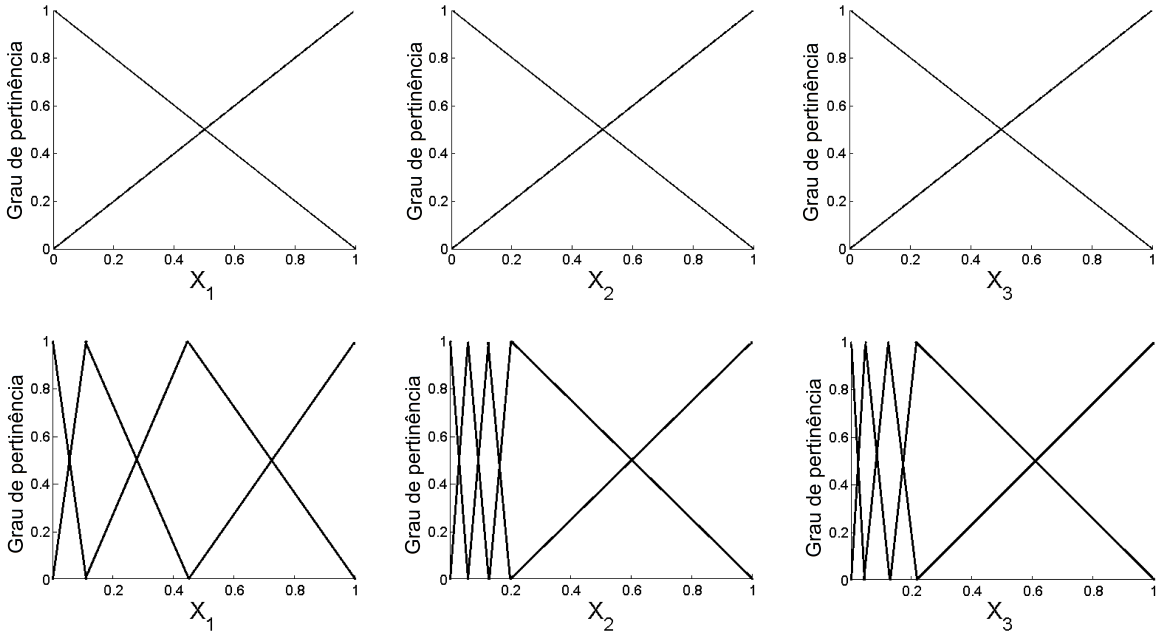

Fig. 5.6: Funções de pertinência iniciais e finais da eNFN na previsão de vazão semanal.

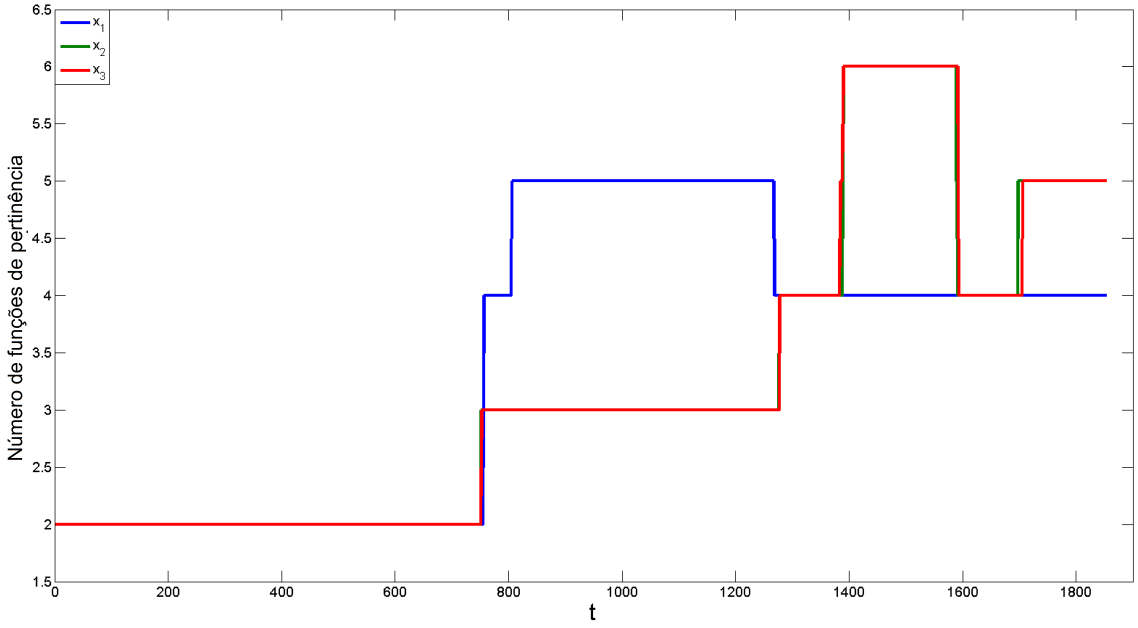

Fig. 5.7: Número de funções de pertinência da eNFN na previsão de vazão semanal.

- $mofn = 3$. Os parâmetros do eMG utilizados foram: $\lambda = 0.05$, $w = 10$, $\sum_{init} = 10^{-2}.I_3$, $\alpha = 0.01$. Os parâmetros adotados na eNFN foram: $\beta = 0.01$, $\omega = 200$ e $\eta = 15$. Os parâmetros do eTS foram $r = 0.02$ e $\Omega = 100$ e do xTS $\Omega = 250$.

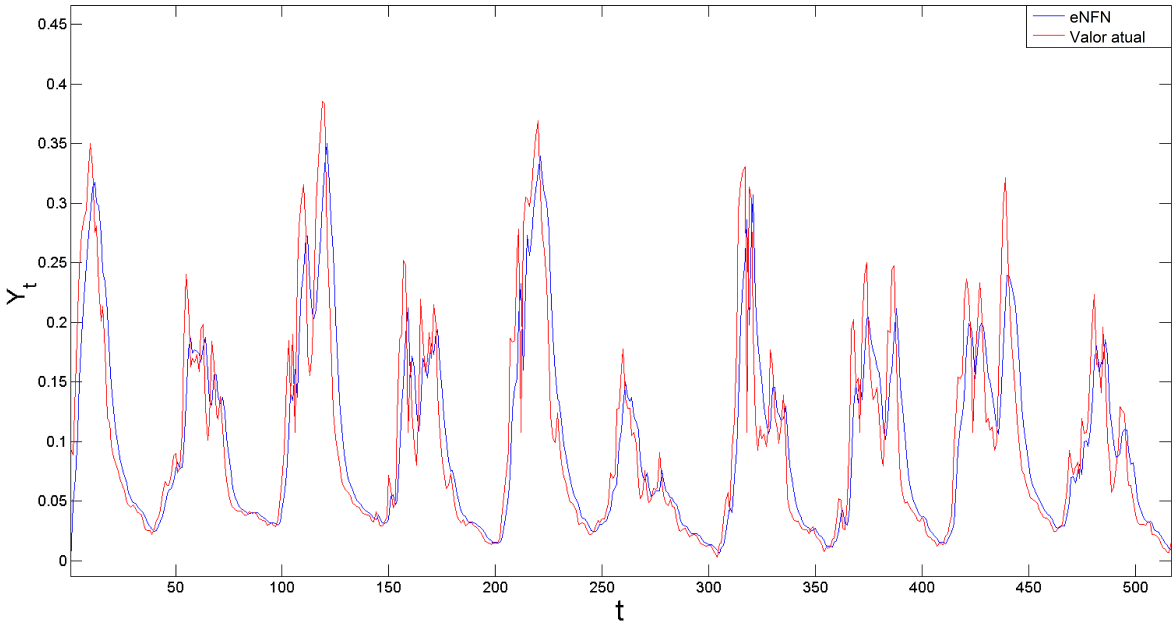

Fig. 5.8: Previsão de vazão semanal.

A Tabela 5.6 mostra os resultados obtidos. Esta tabela inclui o número de regras e o RMSE. Conforme sugere a Tabela 5.6, a eNFN obteve um desempenho comparável ao DENFIS, e inferior ao eMG, eTS e xTS.

Tab. 5.6: Desempenho na previsão de vazão semanal.

Modelo	N. Regras	RMSE
DENFIS	15	0.0562
eMG	05	0.0454
eNFN	14	0.0551
eTS	11	0.0409
xTS	05	0.0408

A Figura 5.9 apresenta a soma dos quadrados residuais (SQR) dos modelos na previsão de vazão. Por meio desta figura, pode-se perceber, pela curva do erro acumulado, que a eNFN e o DENFIS obtiveram um desempenho comparável, sendo superados pelo eMG, eTS e xTS.

Os resultados da comparação entre a eNFN e os modelos alternativos pelo teste MGN na previsão de vazão semanal são apresentados na Tabela 5.7. A tabela mostra os valores da estatística do teste MGN e o p-valor correspondente. Os resultados apresentados evidenciam

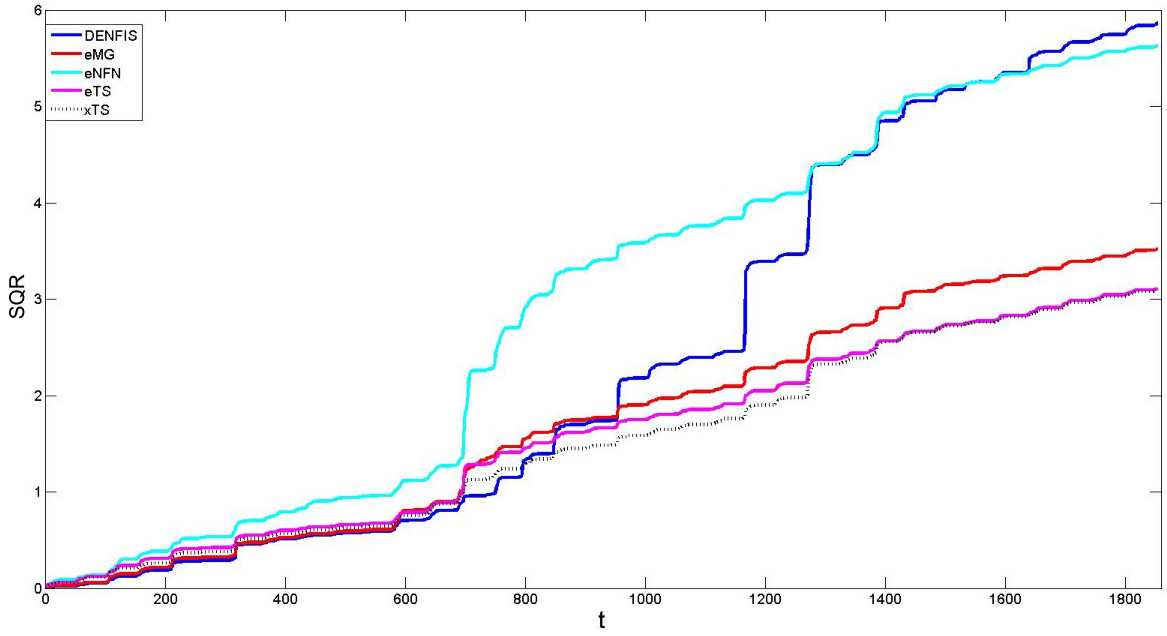

Fig. 5.9: Soma dos quadrados residuais na previsão de vazão semanal.

que a eNFN é estatisticamente similar ao DENFIS, mas inferior ao eMG, eTS e xTS, dado um nível de significância de 0,05.

Tab. 5.7: Avaliação da eNFN pelo teste MGN na previsão de vazão semanal.

Modelo	MGN	p-valor
eNFN vs DENFIS	1.0249	0.1528
eMG vs eNFN	15.0234	0.0000
eTS vs eNFN	19.2015	0.0000
xTS vs eNFN	18.0250	0.0000

5.3 Análise do Tempo de Processamento

Esta seção tem como objetivo avaliar o tempo de processamento dos modelos abordados neste trabalho. Estes foram executados para os conjuntos de dados descritos na Seção 5.2 e inicializados com os mesmos parâmetros definidos na referida seção.

A Tabela 5.8 ilustra os tempos na identificação de processo não linear com alta dimensionalidade. Os resultados apresentados na Tabela 5.9 se referem ao tempo na previsão da série de Mackey-Glass. E, por fim, a Tabela 5.10 mostra o tempo de processamento de cada modelo na previsão de vazão semanal. Os valores apresentados nessas tabelas estão expressos em milissegundos (ms) e se referem ao tempo médio e ao desvio padrão para processamento de uma

amostra do conjunto de dados. Estes foram calculados com base em 10 repetições para cada conjunto de dados.

Tab. 5.8: Tempo de processamento na identificação de processo não linear com alta dimensionalidade.

Modelo	Tempo (ms)	Desvio Padrão
DENFIS	3.3646	0.0216
eMG	3.8738	0.0189
eNFN	1.3825	0.0258
eTS	37.2541	0.0987
xTS	7.1874	0.0584

Tab. 5.9: Tempo de processamento na previsão da série temporal de Mackey-Glass.

Modelo	Tempo (ms)	Desvio Padrão
DENFIS	6.0275	0.0245
eMG	3.3908	0.0458
eNFN	0.5607	0.0365
eTS	1.8574	0.0218
xTS	1.5416	0.0167

Tab. 5.10: Tempo de processamento na previsão de vazão semanal.

Modelo	Tempo (ms)	Desvio Padrão
DENFIS	1.9643	0.0214
eMG	1.3028	0.0175
eNFN	0.3128	0.0115
eTS	1.5417	0.0125
xTS	1.4580	0.0152

Analisando-se o tempo de execução dos algoritmos apresentados nas tabelas 5.8, 5.9, e 5.10, verifica-se que o eNFN possui o menor custo computacional entre os algoritmos analisados. O tempo de processamento do eNFN é, ao menos três vezes menor do que o tempo de processamento do segundo algoritmo mais rápido, para todos os conjuntos de dados.

5.4 Relação entre Dimensão do Espaço de Entrada/Saída e Tempo de Processamento

O objetivo desta seção é analisar a escalabilidade das redes propostas com diferentes dimensões do espaço de entrada/saída e com diferentes números de regras *fuzzy*. Para isso, foram gerados 14 conjuntos de dados com 3500 amostras por meio das equações 5.4 e 5.5. Nelas variou-se o número de variáveis de entrada de $n = 2, \ldots, 15$ e para o eNFN_S (eNFN com múltiplas saídas) variou-se também o número de saídas de $s = 2, \ldots, 4$.

Nesta simulação, analisa-se o tempo de processamento das redes, alterando o número de variáveis de entrada/saída e a quantidade de regras *fuzzy*. O comportamento esperado é que o número de regras não influencie no tempo de processamento das redes, tendo em vista que, para cada variável de entrada, somente duas regras são ativadas a cada iteração. Espera-se também que o tempo de processamento das redes tenha um crescimento linear com o aumento do número de variáveis, e que o tempo de processamento da eNFN_S com s saídas seja inferior ao de s redes eNFN_1 com uma única saída.

A Figura 5.10 ilustra o comparativo[5] entre o número de regras e o tempo de processamento das redes eNFN_1 com uma única saída para diferentes dimensões do espaço de entrada. Para este comparativo, modificaram-se os parâmetros da rede com o objetivo de variar o número de regras. Executou-se a eNFN_{1A} com $\eta = 10$ e a eNFN_{1B} com $\eta = 20$, pode-se ver o número de regras gerado pelas redes eNFN_{1A} e eNFN_{1B} na Tabela 5.11.

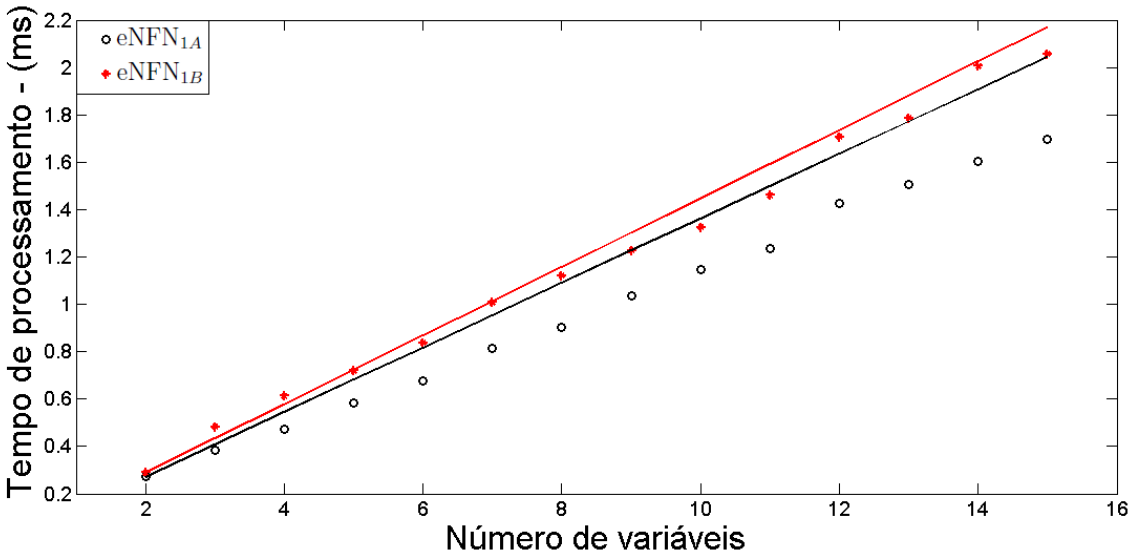

Fig. 5.10: Escalabilidade das redes eNFN_1 com uma única saída.

Analisando-se os resultados apresentados na Figura 5.10, percebe-se que, conforme esperado, as redes eNFN_1 possuem um crescimento do tempo de execução linear com relação ao número de

[5]Nas figuras 5.10 e 5.11 as retas indicam o crescimento linear do tempo de processamento baseado no tempo de processamento das redes com duas variáveis de entrada.

variáveis de entrada e o número de regras não é um fator significante no tempo de processamento dessas redes. Isto se deve ao fato das redes eNFN ativarem somente $2n$ regras para cada amostra.

O comparativo entre o tempo de processamento das eNFN_S com múltiplas saídas e de s redes eNFN_1 com uma única saída é ilustrado na Figura 5.11. Esta simulação foi realizada com três redes eNFN_S com múltiplas saídas, eNFN_{S2} com duas saídas, eNFN_{S3} com três, e eNFN_{S4} com quatro saídas. O tempo de processamento da rede eNFN_{1A} (Figura 5.10) foi multiplicado por dois, três e quatro, para simular os referidos números de saídas, conforme apresentado na Figura 5.11. O número de regras gerado para cada rede pode ser encontrado na Tabela 5.11.

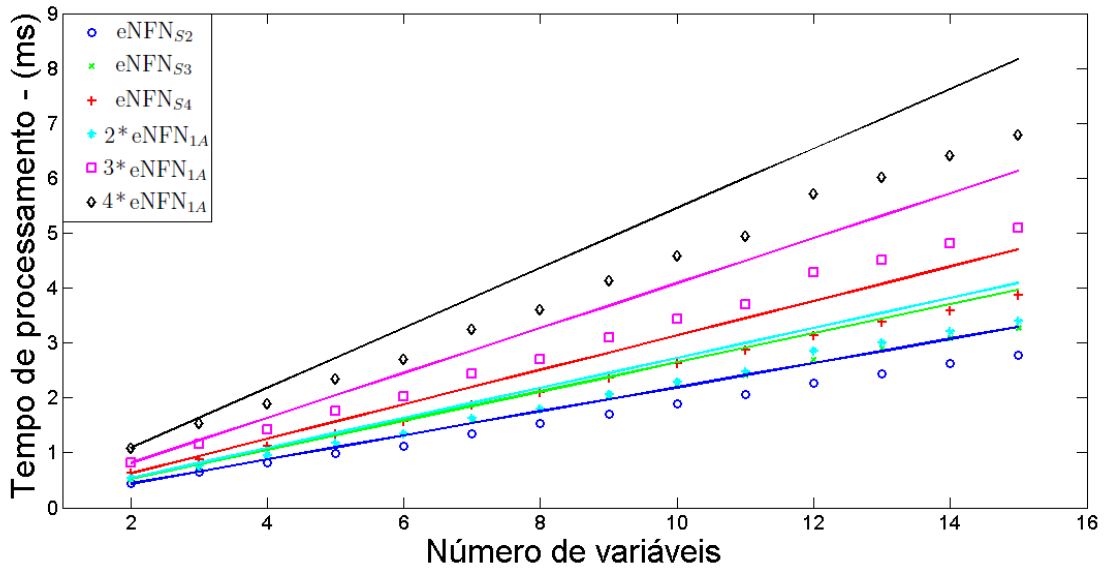

Fig. 5.11: Tempo de processamento das redes eNFN_S com múltiplas saídas e de s redes eNFN_1 com uma única saída.

Conforme esperado, pode-se perceber pelo gráfico apresentado na Figura 5.11 que as redes eNFN_S com múltiplas saídas possuem um tempo de processamento menor que o de s redes eNFN_1 com uma única saída. Isso se dá, principalmente, pelo fato das redes com múltiplas saídas efetuarem o cálculo do grau de ativação das funções de pertinência e de realizarem o procedimento de evolução da estrutura apenas uma vez para cada amostra apresentada à rede. Nota-se também na Figura 5.11 que o acréscimo no tempo de processamento das redes eNFN_S em relação ao aumento no número de saídas é relativamente pequeno.

Tab. 5.11: Número de regras das redes eNFN por dimensão do espaço de entrada.

N. Var.	eNFN_{1A}	eNFN_{1B}	eNFN_{S2}	eNFN_{S3}	eNFN_{S4}
02	16	24	16	16	16
03	29	46	29	29	29
04	41	65	41	41	41
05	51	75	51	51	51
06	58	95	58	58	58
07	66	105	66	66	66
08	78	119	78	78	78
09	86	139	86	86	86
10	98	138	98	98	98
11	110	169	110	110	110
12	118	183	118	118	118
13	132	209	132	132	132
14	136	212	136	136	136
15	147	218	147	147	147

5.5 Resultados Experimentais em Tempo Real

Esta seção descreve os experimentos computacionais realizados em tempo real para avaliar as redes eNFN. As redes foram avaliadas em problemas de estimação de variáveis de estado utilizando o módulo de levitação magnética (*Magnetic Levitation System* - MagLev) e o sistema MIMO de duplo rotor (*Twin Rotor MIMO System* - TRMS), ambos produzidos pela Feedback Instruments Limited. Estes sistemas simulam problemas e desafios encontrados em sistemas reais.

A metodologia empregada nos experimentos em tempo real é apresentada na Seção 5.5.1. O sistema de levitação magnética é o assunto da Seção 5.5.2. Inicialmente, apresenta-se o sistema de levitação magnética e sua modelagem matemática. Depois, são ilustrados os experimentos e seus respectivos resultados. A Seção 5.5.3 começa com a introdução ao TRMS e em seguida apresenta seu modelo matemático. Depois, são descritos os experimentos realizados e os resultados obtidos utilizando a rede eNFN com múltiplas saídas. Nos experimentos em tempo real com alta taxa de amostragem (dinâmica rápida), como é o caso do MagLev e do TRMS, é importante, além da acurácia, baixo custo computacional.

5.5.1 Metodologia

Nos sistemas do Maglev e do TRMS, o software de aquisição de dados e controle é executado em um microcomputador com sistema operacional Windows e utiliza uma placa de aquisição de dados Advantech. As unidades de controle são implementadas em ambientes Matlab e Simulink. O sistema operacional Windows não foi desenvolvido para aplicações em tempo real. Por isso utiliza-se a *Real-Time Workshop* do Matlab (MathWorks, 2009) para simular o processamento

em tempo real. A *Real-Time Workshop* não executa um processamento determinístico com taxa de amostragem fixa, mas possibilita ao sistema a mais alta prioridade entre todas as tarefas executadas pelo sistema operacional. Quando uma tarefa é executada pelo núcleo de processamento em tempo real, o mecanismo de alocação de código do Windows é bloqueado. As tarefas de prioridade mais baixa são executadas nos intervalos de tempo restante. Caso não seja possível obter características de tempo real para uma determinada taxa de amostragem, o núcleo de processamento em tempo real emite um alerta.

Na estimação em tempo real p passos à frente, a rede eNFN é executada em duas etapas. A primeira etapa visa ao ajuste dos parâmetros e a evolução da estrutura. Nesta etapa, a eNFN tem como entrada a posição desejada em $(t - \ddot{a})$[6] e as medidas da posição em $(t - \ddot{a})$. O valor estimado da posição é comparado com a posição medida atual (y_t) e a diferença (erro) é utilizada para atualizar os parâmetros e adaptar a estrutura da rede. Conclui-se esta etapa fixando parâmetros e estrutura da eNFN. Na segunda etapa, estima-se a posição p passos à frente. Nesta etapa, o modelo tem como entrada a posição desejada em $(t - \ddot{a} + p)$ e as medidas da posição em $(t - \ddot{a} + p)$. Utilizam-se estas entradas e os parâmetros e estrutura fixados na primeira etapa para estimar a posição p passos à frente, isto é, $(\hat{y}_{t+p})$. A Figura 5.12 mostra a estrutura da eNFN para a estimação em tempo real. No experimento com o MagLev, empregou-se a rede eNFN com uma única saída. No experimento com o TRMS, uma eNFN com duas saídas foi utilizada.

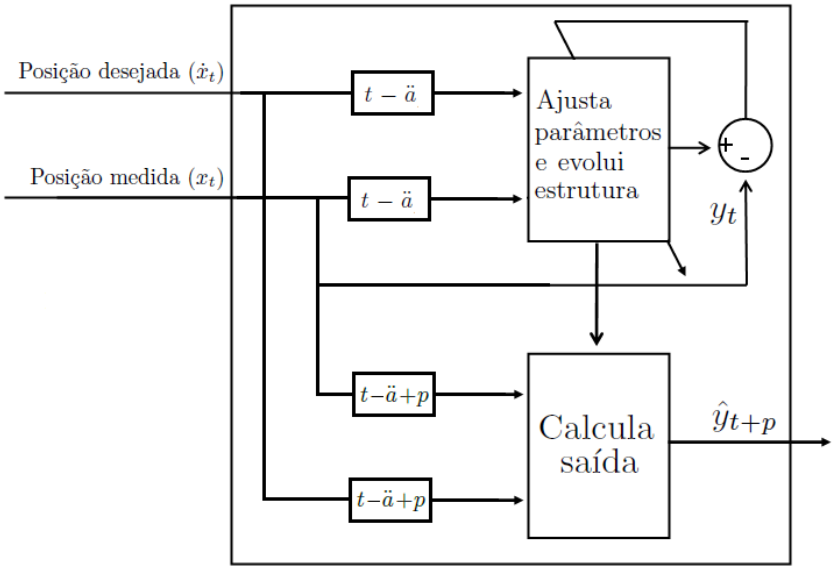

Fig. 5.12: Estimador de estado com eNFN.

Para avaliar o desempenho da eNFN na estimação em tempo real p passos à frente será utilizado o erro quadrático E_t (3.7).

[6] t é o passo atual e $\ddot{a}$ são os atrasos

5.5.2 Sistema de Levitação Magnética

A levitação magnética (MagLev) é uma tecnologia utilizada para eliminar o contato mecânico entre as partes móveis e fixas de um sistema para impedir problemas com atrito, desgaste mecânico, fricção, ruído, e geração de calor e pó metálico. Essa tecnologia é empregada com sucesso em diversas aplicações de engenharia incluindo mancais magnéticos, sistemas de isolamento de vibração, rolamento sem atrito, trens de alta velocidade, modelos de levitação em túnel de vento, dentre outras.

O sistema de levitação magnética considerado neste trabalho consiste no módulo do sistema MagLev produzido pela Feedback Instruments Limited Feedback (2006a), o qual se destina ao estudo do fenômeno da levitação de uma esfera de aço decorrente da força magnética gerada por uma corrente elétrica circulando em uma bobina. O MagLev é um processo não linear instável em malha aberta. Sob o ponto de vista de controle, o objetivo é manter a esfera em uma posição especificada. O módulo MagLev é um processo SISO (*Single Input / Single Output*), isto é, possui uma única entrada, a tensão aplicada à bobina, e uma única saída, a posição vertical da esfera.

O módulo, ilustrado na Figura 5.13, possui uma bobina para gerar o campo magnético e um sensor óptico infravermelho para medir a posição vertical da esfera. O controle da posição da esfera é feito somente dentro da faixa de trabalho desse sensor.

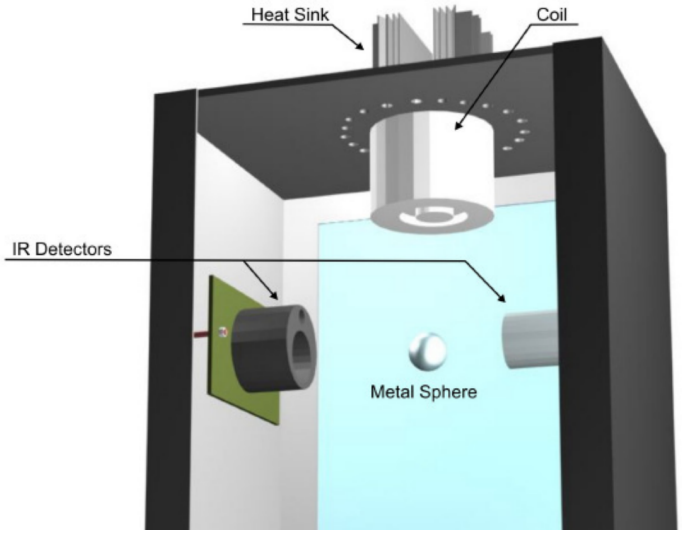

Fig. 5.13: Módulo MagLev - Fonte: Feedback, 2006a.

5.5.2.1 Modelo Matemático do Sistema MagLev

O modelo MagLev, conforme o módulo descrito, possui uma variável de entrada, a tensão (u), e uma variável de estado que é a própria saída, a posição vertical da esfera (x). O modelo não linear mais simples do sistema de levitação magnética relativo à posição da esfera (x) e a corrente da bobina (i) é dado por:

$$\ddot{x} = g - \frac{k}{m}\frac{i^2}{x^2}, \tag{5.8}$$

onde k é uma constante que depende dos parâmetros da bobina, g é a constante gravitacional e m é a massa da esfera. A Equação 5.8 mostra que o modelo tem dinâmica tal que o aumento da corrente leva a redução da aceleração. Para apresentar o modelo fenomenológico completo deve ser introduzida uma relação entre a tensão u (variável de controle) e a corrente da bobina i. O MagLev está equipado com um circuito de controle interno que fornece uma corrente proporcional à tensão de controle u:

$$i = k_1 u. \tag{5.9}$$

As equações 5.8 e 5.9 constituem o modelo não linear implementado no Simulink. Os limites para a variável de controle foram definidos com a amplitude entre $-5V$ e $5V$.

5.5.2.2 Experimento - Estimação de Posição

A rede eNFN com uma única saída foi usada para estimar em tempo real a posição de uma esfera em um sistema de levitação magnética. A eNFN tem como objetivo estimar a posição da esfera p passos à frente. A posição da esfera é uma das variáveis de estado cujo valor depende da tensão aplicada na bobina que gera o campo magnético. O controle da posição da esfera é feito por um controlador PID cuja entrada é a diferença entre a posição atual (medida) e a posição desejada, e a saída é a tensão. A Figura 5.14 ilustra o diagrama simplificado do estimador de estado para o sistema de levitação magnética.

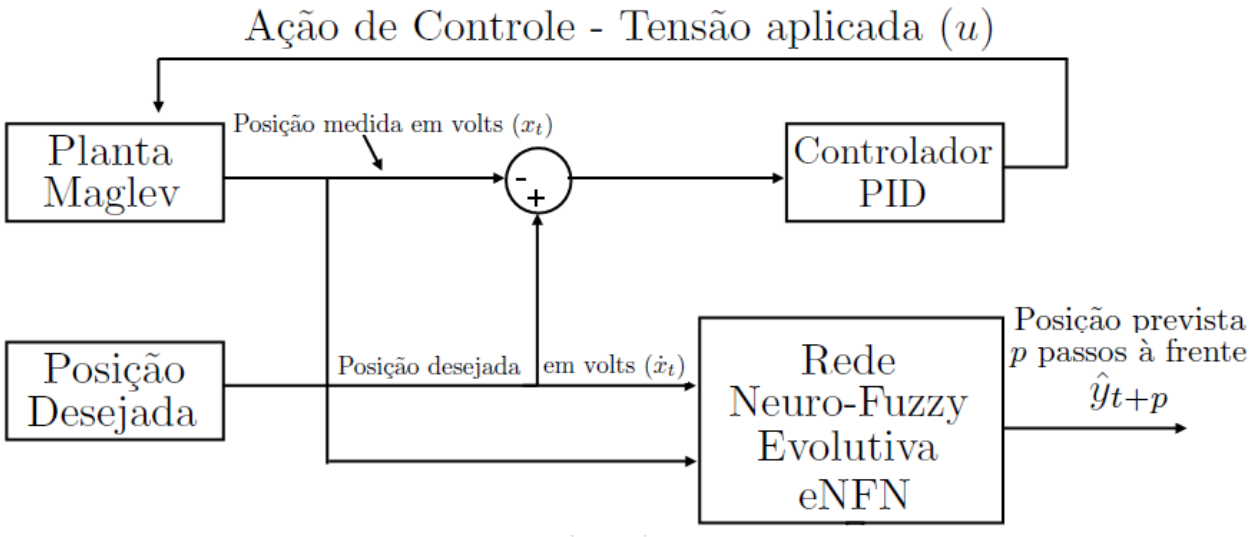

Fig. 5.14: eNFN como estimador de estado para o sistema de levitação magnética

Neste experimento, para ajuste dos parâmetros e evolução da estrutura, a eNFN tem como entrada a posição desejada em $(t-5)$ e as medidas da posição em $(t-5)$, $(t-4)$ e $(t-3)$.

Para estimação da posição da esfera p ($p = 1, ..., 3$) passos à frente, ou seja, no instante $(t + p)$, a rede eNFN tem como entrada a posição desejada em $(t - 5 + p)$ e as medidas da posição em $(t - 5 + p)$, $(t - 4 + p)$ e $(t - 3 + p)$.

O experimento foi executado durante 60 segundos com uma taxa de amostragem de 10^{-4} segundos (1 milissegundo), resultando em um total de 60.000 amostras. A posição desejada foi definida inicialmente por uma função seno com amplitude e frequência iguais a 0,5 Hertz. Aos 17 segundos, alterou-se a posição desejada para uma função quadrada com amplitude de 0,4 Hertz e frequência 0,5 Hertz. Aos 31 segundos de simulação, a posição desejada passou a ser uma função degrau (com amplitude entre -1 e -2). Aos 41 segundos, voltou-se a ter como posição desejada a função seno e, por fim, aos 51 segundos, aplicou-se uma função quadrada. O objetivo da alternância entre os valores da posição desejada é analisar o comportamento do modelo evolutivo em diferentes condições operacionais. Espera-se que a eNFN consiga computar a saída para atualização dos parâmetros, evoluir sua estrutura e estimar a posição p passos à frente em tempo inferior ao da taxa de amostragem e com precisão.

Para avaliar o desempenho da eNFN na estimação com $p = 1$, $p = 2$ e $p = 3$ passos à frente utilizou-se o erro quadrático E_t (3.7). A Figura 5.15 mostra: (A) a posição medida da esfera; (B) a evolução da estrutura da eNFN; (C), (D) e (E) o erro quadrático instantâneo E_t para a estimação da posição com um, dois, e três passos à frente, respectivamente.

Analisando os resultados apresentados na Figura 5.15 dois momentos podem ser destacados:

1. *Posição desejada é dada pela função seno ou degrau*: a eNFN mantém uma taxa de erro baixa e a estrutura do modelo se mantém estável com o número mínimo de regras.

2. *Posição desejada é dada pela função quadrada*: ocorre um aumento significativo na taxa de erro e logo após o início desse aumento a eNFN evolui sua estrutura para reduzir o erro.

Nota-se na Figura 5.15 que a eNFN adapta/evolui sua estrutura em função dos dados de entrada e da taxa de erro, buscando manter uma estrutura compacta e uma taxa de erro baixa. Nas figuras 5.15c, 5.15d e 5.15e, que mostram os valores da medida de erro E_t, observa-se que a eNFN obteve uma melhor acurácia na estimação com um passo à frente, seguido pela estimação com dois e três passos à frente. A Figura 5.16 mostra a posição medida e a estimada (um passo à frente) pela eNFN durante o experimento. A Figura 5.16a ilustra a posição desejada e a estimada para a função quadrada, 5.16b para a função degrau, e 5.16c para a função seno.

Os experimentos mostram que a eNFN consegue fazer a estimativa em tempo hábil, isto é, abaixo do período da taxa de amostragem, obtendo ao mesmo tempo uma boa acurácia, principalmente para a estimação com um passo à frente. Os experimentos também mostram que a estrutura da eNFN evolui de acordo com o fluxo dos dados de entrada para obter o modelo mais adequado e preciso para a condição operacional corrente, mantendo uma estrutura compacta e parcimoniosa.

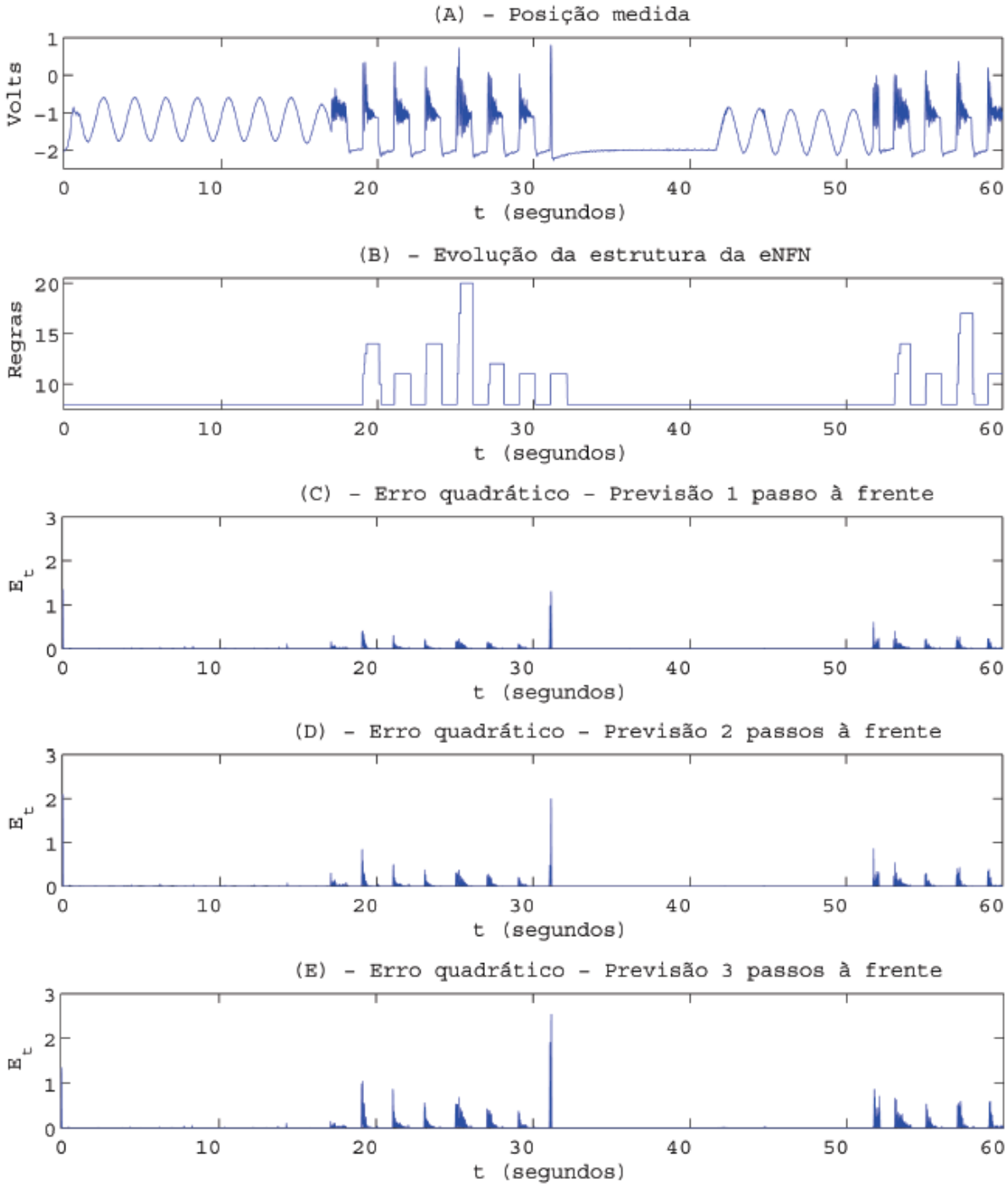

Fig. 5.15: (A) Posição medida da esfera, (B) Evolução da estrutura da eNFN, (C) Erro quadrático na estimação com um passo à frente, (D) Erro quadrático na estimação com dois passos à frente, (E) Erro quadrático na estimação com três passos à frente.

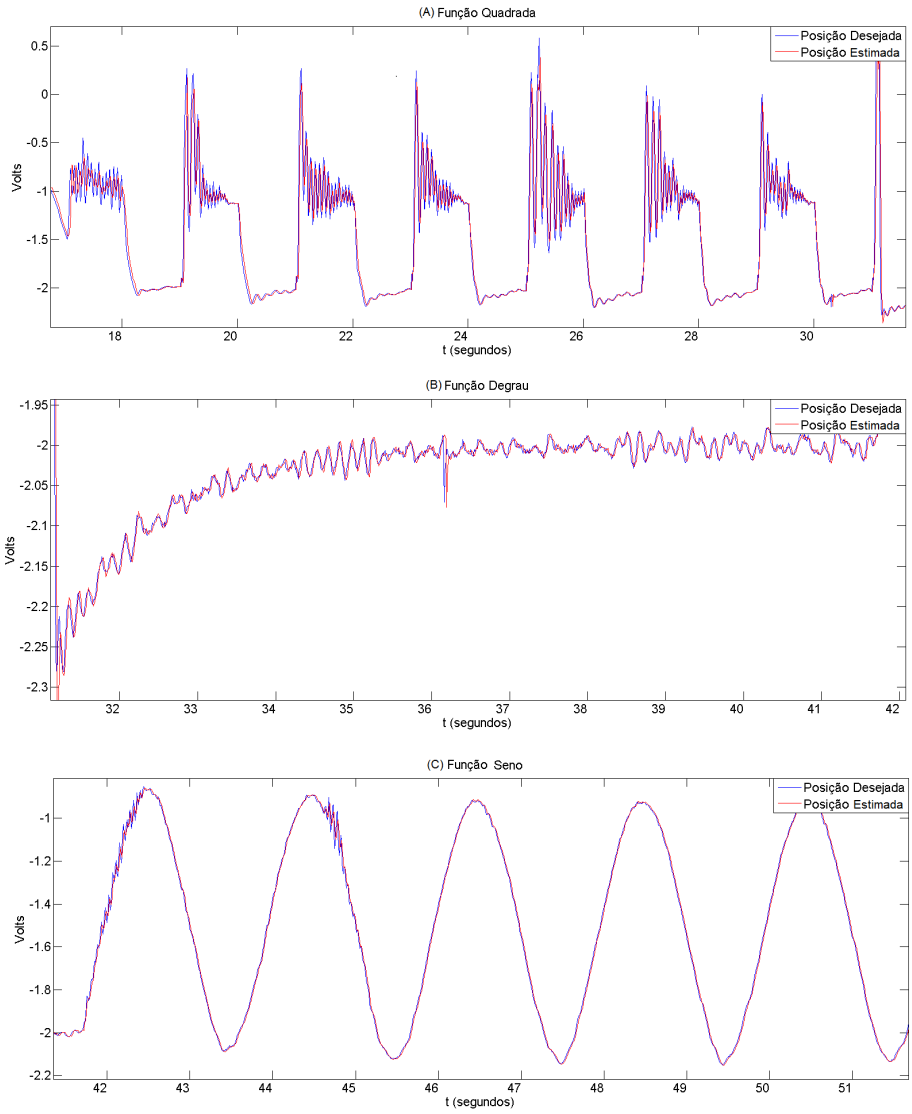

Fig. 5.16: Posição medida e posição estimada (um passo à frente) pela eNFN.

5.5.3 Sistema MIMO de Duplo Rotor

O Sistema MIMO de Duplo Rotor (*Twin Rotor MIMO System* - TRMS) é um módulo para experimentos desenvolvido pela Feedback Instruments Limited (Feedback, 2006b). O TRMS (Figura 5.17) é constituído por uma haste fixada a um conjunto torre/base, e possui em cada extremidade da haste um rotor acionado por um motor de corrente contínua com velocidade variável que possibilita o movimento vertical (arfagem - *pitch*) e horizontal (guinada - *yaw*). O rotor principal atua no ângulo de elevação (eixo de arfagem - *pitch axis*) e o rotor de calda é usado para movimentar para direita ou esquerda, atuando no ângulo de rotação (eixo de guinada - *yaw axis*).

Em certos aspectos o TRMS se assemelha a um helicóptero. Por exemplo, assim como um helicóptero, ele possui um forte acoplamento cruzado entre os dois rotores, uma dinâmica complexa e altamente não linear. O TRMS possui dois dos três movimentos de um helicóptero, o ângulo de arfagem e o de guinada. Além dos dois movimentos citados, o helicóptero real também possui o ângulo de rolagem. No módulo do duplo rotor a posição é controlada por meio da variação da velocidade do rotor. No entanto, no helicóptero real a força aerodinâmica é controlada pela modificação do ângulo das lâminas do rotor, e a velocidade do rotor é mais ou menos constante.

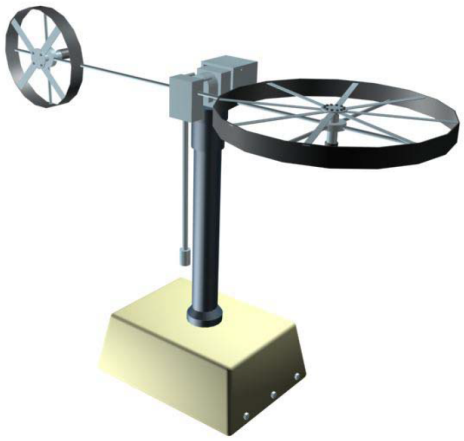

Fig. 5.17: Módulo TRMS - Fonte: Feedback, 2006b.

O sistema do duplo rotor é um processo com múltiplas entradas e múltiplas saídas (MIMO - *Multiple Input / Multiple Output*), sendo duas variáveis de entrada, a tensão aplicada ao rotor principal u_1 e a voltagem aplicada ao rotor de calda u_2, e no mínimo duas variáveis de saída, o ângulo de arfagem (*pitch angle*) Ψ e o ângulo de guinada (*yaw angle*) φ. Outras saídas também podem ser consideradas, como por exemplo, as respectivas velocidades angulares.

A obtenção de um modelo não linear do TRMS a partir de dados coletados é uma tarefa difícil, principalmente devido às suas características como dinâmica complexa, alta não linearidade, acoplamento cruzado entre os dois eixos, fricção, saturação, ruído e à inacessibilidade de alguns de seus estados e saídas. Para tais sistemas, as abordagens de modelagem adaptativa são ferramentas apropriadas para lidar com as incertezas da planta (Toha and Tokhi, 2009).

A Figura 5.18 mostra o diagrama simplificado do TRMS. Como pode ser visto na Figura 5.18, u_1 controla o ângulo de arfagem e u_2 atua no ângulo de guinada. Esta figura também ilustra a dinâmica do acoplamento cruzado entre os movimentos dos dois eixos, com u_1 afetando φ e u_2 perturbando Ψ.

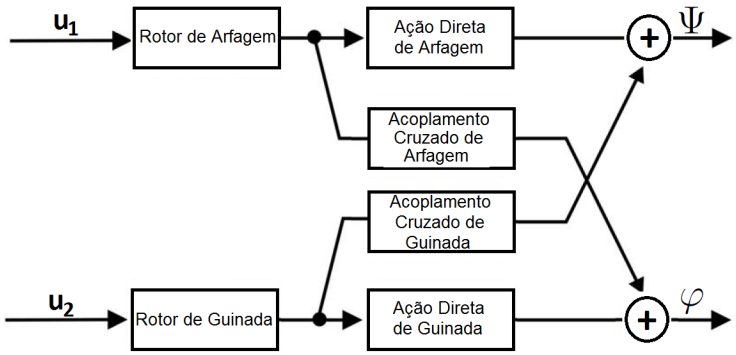

Fig. 5.18: Diagrama simplificado do TRMS.

5.5.3.1 Modelo Matemático do TRMS

A seguir é apresentada a abordagem fenomenológica para o TRMS, isto é, as equações que representam a física do sistema. Para o movimento vertical (arfagem), tem-se:

$$I_1\ddot{\Psi} = M_1 - (M_{FG} + M_{B\Psi} + M_{R\varphi}), \tag{5.10}$$

onde M_1 é a característica estática não linear. Esta é descrita por uma relação estática quadrática entre o conjugado propulsor e a velocidade angular do rotor, e é calculada por

$$M_1 = a_1\tau_1^2 + b_1\tau_1, \tag{5.11}$$

M_{FG} representa o conjugado gravitacional, porque o centro de massa da estrutura do giroscópio não coincide com o eixo de arfagem, e é encontrado como se segue

$$M_{FG} = M_g sin\Psi, \tag{5.12}$$

$M_{B\Psi}$ é o conjugado de atrito seco e viscoso, e é obtido por

$$M_{B\Psi} = B_{1\Psi}\dot{\Psi} + B_{2\Psi}sign(\dot{\Psi}), \tag{5.13}$$

$M_{R\varphi}$ representa o conjugado cruzado de guinada, isto é, o conjugado do giroscópio, e é dado por

$$M_{R\varphi} = K_{gy}M_1\dot{\varphi}cos\Psi. \tag{5.14}$$

O conjugado do rotor para o movimento vertical é descrito por

$$\tau_1 = \frac{k_1}{T_{11}s + T_{10}} u_1. \tag{5.15}$$

Similarmente, para o movimento horizontal (guinada - *yaw*) tem-se:

$$I_2\ddot{\varphi} = M_2 - (M_{B\varphi} + M_{R\Psi}), \tag{5.16}$$

onde M_2 é uma característica estática não linear descrita por

$$M_2 = a_2\tau_2^2 + b_2\tau_2, \tag{5.17}$$

$M_{B\varphi}$ representa o conjugado de atrito obtido por

$$M_{B\varphi} = B_{1\varphi}\dot{\varphi} + B_{2\varphi}sign(\dot{\varphi}), \tag{5.18}$$

$M_{R\Psi}$ é o conjugado cruzado devido ao movimento de arfagem que é calculado por

$$M_{R\Psi} = \frac{k_c(T_0 s + 1)}{T_p s + 1}\tau_1. \tag{5.19}$$

Finalmente, o conjugado do rotor para o movimento horizontal é

$$\tau_2 = \frac{k_2}{T_{21}s + T_{20}} u_2. \tag{5.20}$$

Uma breve descrição dos parâmetros do TRMS é dada a seguir, e um exemplo de valores aproximados para estes parâmetros pode ser encontrado em Feedback (2006b): I_1 conjugado de inércia do rotor vertical; I_2 conjugado de inércia do rotor horizontal; a_1, a_2, b_1, b_2 parâmetros da característica estática; M_g força de gravitacional; $B_{1\Psi}$, $B_{2\Psi}$, $B_{1\varphi}$, $B_{2\varphi}$ parâmetros do conjugado de atrito; K_{gy} parâmetros do conjugado cruzado; k_1 ganho do motor 1; k_2 ganho do motor 2; T_{11}, T_{10} parâmetro do denominador do motor 1; T_{21}, T_{20} parâmetro do denominador do motor 2; T_p, T_0 parâmetros do conjugado cruzado; k_c ganho do conjugado cruzado.

5.5.3.2 Experimento - Estimação de Variáveis de Estado

Esta seção apresenta experimentos computacionais realizados em tempo real utilizando a rede eNFN com múltiplas saídas na estimação de variáveis de estado do TRMS. No TRMS o controle dos ângulos de arfagem e guinada é realizado por dois controladores PID, um para o movimento de arfagem e o outro para o de guinada. A entrada dos controladores é a diferença entre a posição medida e a posição desejada, e a saída é a tensão aplicada aos rotores de arfagem e guinada. A Figura 5.19 ilustra o diagrama simplificado do estimador de estado para o sistema MIMO de duplo rotor.

Dois experimentos foram realizados: 1) modelagem do TRMS tendo como saídas os ângulos de arfagem e guinada, 2) modelagem da saída do controlador, isto é, a voltagem aplicada aos rotores de arfagem e guinada. Em ambos experimentos, o tempo de simulação é de 100 segundos com a taxa de amostragem de 10^{-4} segundos (1 milissegundo), resultando num total de 100.000 amostras. O ângulo desejado de arfagem é obtido pela soma de três funções seno com amplitude 0,8, 0,3 e 0,3, e frequência 0,1, 0,05 e 0,01. Acrescenta-se a esse sinal uma amplitude constante

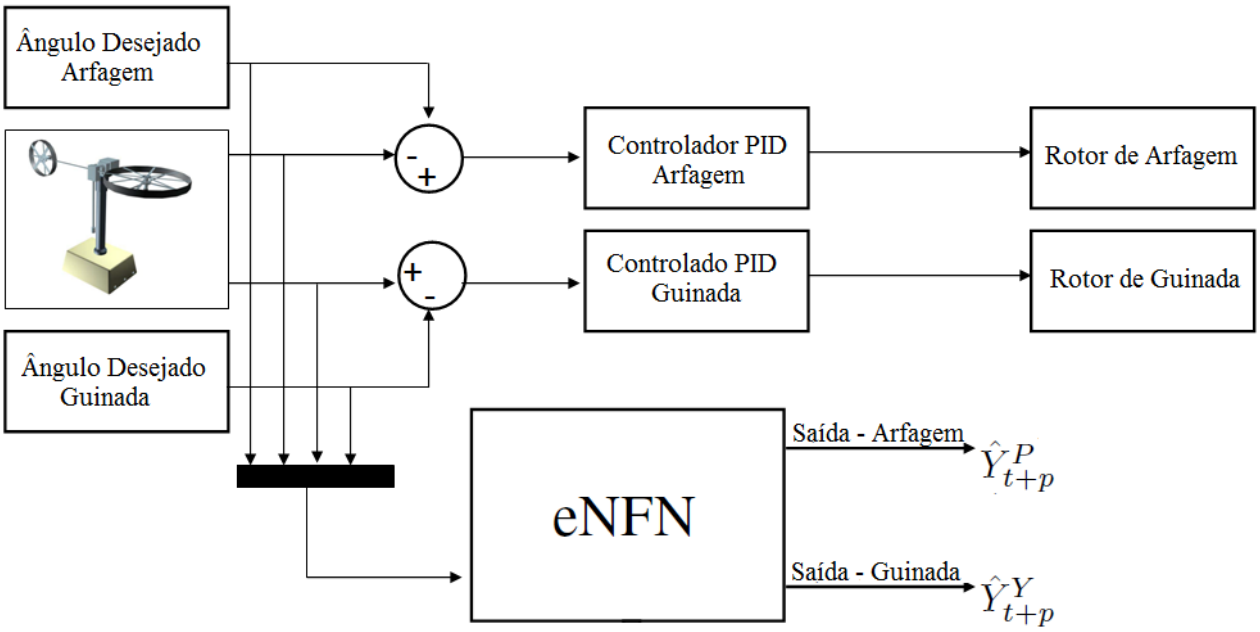

Fig. 5.19: eNFN como estimador de estado para o sistema MIMO de duplo rotor

de 0,4. O ângulo desejado de guinada é dado pela soma de três funções seno com amplitude 0,1 e frequência 0,1, 0,05, 0,02.

Nos experimentos de estimação em tempo real do TRMS, a rede eNFN é executada em duas etapas, conforme descrito na Seção 5.5.1. Na etapa de ajuste dos parâmetros e estrutura, a rede recebe como entrada os valores desejados em t e os medidos em *t-1* para os ângulos de arfagem e guinada. Para a segunda etapa que estima a saída p passos à frente (t+p), a rede tem como entrada os ângulos desejados em *t+p*, os medidos em *t-1+p* e os parâmetros e a estrutura definidos na primeira etapa. O desempenho da eNFN na estimação das variáveis de estado do TRMS é avaliado pelo erro quadrático E_t.

5.5.3.2.1 Estimação de Posição

Este experimento avalia o comportamento da rede eNFN na estimação um passo à frente dos ângulos de arfagem e guinada - duas variáveis de estado do TRMS. A Figura 5.20 mostra: (A) o ângulo medido e o estimado para o movimento de arfagem; (B) o ângulo medido e o estimado para o movimento de guinada; (C) a evolução da estrutura da eNFN; (D) e (E) o erro quadrático instantâneo E_t para a estimação do ângulo de arfagem e para o ângulo de guinada, respectivamente. Assim como no primeiro experimento (Seção 5.5.2), a eNFN obteve uma boa acurácia e a estrutura da rede evoluiu de acordo com os dados de entrada e com o erro de modelagem.

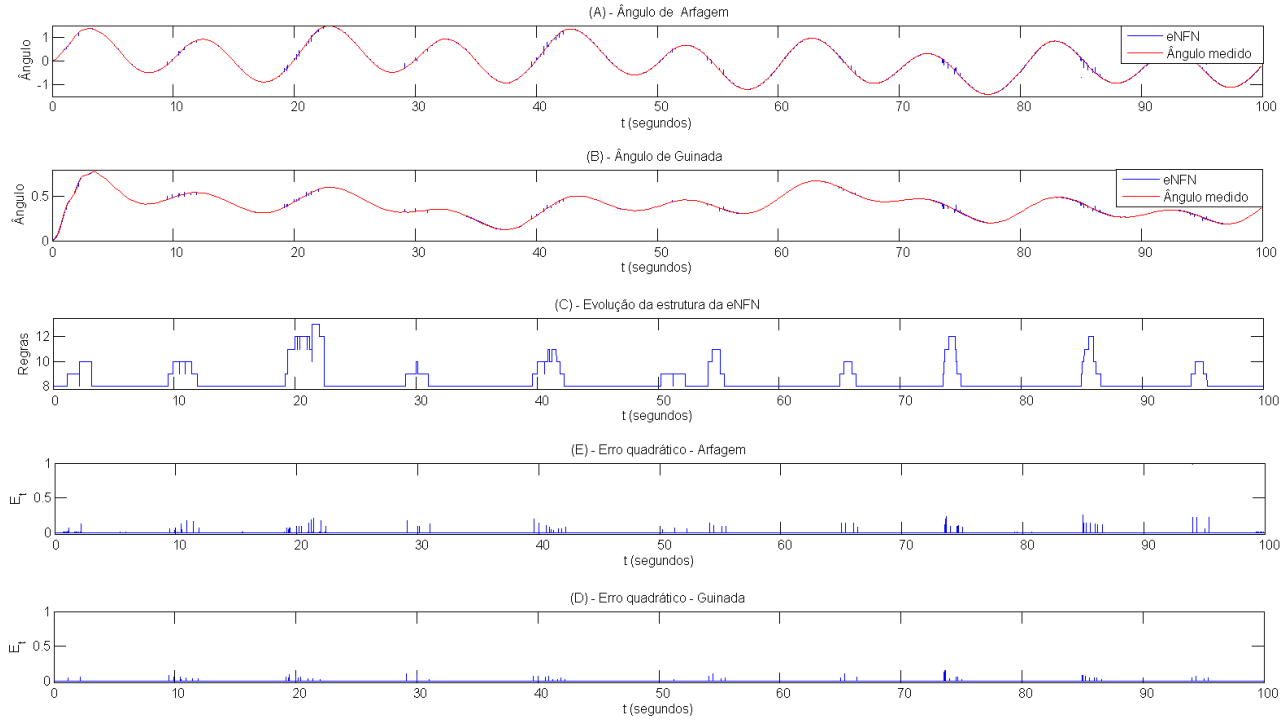

Fig. 5.20: (A) Ângulo de arfagem medido e estimado; (B) Ângulo de guinada medido e estimado; (C) Evolução da estrutura da eNFN; (D) Erro quadrático na estimação do ângulo de arfagem; (E) Erro quadrático na estimação do ângulo de guinada.

5.5.3.2.2 Estimação de Ação de Controle

Neste experimento a rede eNFN é empregada para estimar um passo à frente a voltagem aplicada aos rotores de arfagem e guinada. A Figura 5.21 mostra: (A) a ação de controle real e a estimada para o rotor de arfagem; (B) a ação de controle real e a estimada para o rotor de guinada; (C) a evolução da estrutura da eNFN; (D) e (E) o erro quadrático instantâneo E_t para a estimação da ação de controle do rotor de arfagem e guinada, respectivamente. Os resultados apresentados confirmam a boa acurácia da eNFN.

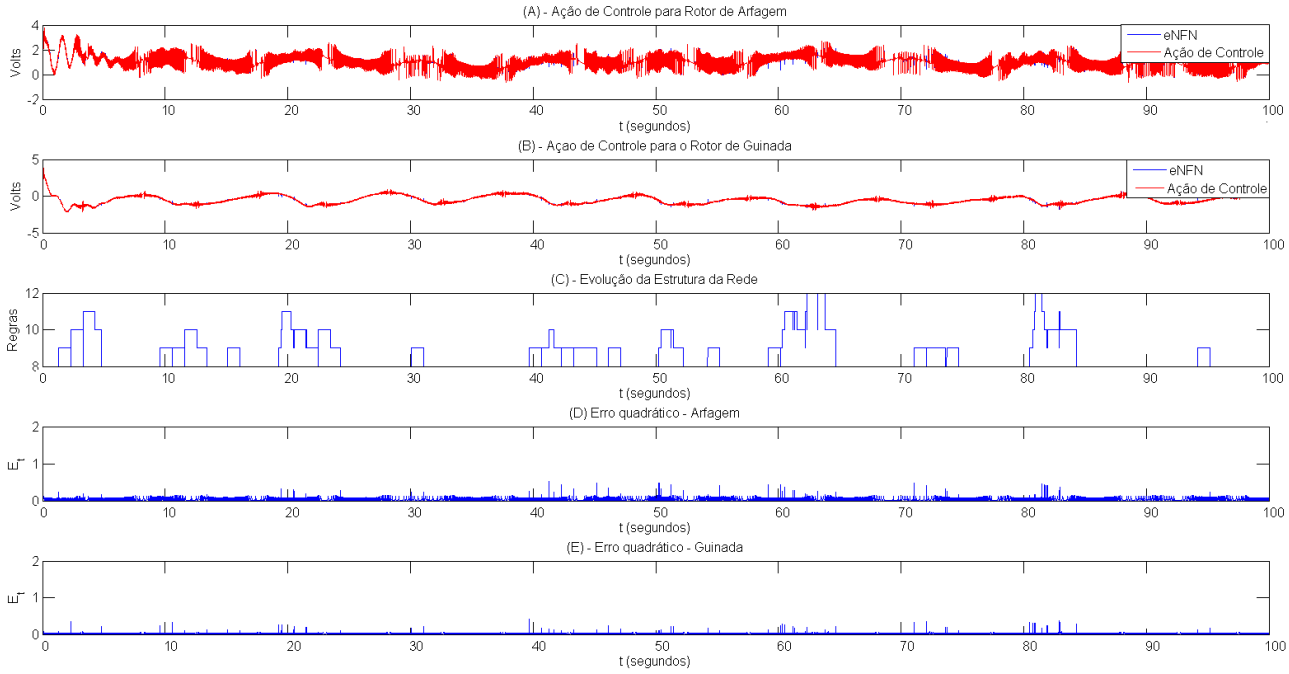

Fig. 5.21: (A) Ação de controle do rotor de arfagem, (B) Ação de controle do rotor de guinada; (C) Evolução da estrutura da eNFN; (D) Erro quadrático na estimação da ação de controle de arfagem; (E) Erro quadrático na estimação da ação de controle de guinada.

Os experimentos realizados em tempo real no TRMS mostram que a rede eNFN é eficiente, isto é, tem boa precisão e é rápida. Além da acurácia e da velocidade, a eNFN possui baixo custo computacional. Neste experimento a rede eNFN estimou a saída obtida de dois controladores independentes, o que torna o processo de estimação mais complexo.

5.6 Resumo

Neste capítulo avaliou-se e comparou-se o desempenho das redes eNFN com outros modelos evolutivos. Realizaram-se simulações computacionais na identificação de sistema não linear e em problemas de previsão. Os resultados obtidos mostram que as redes eNFN têm um desempenho comparável aos demais modelos evolutivos. O tempo de execução dos modelos

também foi avaliado. A rede eNFN foi a mais rápida entre os modelos avaliados, tendo um tempo de processamento cerca de uma ordem de grandeza menor do que a ordem de grandeza dos outros modelos evolutivos considerados.

Diante dos resultados obtidos nas simulações, pode-se concluir que as redes propostas possuem resultados comparáveis aos demais modelos evolutivos e um custo computacional menor. Estes resultados indicam que os modelos propostos são alternativas promissoras no desenvolvimento de sistemas *fuzzy* evolutivos. Devido ao baixo custo computacional e à boa precisão, os modelos propostos se mostram aptos para a utilização em sistemas que necessitam de rápidas respostas.

Após os bons resultados obtidos nas simulações, avaliou-se o desempenho das redes eNFN em problemas práticos, em tempo real. Os experimentos foram realizados em dois sistemas com dinâmica rápida, isto é, sistemas com alta taxa de amostragem. Foram desenvolvidos experimentos na estimação de variáveis de processo. Os resultados experimentais sugerem que a eNFN é efetiva para estimar variáveis de estado de processos com dinâmica rápida. Evidentemente, a rede também pode ser utilizada para processos lentos. Os experimentos também mostram que a estrutura da eNFN evolui de acordo com o fluxo dos dados de entrada, adequando a rede para manter uma estrutura compacta e uma baixa taxa de erro.

Capítulo 6

Redes com Seleção Adaptativa de Entradas

Este capítulo descreve a abordagem proposta para seleção adaptativa de entradas a partir de um fluxo de dados. O método objetiva modelos precisos e parcimoniosos, isto é, com pequeno número de parâmetros livres, baixo custo computacional e eficientes.

A Seção 6.1 apresenta uma breve introdução aos sistemas evolutivos com seleção adaptativa de entradas. O método de seleção adaptativa de entradas é descrito na Seção 6.2. A rede *Neo-Fuzzy-Neuron* (NFN) com seleção adaptativa de entradas é o assunto da Seção 6.3, enquanto a Seção 6.4 introduz o método de seleção adaptativa de entradas na rede *Neo-Fuzzy-Neuron* Evolutiva (eNFN). Na Seção 6.5 avalia-se a complexidade computacional das abordagens propostas. Por fim, a Seção 6.6 discorre sobre as considerações finais do capítulo.

6.1 Introdução

Uma característica importante dos sistemas *fuzzy* evolutivos é a sua flexibilidade para adaptar sua estrutura e parâmetros frente às variações observadas em um fluxo de dados. Contudo, uma limitação das abordagens tratadas na literatura é a pouca ou nenhuma flexibilidade para se escolherem as variáveis de entrada utilizando o fluxo de dados de entrada. Em geral, as variáveis de entrada são definidas ou utilizando-se conhecimento a respeito do sistema, ou/e em técnicas de seleção. Frequentemente, as variáveis de entrada, uma vez escolhidas, permanecem as mesmas (Angelov and Zhou, 2008b; Lemos et al., 2011c). Os algoritmos apresentados nesta seção abordam essa limitação na flexibilidade dos sistemas evolutivos.

Neste capítulo, é apresentada uma abordagem que permite a seleção adaptativa de entradas durante o processo de aprendizado incremental. Nesta abordagem, inicia-se a rede com uma ou mais variáveis de entrada e, a cada nova amostra, é possível, por meio de testes estatísticos, a inclusão, a exclusão, ou a permanência de uma variável de entrada. A topologia é definida utilizando-se informações sobre a qualidade da rede, sendo esta definida pela precisão e pelo número de parâmetros livres. Cada entrada é processada uma única vez e as estatísticas são computadas recursivamente, não sendo necessário o armazenamento de valores passados. A abordagem proposta é aplicada à rede NFN (Capítulo 3) e à rede eNFN (Capítulo 4). Na rede

NFN com seleção adaptativa de entradas a estrutura da rede evolui pela inserção e exclusão de variáveis de entrada e na rede eNFN com seleção adaptativa de entradas a evolução da estrutura se dá, além da inclusão e exclusão de variáveis de entrada, pela inserção e eliminação de funções de pertinência.

6.2 Seleção Adaptativa de Entradas

Esta seção introduz a abordagem para seleção adaptativa de entradas para as redes NFN e eNFN. A seleção de variáveis requer a pré-seleção de um conjunto de possíveis variáveis de entrada. A partir desse conjunto, seleciona-se uma ou mais variáveis para iniciar a rede. A escolha do conjunto inicial de variáveis pode ser realizada ou a partir de conhecimento *a priori* sobre o sistema, ou empregando métodos de ordenação de variáveis (Guyon and Elisseeff, 2003). Este trabalho sugere uma abordagem estatística para a seleção de variáveis de entrada.

A estatística utilizada na seleção adaptativa de entradas é baseada no teste F de modelos aninhados, conforme Allen (1997) e Potts and Sammut (2004). O teste F avalia a qualidade dos modelos, considerando-se a precisão e o número de parâmetros livres. São analisados dois modelos, sendo um mais simples e o outro mais complexo. Neste trabalho, a complexidade dos modelos é caracterizada pelo número de variáveis de entrada e pelo número de funções de pertinência. Supondo-se que um modelo mais complexo é mais preciso e que um modelo mais simples pode ser aninhado ao mais complexo, o teste F analisa a relação custo/benefício entre o ganho na precisão e o aumento da complexidade do modelo. Este teste foi também empregado no aprendizado incremental de árvores de regressão linear (Potts and Sammut, 2004; Lemos et al., 2011a,b).

O teste F (Allen, 1997) utiliza a seguinte estatística:

$$F = \frac{(SQR_a - SQR_c)(N - p_c)}{SQR_c(p_c - p_a)}, \tag{6.1}$$

onde SQR_a e SQR_c correspondem, respectivamente, às somas dos quadrados residuais do modelo atual e do modelo candidato, p_a e p_c correspondem ao número de parâmetros dos dois modelos e N é o número de amostras utilizadas para estimar os parâmetros dos modelos. O número de parâmetros de um modelo é dado pela multiplicação do número de variáveis de entrada (n) pelo número de funções de pertinência (m). Assumindo-se que a distribuição dos resíduos é normal, F segue uma distribuição de *Fisher* com $(p_c - pa,\, N - p_c)$ graus de liberdade.

No teste F (Allen, 1997) os parâmetros dos modelos são estimados utilizando-se o mesmo número de amostras. No algoritmo proposto, o número de amostras utilizadas para estimar os parâmetros dos modelos pode não ser igual. Isto acontece porque os modelos candidatos são criados sempre que ocorre a troca do modelo atual por um modelo com qualidade superior, e o novo modelo atual continua com os mesmos parâmetros e estatísticas de quando era modelo candidato. Com isso, o número de amostras para estimar o modelo atual sempre será igual ou maior que o número de amostras utilizadas para os modelos candidatos.

Uma modificação no mecanismo do teste F (Allen, 1997) para tratar um número distinto de amostras entre os modelos foi proposta por Potts and Sammut (2004). Esta modificação possibilita a utilização do teste na análise de modelos incrementais e será utilizada neste trabalho para comparar o modelo atual com os modelos candidatos.

Na metodologia proposta, os modelos candidatos podem ser divididos em: "*modelos candidatos à inclusão de variáveis*" - que visam à substituição do modelo atual por um modelo mais complexo; e "*modelos candidatos à exclusão de variáveis*" - que buscam substituir o modelo atual por um modelo menos complexo. Em outras palavras, os modelos candidatos à inclusão de variáveis buscam melhorar a qualidade do modelo adicionando novas variáveis ao modelo atual, enquanto os modelos candidatos à exclusão têm por objetivo melhorar a qualidade do modelo, eliminando as variáveis pouco relevantes e/ou que reduzem sua acurácia.

Seja (n) o número de variáveis de entrada disponíveis e (a) o número de variáveis de entrada do modelo atual. Tem-se $(n-a)$ modelos candidatos à inclusão de variáveis, cada um com $(a+1)$ variáveis de entrada, e (a) modelos candidatos à exclusão de variáveis, cada um com $(a-1)$ variáveis de entrada. Por exemplo, supondo-se que o conjunto de possíveis variáveis de entrada é $[v_1,\ v_2,\ v_3,\ v_4,\ v_5]$ e o modelo atual é formado pelas variáveis $[v_1,\ v_2,\ v_3]$. Tem-se, então: dois modelos candidatos à inclusão de variáveis $[v_1,\ v_2,\ v_3,\ v_4]$ e $[v_1,\ v_2,\ v_3,\ v_5]$; três modelos candidatos à exclusão de variáveis $[v_1,\ v_2]$, $[v_1,\ v_3]$ e $[v_2,\ v_3]$.

A estatística do teste para os modelos candidatos à inclusão é dada por

$$F_{inc} = \frac{(SQR_a - SQR_c)(N_c - p_c)}{SQR_c(N_c - N_a + p_a - p_c)}, \tag{6.2}$$

onde N_a e N_c correspondem aos números de amostras utilizadas para estimar os parâmetros do modelo atual e dos modelos candidatos, respectivamente. F_{inc} segue uma distribuição de *Fisher* com $(N_c - N_a + p_a - p_c,\ N_c - p_c)$ graus de liberdade.

A estatística do teste para os modelos candidatos à exclusão F_{exc} é dada por (6.3) e segue uma distribuição de *Fisher* com $(N_c - N_a + p_c - p_a,\ N_c - p_c)$ graus de liberdade.

$$F_{exc} = \frac{(SQR_a - SQR_c)(N_c - p_c)}{SQR_c(N_c - N_a + p_c - p_a)}. \tag{6.3}$$

Para utilizar as estatísticas F_{inc} e F_{exc} é necessário o cálculo do p-valor para todos os modelos candidatos. O modelo candidato com o menor p-valor é selecionado e substituirá o modelo atual se o p-valor é menor que um nível de significância κ, um valor de entrada do algoritmo. No algoritmo proposto, uma mesma hipótese é testada λ vezes empregando o mesmo conjunto de dados. Neste caso, é necessário fazer uma correção de comparações múltiplas (Potts and Sammut, 2004) utilizando, por exemplo, a correção de Bonferroni. Esta correção é dada pela divisão do nível de significância desejado pelo número de testes. Desta maneira, um modelo candidato substitui o modelo corrente se:

$$p - valor < \frac{\kappa}{\lambda}, \tag{6.4}$$

onde λ é $(n-a)$ para F_{inc} e (a) para F_{exc}. Os valores típicos para o nível de significância κ geralmente são 0,01 ou 0,05.

Para utilizar o teste de seleção de modelos é necessário computar recursivamente, para cada amostra, a soma dos quadrados dos resíduos e o número de amostras dos modelos. O número de parâmetros dos modelos é atualizado na criação dos modelos e/ou na evolução de sua estrutura.

6.3 *Neo-Fuzzy-Neuron* com Seleção Adaptativa de Entradas (NFN-SAE)

Esta seção detalha a abordagem proposta para seleção adaptativa de entradas para a rede NFN (Capítulo 3). A rede NFN com seleção adaptativa de entradas (NFN-SAE) utiliza a estrutura da rede NFN e o método de seleção adaptativa de entradas descrito na Seção 6.2. Na NFN-SAE, o particionamento do domínio das variáveis de entrada é fixo, e a estrutura da rede evolui pela inclusão ou exclusão de variáveis de entrada.

O aprendizado e a seleção adaptativa de entradas da NFN-SAE podem ser resumidos em seis etapas:

1. Inicializar as funções de pertinência. O processo de inicialização das funções de pertinência ocorre de forma idêntica ao descrito na Seção 3.3. O número de funções de pertinência que compõe as partições é fixo e definido empiricamente. (Harris et al., 1996) e (Cao and Wang, 2009) sugerem a granulação do espaço de entrada de cada uma das dimensões da entrada com $m_i = 7$, $i = 1 \ldots n$. Este passo ocorre somente uma vez na inicialização da rede.

2. Selecionar as variáveis de entrada e definir os modelos. Após a definição das possíveis variáveis de entrada, selecionar as variáveis para compor o modelo atual e, depois, definir os modelos candidatos a inclusão e/ou a exclusão de variáveis.

3. Verificar se o valor da entrada atual é maior que max_{x_i}, ou menor que min_{x_i}, e decidir se o valor de max_{x_i} ou min_{x_i} deve ser atualizado.

4. Calcular os graus de pertinência da entrada x_{ti}, encontrar a função de pertinência mais ativa e atualizar o valor modal da função de pertinência mais ativa.

5. Calcular a saída e atualizar os parâmetros do modelo atual e de todos os modelos candidatos.

6. Encontrar o melhor modelo candidato, e decidir se o melhor modelo candidato deve substituir o modelo atual.

A Figura 6.1 ilustra o diagrama da abordagem proposta e o Algoritmo 5 detalha os processos de estimação da saída, atualização dos parâmetros e seleção adaptativa de entradas para a abordagem proposta. Os cálculos para estimação da saída e atualização dos pesos da rede são os mesmos descritos no Capítulo 3, e o método para adaptação de contexto e atualização do valor modal das funções de pertinência são aqueles introduzidos no Capítulo 4.

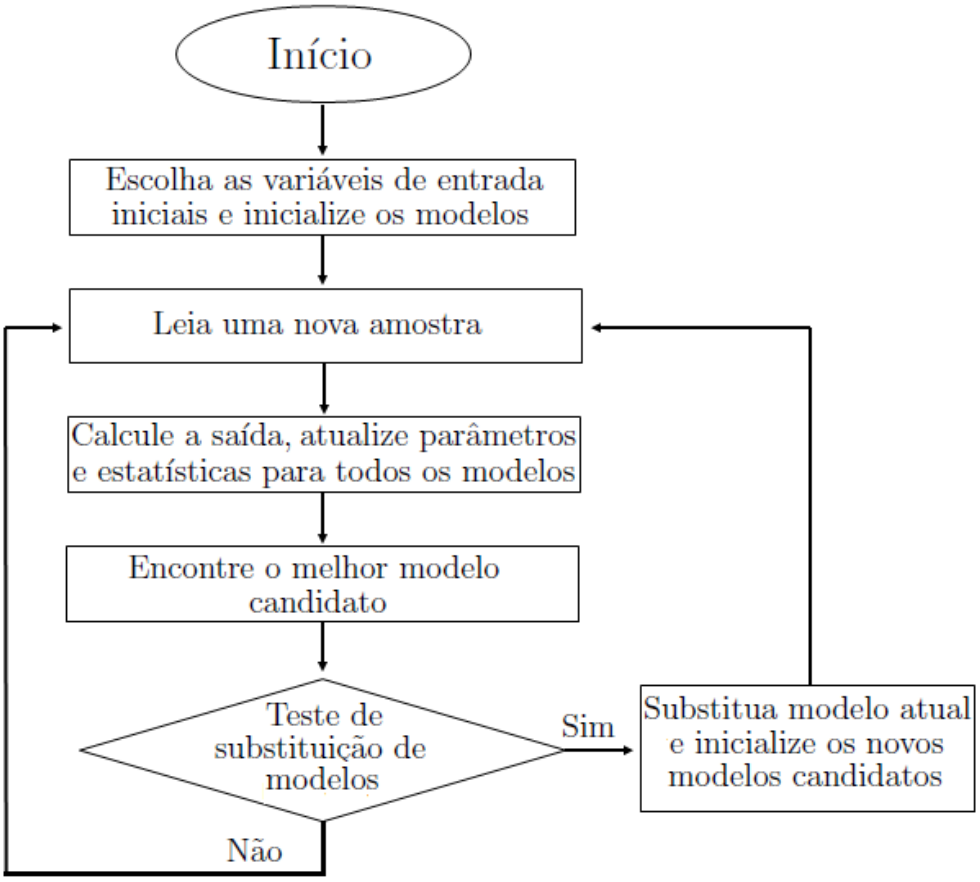

Fig. 6.1: Visão geral do aprendizado da rede NFN com seleção adaptativa de entradas.

Algorithm 5: Rede NFN-SAE.

Entrada: x_t, $\hat{y}_t$, n, κ;
Saída: y_t;
Inicializar b_{ij};
Definir e inicializar o modelo atual e os modelos candidatos;
for $t = 1, 2, ...$ **do**
  Ler x_t, $\hat{y}_t$;
  Verificar adaptação de contexto;
  for $i = 1 : n$ **do**
    Calcular $\mu_{A_{ik_i}}(x_{ti})$, $\mu_{A_{ik_i+1}}(x_{ti})$;
  end for
  Atualizar b_{ij};
  //**Modelo Atual**;
  Calcular y_t;
  Calcular α;
  Atualizar N_a, p_a, SQR_a, q_{ik_i}, q_{ik_i+1};
  //**Modelos Candidatos** - (nc é o número de modelos candidatos);
  for $l = 1 : nc$ **do**
    Calcular y_t^l;
    Calcular α;
    Atualizar N_c^l, p_c^l, SQR_c^l, $q_{ik_i}^l$, $q_{ik_i+1}^l$;
  end for
  // **Seleção Adaptativa de Entradas**;
  for $l = 1 : nc$ **do**
    Computar F^l;
    Computar p_valor^l;
  end for
  Identificar o modelo candidato com o menor p_valor (modelo indexado por z);
  // **Decidir se modelo atual deve ser substituído**;
  if $p\_valor^z < \frac{\kappa}{\lambda}$ **then**
    Substituir o modelo atual pelo modelo candidato indexado por z;
    Modelo atual recebe estatísticas e pesos do modelo candidato z;
    Definir e inicializar os novos modelos candidatos;
  end if
end for

6.4 *Neo-Fuzzy-Neuron* Evolutiva com Seleção Adaptativa de Entradas

A abordagem proposta para a rede NFN evolutiva com seleção adaptativa de entradas combina a abordagem para evolução da estrutura da rede eNFN (Capítulo 4) e o método de seleção adaptativa de entradas apresentado na Seção 6.2. Ao contrário das abordagens anteriores, nesta, a estrutura da rede evolui tanto pela inclusão e exclusão de funções de pertinência quanto pela inserção e eliminação de variáveis de entrada. Os processos de evolução da estrutura e de seleção adaptativa de variáveis de entrada ocorrem simultaneamente com o ajuste nos pesos da rede. A rede inicia-se com uma ou mais variáveis de entrada e o domínio de cada variável de entrada inicialmente é granulado por duas funções de pertinência triangulares e complementares. A cada nova amostra de dados apresentada, a rede pode manter, incluir ou excluir, variáveis de entrada e funções de pertinência.

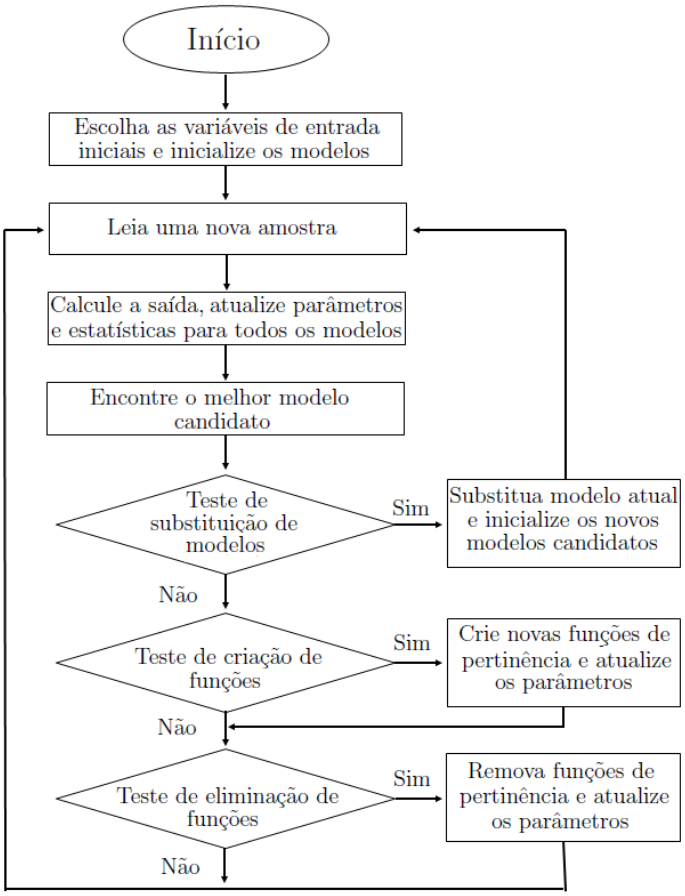

Fig. 6.2: Visão geral do aprendizado da rede eNFN com seleção adaptativa de entradas.

A Figura 6.2 mostra uma visão geral da abordagem para a rede *Neo-Fuzzy-Neuron* evolutiva com seleção adaptativa de entradas. Baseados nesta abordagem foram propostos dois algoritmos:

- **eNFN-SAE** - Rede *Neo-Fuzzy-Neuron* Evolutiva com Seleção Adaptativa de Entradas: na rede eNFN-SAE (Algoritmo 6) o processo de granulação do domínio das variáveis de entrada e o cálculo do grau de ativação das funções de pertinência ocorre somente uma vez para cada amostra do fluxo de dados.

- **X-eNFN-SAE** - Rede *Neo-Fuzzy-Neuron* Evolutiva com Seleção Adaptativa de Entradas Estendida: a rede X-eNFN-SAE (Algoritmo 7) prima por uma maior independência entre o modelo atual e os modelos candidatos. Por isso, o processo de granulação do domínio das variáveis de entrada ocorre de forma independente para cada modelo e, consequentemente, o cálculo do grau de ativação das funções de pertinência deve ser executado para cada um dos modelos. Essa maior independência entre os modelos tem um impacto no custo computacional do algoritmo.

6.5 Complexidade Computacional

Nesta seção, é realizada a análise da complexidade computacional (temporal e espacial) das redes com seleção adaptativa de entradas. Utiliza-se para fins de comparação os mesmos algoritmos apresentados na Seção 4.9. Primeiro é apresentada a complexidade temporal e em seguida a complexidade espacial. Simulações computacionais que analisam o tempo de processamento das redes propostas neste capítulo são apresentadas no Capítulo 7.

6.5.1 Complexidade Temporal

Para avaliar-se a complexidade temporal das redes com seleção adaptativa de entradas (NFN-SAE, eNFN-SAE, X-eNFN-SAE), considera-se que o algoritmo calcula, simultaneamente, a saída e atualiza os parâmetros para o modelo atual e para os modelos candidatos. A complexidade para o modelo atual é dada, como na eNFN_1 (com uma única saída), pelo número de variáveis de entrada e é $O(n)$. Para cada um dos modelos candidatos, a complexidade será no máximo $O(n-1)$. Este caso ocorrerá quando o modelo atual for composto por todas as variáveis de entrada disponíveis e o número de modelos candidatos for igual a n. Neste caso, a complexidade de todos os modelos candidatos será $O(n(n-1))$. A complexidade das redes com seleção adaptativa é dada pela soma da complexidade do modelo atual com a dos modelos candidatos, sendo $O(n + n(n-1))$. Assim, a complexidade temporal das redes NFN-SAE, eNFN-SAE, e X-eNFN-SAE é $O(n^2)$.

A Tabela 6.1 ilustra a complexidade temporal das redes propostas neste capítulo e dos modelos apresentados na Seção 4.9. As redes com seleção adaptativa de entradas (NFN-SAE, eNFN-SAE, X-eNFN-SAE) possuem uma complexidade temporal maior que as eNFN, pois necessitam calcular a saída do modelo atual e também dos modelos candidatos. Quando se comparam as redes com seleção de entradas com os modelos alternativos, verifica-se que elas possuem uma complexidade menor ou igual.

Algorithm 6: Rede eNFN-SAE.

Entrada: x_t, $\hat{y}_t$, κ, β, n, η, w;
Saída: y_t;
Inicializar b_{ij};
Definir e inicializar o modelo atual e os modelos candidatos;
for $t = 1, 2, ...$ **do**
  Ler x_t, $\hat{y}_t$;
  Verificar adaptação de contexto;
  Calcular $\mu_{A_{ik_i}}(x_{ti})$, $\mu_{A_{ik_i+1}}(x_{ti})$ (para as variáveis disponíveis);
  Atualizar b_{ij};
  //**Modelo Atual**;
  Calcular y_t;
  Calcular α, $\hat{\mu}_{g_t}$, $\hat{\sigma}^2_{g_t}$;
  Atualizar q_{ik_i}, q_{ik_i+1}, N_a, p_a, SQR_a;
  //**Modelos Candidatos** - (nc é o número de modelos candidatos);
  for $l = 1 : nc$ **do**
    Calcular y^l_t;
    Calcular α, $\hat{\mu}_{g^l_t}$, $\hat{\sigma}^2_{g^l_t}$;
    Atualizar $q^l_{ik_i}$, $q^l_{ik_i+1}$, p^l_c, N^l_c,SQR^l_c;
  end for
  // **Seleção Adaptativa de Entradas**;
  for $l = 1 : nc$ **do**
    Computar F^l;
    Computar p_valor^l;
  end for
  Identificar o modelo candidato com o menor p_valor (modelo indexado por z);
  Definir $CriaExclui = 1$;
  // **Decidir se modelo atual deve ser substituído**;
  if $p\_valor^z < \frac{\kappa}{\lambda}$ **then**
    $CriaExclui = 0$;
    Substituir o modelo atual pelo modelo candidato indexado por z;
    Modelo atual recebe estatísticas e pesos do modelo candidato z;
    Definir e inicializar os novos modelos candidatos;
  end if
  if $CriaExclui = 1$ **then**
    //**Realizar o procedimento para inclusão e/ou exclusão de funções de pertinência para todas as variáveis disponíveis**;
    for $i = 1 : n$ **do**
      // **Criação e inserção de funções de pertinência**;
      Encontrar b^*_i;
      Calcular $\hat{\mu}_{b^*_{ti}}$, τ $dist$;
      if $(\hat{\mu}_{b^*_{ti}} > \hat{\mu}_{g_t} + \hat{\sigma}^2_{g_t})$ *e* $(dist > \tau)$ **then**
        Criar e inserir funções de pertinência;
        Atualizar parâmetros;
      end if
      // **Exclusão de funções de pertinência**;
      Atualizar $idade_i$;
      Encontrar b^-_i;
      if $(idade_{b^-_i} > \omega)$ *e* $(m_i > 2)$ **then**
        Excluir a função de pertinência indexada por b^-_i;
        Atualizar parâmetros;
      end if
    end for
  end if
end for

Algorithm 7: Rede X-eNFN-SAE.

Entrada: x_t, $\hat{y}_t$, κ, β, n, η, w;
Saída: y_t;
Inicializar b_{ij};
Definir e inicializar o modelo atual e os modelos candidatos;
for $t = 1, 2, ...$ **do**
  Ler x_t, $\hat{y}_t$;
  Verificar adaptação de contexto;
  //**Modelo Atual**;
  Calcular $\mu_{A_{ik_i}}(x_{ti})$, $\mu_{A_{ik_i+1}}(x_{ti})$ (somente para as variáveis ativas para o modelo atual);
  Calcular y_t;
  Calcular α, $\hat{\mu}_{g_t}$, $\hat{\sigma}^2_{g_t}$;
  Atualizar b_{ij}, q_{ik_i}, q_{ik_i+1}, N_a, p_a, SQR_a;
  //**Modelos Candidatos** - (nc é o número de modelos candidatos);
  for $l = 1 : nc$ **do**
    Calcular $\mu_{A_{ik_i}}(x_{ti})$, $\mu_{A_{ik_i+1}}(x_{ti})$ (somente para as variáveis ativas para cada modelo candidato);
    Calcular y^l_t;
    Calcular α, $\hat{\mu}_{g^l_t}$, $\hat{\sigma}^2_{g^l_t}$;
    Atualizar b^l_{ij}, $q^l_{ik_i}$, $q^l_{ik_i+1}$, p^l_c, N^l_c,SQR^l_c;
  end for
  // **Seleção Adaptativa de Entradas**;
  for $l = 1 : nc$ **do**
    Computar F^l;
    Computar p_valor^l;
  end for
  Identificar o modelo candidato com o menor p_valor (modelo indexado por z);
  Definir $CriaExclui = 1$;
  // **Decidir se modelo atual deve ser substituído**;
  if $p\_valor^z < \frac{\kappa}{\lambda}$ **then**
    $CriaExclui = 0$;
    Substituir o modelo atual pelo modelo candidato indexado por z;
    Modelo atual recebe estatísticas e pesos do modelo candidato z;
    Definir e inicializar os novos modelos candidatos;
  end if
  //**Criar ou excluir funções de pertinência)**;
  if $CriaExclui = 1$ **then**
    //**Realizar o procedimento para inclusão e/ou exclusão de funções de pertinência para todos os modelos (atual e candidatos)**;
    //nm é o número total de modelos (atual e candidatos);
    for $l = 1 : nm$ **do**
      for $i = 1 : n$ **do**
        // **Criação e inserção de funções de pertinência**;
        Encontrar $b^{*^l}_i$;
        Calcular $\hat{\mu}_{b^{*^l}_{ti}}$, τ $dist$;
        if $(\hat{\mu}_{b^{*^l}_{ti}} > \hat{\mu}_{g^l_t} + \hat{\sigma}^2_{g^l_t})$ e $(dist > \tau)$ **then**
          Criar e inserir funções de pertinência;
          Atualizar parâmetros;
        end if
        // **Exclusão de funções de pertinência**;
        Atualizar $idade^l_i$;
        Encontrar $b^{-^l}_i$;
        if $(idade_{b^{-^l}_i} > \omega)$ e $(m^l_i > 2)$ **then**
          Excluir a função de pertinência indexada por $b^{-^l}_i$;
          Atualizar parâmetros;
        end if
      end for
    end for
  end if
end for

Tab. 6.1: Complexidade temporal dos algoritmos evolutivos.

Modelo	Custo Temporal
DENFIS	$O(rn^2)$
eMG	$O((rn)^3)$
eNFN_1	$O(n)$
eNFN_S	$O(ns)$
eNFN-SAE	$O(n^2)$
eTS	$O(rn^3)$
NFN-SAE	$O(n^2)$
X-eNFN-SAE	$O(n^2)$
xTS	$O(rn^3)$

6.5.2 Complexidade Espacial

A complexidade espacial das redes propostas, isto é, o custo espacial para armazenar todas as constantes, variáveis e parâmetros são mostrados na Tabela 6.2, onde n é o número de variáveis de entrada, m o número de funções de pertinência, e s o número de saídas. Assim como na Seção 4.9.1 assume-se o custo espacial de 2 bytes para inteiro e 4 bytes para número real.

Tab. 6.2: Complexidade espacial dos algoritmos propostos.

Modelo	Custo Espacial
eNFN_1	$62 + 32n + 20nm$
eNFN_S	$56 + 32n + 16nm + 8s + 48snm$
eNFN-SAE	$100 + 84n + 20nm + 4n^2 + 8n^2m$
NFN-SAE	$62 + 64n + 16nm + 4n^2$
X-eNFN-SAE	$102 + 84n + 20nm + 12n^2 + 40n^2m$

Como esperado, as redes com seleção adaptativa de entradas possuem um maior custo espacial que as rede eNFN. Isto se deve à necessidade de se calcular a saída para n modelos candidatos. Pode-se perceber também que a rede NFN-SAE tem um menor custo espacial entre as redes com seleção adaptativa de entradas, seguida pela rede eNFN-SAE. O maior custo espacial entre as redes introduzidas neste trabalho é o da rede X-eNFN-SAE. Este maior custo espacial se deve à necessidade da rede calcular o grau de ativação das funções de pertinência e o evoluir a estrutura de forma independente para o modelo atual e para cada um dos modelos candidatos. Para ilustrar os resultados apresentados na Tabela 6.2, é apresentada a seguir a curva de crescimento do custo de armazenamento. A Figura 6.3 mostra o crescimento do custo espacial para as redes com seleção adaptativa de entradas. Nesta figura, utilizou-se o custo espacial ilustrado na Tabela 6.2, o número de funções de pertinência $m = 10$, e o número de entradas de $n = 1, \ldots, 20$ para gerar o gráfico.

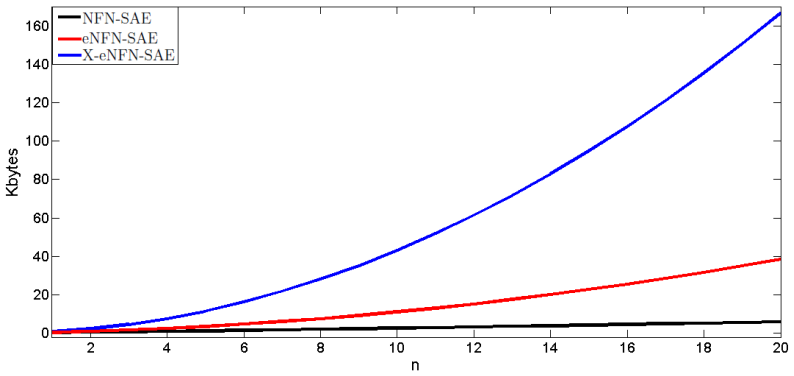

Fig. 6.3: Custo espacial das redes com seleção adaptativa de entradas.

Na Figura 6.3 pode-se ver que a rede NFN-SAE possui um crescimento quase linear no seu custo espacial, enquanto o crescimento das redes eNFN-SAE e X-eNFN-SAE tende para o quadrático. Apesar do maior custo computacional das redes com seleção adaptativa de entradas em relação à rede eNFN (o que é esperado e aceitável), o maior custo espacial é de apenas 160 KBytes para a X-eNFN-SAE com 20 variáveis de entrada e não passa de 40 KBytes para a rede eNFN-SAE com o mesmo número de entradas.

6.6 Resumo

Este capítulo apresentou uma nova abordagem para o desenvolvimento de modelos *neuro-fuzzy* com seleção adaptativa de entradas. A metodologia utiliza uma rede *neuro-fuzzy* e emprega testes estatísticos que possibilitam a seleção adaptativa de variáveis de entrada a partir de um fluxo de dados. O teste estatístico empregado considera a acurácia e o número de parâmetros a fim de obter-se um modelo preciso, parcimonioso e com baixo custo computacional. Foram introduzidas duas abordagens. A primeira utiliza o método de seleção adaptativa de entradas descrito neste capítulo e a estrutura da rede NFN apresentada no Capítulo 3. A segunda abordagem emprega o método de seleção adaptativa de entradas e a estrutura de rede *neuro-fuzzy* evolutiva eNFN proposta no Capítulo 4. Para a segunda abordagem foram propostos dois algoritmos. O primeiro algoritmo proposto, o eNFN-SAE, possui um menor custo computacional em consequência da maior rigidez entre os modelos candidatos. Nesse algoritmo, o cálculo do grau de ativação das funções de pertinência e o procedimento de granulação do domínio das variáveis de entrada ocorrem uma única vez para cada amostra. O segundo, o X-eNFN-SAE, possui uma maior independência entre os modelos e um maior custo computacional, pois todas as etapas são executadas de forma independente para o modelo atual e para os modelos candidatos. As redes evolutivas com seleção adaptativa de entradas propostas apresentam uma baixa complexidade computacional, se comparadas com outros modelos evolutivos e, principalmente, se comparadas com modelos que possuem métodos de seleção de entradas incorporados. Essas redes são uma alternativa para a para construção de modelos *fuzzy* com alto nível de adaptabilidade e seleção de entradas.

Capítulo 7

Resultados de Simulações - Redes com Seleção Adaptativa de Entradas

Este capítulo apresenta simulações computacionais realizadas para avaliar as redes com seleção adaptativa de entradas (eNFN-SAE, NFN-SAE e X-eNFN-SAE)[1]. As redes foram avaliadas em problemas de identificação e previsão. Os resultados obtidos são comparados com algoritmos de modelagem evolutivos alternativos como DENFIS (Kasabov and Song, 2002), eMG (Lemos et al., 2011c), eNFN (Capítulo 4), eTS (Angelov and Filev, 2004) e xTS (Angelov and Zhou, 2006), ou seja, os mesmos utilizados no Capítulo 5. A próxima seção contém uma breve introdução ao capítulo. Os resultados na identificação de processo não linear são apresentados na Seção 7.2, e os resultados obtidos em problemas de previsão são apresentados nas seções 7.3 e 7.4. A análise do tempo de processamento é descrita na Seção 7.5 e a escalabilidade das redes com seleção adaptativa de entradas na Seção 7.6. Concluindo o capítulo, a Seção 7.7 apresenta uma discussão final sobre o desempenho das redes com seleção adaptativa de entradas.

7.1 Introdução

As redes com seleção adaptativa de entradas podem ser iniciadas com qualquer número de variáveis de entrada entre 1 e n, onde n é o número de variáveis disponíveis. Para as simulações foram definidas duas configurações para as redes eNFN-SAE, NFN-SAE e X-eNFN-SAE. Na primeira, iniciam-se as redes com uma única variável de entrada[2]. Na segunda, as redes são iniciadas com todas as variáveis de entrada disponíveis[3]. Os resultados apresentados se referem a configuração de rede que obteve o melhor desempenho.

As simulações computacionais deste capítulo seguem a mesma metodologia do Capítulo 5 (Seção 5.2.1). Nesta metodologia, o conjunto de dados é dividido em dois subconjuntos com 50% das amostras cada, um subconjunto para estimar os parâmetros e o outro para avaliar o desempenho. A faixa de variação dos parâmetros dos algoritmos evolutivos na estimação do melhor conjunto de parâmetros é ilustrada na Tabela 7.1.

[1] As redes eNFN-SAE, NFN-SAE, e X-eNFN-SAE foram desenvolvidas em Matlab.

[2] As redes eNFN-SAE$_1$, NFN-SAE$_1$ e X-eNFN-SAE$_1$ foram iniciadas com uma única variável de entrada.

[3] As redes eNFN-SAE$_2$, NFN-SAE$_2$ e X-eNFN-SAE$_2$ foram iniciadas com todas as variáveis disponíveis.

Tab. 7.1: Faixa de variação dos parâmetros dos modelos evolutivos.

Modelo	Parâmetro	*Step*
DENFIS	$dthr = 0.01, ..., 0.10$	0.01
	$mofn = 1, ..., 10$	1.00
eMG	$\lambda = 0.05$	fixo
	$w = 5, ..., 40$	5.00
	$\sum_{init} = 10^{-1}, ..., 10^{-4}$	1.00
	$\alpha = 0.01$	fixo
eNFN	$\beta = 0.01$	fixo
	$\omega = 50, ..., 250$	50.00
	$\eta = 5, ..., 20$	5.00
eTS	$r = 0.02, ..., 0.10$	0.02
	$\Omega = 100, ..., 900$	50.00
xTS	$\Omega = 100, ..., 900$	50.00
eNFN-SAE	$\beta = 0.01$	fixo
X-eNFN-SAE	$\omega = 50, ..., 250$	50.00
	$\eta = 5, ..., 20$	5.00
	$\lambda = 0.01$	fixo
NFN-SAE	$\beta = 0.01$	fixo
	$\lambda = 0.01$	fixo
	$m = 2, ..., 10$	1.00

O desempenho é avaliado pelas medidas de erro RMSE (5.1) e SQR (5.2), e pelo teste estatístico MGN (5.3).

7.2 Identificação de Processo Não Linear

Nesta seção, as redes propostas são comparadas com modelos alternativos na identificação de processo não linear clássico. O processo não linear a ser identificado é dado por:

$$y_t = \frac{y_{t-1}y_{t-2}(y_{t-1} - 0.5)}{1 + (y_{t-1})^2 + (y_{n-2})^2} + u_{t-1}, \tag{7.1}$$

onde $u_t = sin(2\pi t/25)$, e $y_0 = y_1 = 0$.

O objetivo é prever a saída corrente com base em informações passadas e nas saídas. O modelo para este conjunto de dados é descrito por:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, u_{t-1}), \tag{7.2}$$

onde $\hat{y}_t$ é a saída. Para o processo de aprendizagem e evolução dos modelos foram criadas 5200 amostras, 2600 para estimar os parâmetros e 2600 para avaliar os modelos.

Os parâmetros dos algoritmos são definidos como se segue: DENFIS ($dthr = 0.07$, $mofn = 3$); eMG ($\lambda = 0.05$, $w = 5$, $\sum_{init} = 10^{-2}.I_3$, $\alpha = 0.01$); eNFN ($\beta = 0.01$, $w = 100$, $\eta = 10$); eNFN-SAE$_1$ ($\beta = 0.01$, $w = 100$, $\lambda = 0.01$, $\eta = 10$); eTS ($r = 0.06$, $\Omega = 550$); NFN-SAE$_2$ ($m = 4$, $\beta = 0.01$, $\lambda = 0.01$); X-eNFN-SAE$_2$ ($\beta = 0.01$, $w = 100$, $\lambda = 0.01$, $\eta = 10$); xTS ($\Omega = 650$). A rede eNFN-SAE$_1$ inicia com y_{t-1}, as redes NFN-SAE$_2$ e X-eNFN-SAE$_2$ iniciam com todas as três variáveis de entrada, enquanto DENFIS, eMG, eNFN, eTS e xTS iniciam e mantêm todas as três variáveis de entrada.

A evolução da estrutura (número de funções de pertinência e número de variáveis de entrada) das redes com seleção adaptativa de entradas para cada amostra do conjunto de dados pode ser vista na Figura 7.1. Para este conjunto de dados, os mecanismos de seleção adaptativa de entradas das redes eNFN-SAE$_1$ e X-eNFN-SAE$_2$ selecionaram y_{t-1} e y_{t-2}, e o da rede NFN-SAE$_2$ selecionou as três variáveis de entrada. Os valores esperados e os previstos pela NFN-SAE$_2$ na identificação de processo não linear são ilustrados na Figura 7.2.

Os resultados experimentais são apresentados na Tabela 7.2. Eles indicam que o melhor desempenho foi obtido pela NFN-SAE$_2$, seguido pela eNFN, X-eNFN-SAE$_2$, DENFIS, eMG e eNFN-SAE$_1$. Os resultados obtidos por estes modelos são comparáveis e superam os obtidos pelo eTS e xTS em uma ordem de grandeza.

O logaritmo da soma dos quadrados residuais ($Log\ (SQR)$) é apresentado na Figura 7.3. A rede NFN-SAE$_2$ possui a menor curva de erro, sendo comparável à curva da eNFN. Entre os demais a menor curva de erro é a da X-eNFN-SAE$_2$, DENFIS, eMG e eNFN-SAE$_1$, respectivamente. As maiores curvas de erro foram as obtidas pelo eTS e xTS.

A Tabela 7.3 mostra os resultados comparativos da rede eNFN-SAE$_1$ pelo teste MGN. Os resultados apresentados sugerem que dado um nível de significância de 0,05, a eNFN-SAE$_1$ é estatisticamente superior ao eTS e xTS. O resultado das comparações entre o NFN-SAE$_2$ e os outros modelos evolutivos utilizando o teste de MGN são descritos na Tabela 7.4. Com um nível de significância de 0,05, percebe-se que o NFN-SAE$_2$ é estatisticamente superior ao

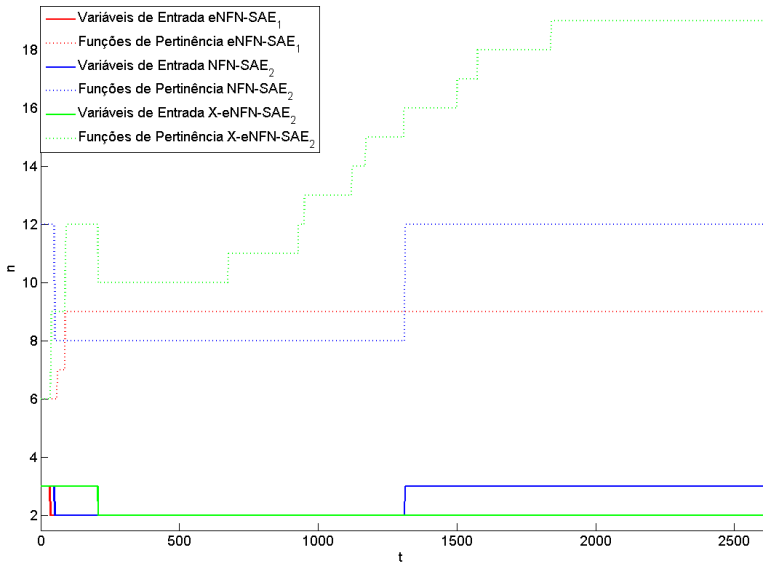

Fig. 7.1: Evolução da estrutura das redes com seleção adaptativa de entradas na identificação de processo não linear.

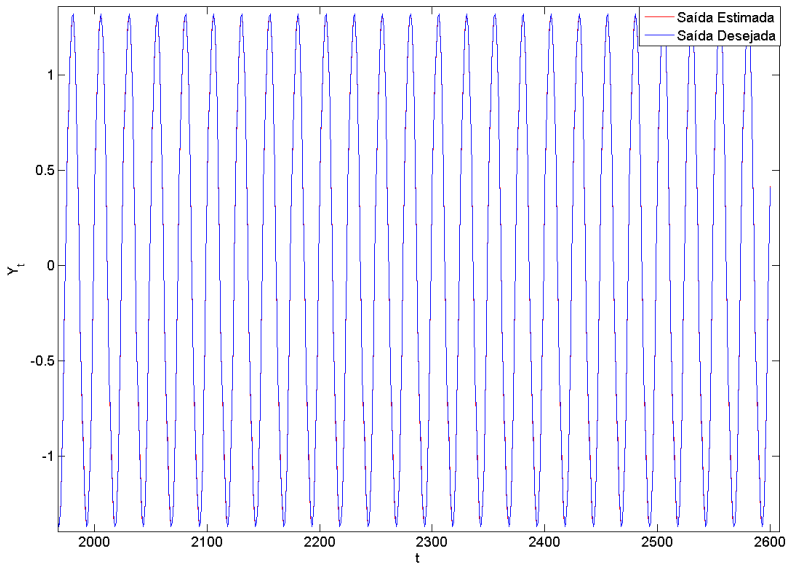

Fig. 7.2: Identificação de processo não linear.

Tab. 7.2: Desempenho na identificação de processo não linear.

Modelo	N. Regras	RMSE
DENFIS	10	0.0530
eMG	17	0.0654
eNFN	28	0.0473
eNFN-SAE$_1$	07	0.0667
eTS	07	0.7994
NFN-SAE$_2$	12	0.0469
X-eNFN-SAE$_2$	19	0.0494
xTS	05	0.7990

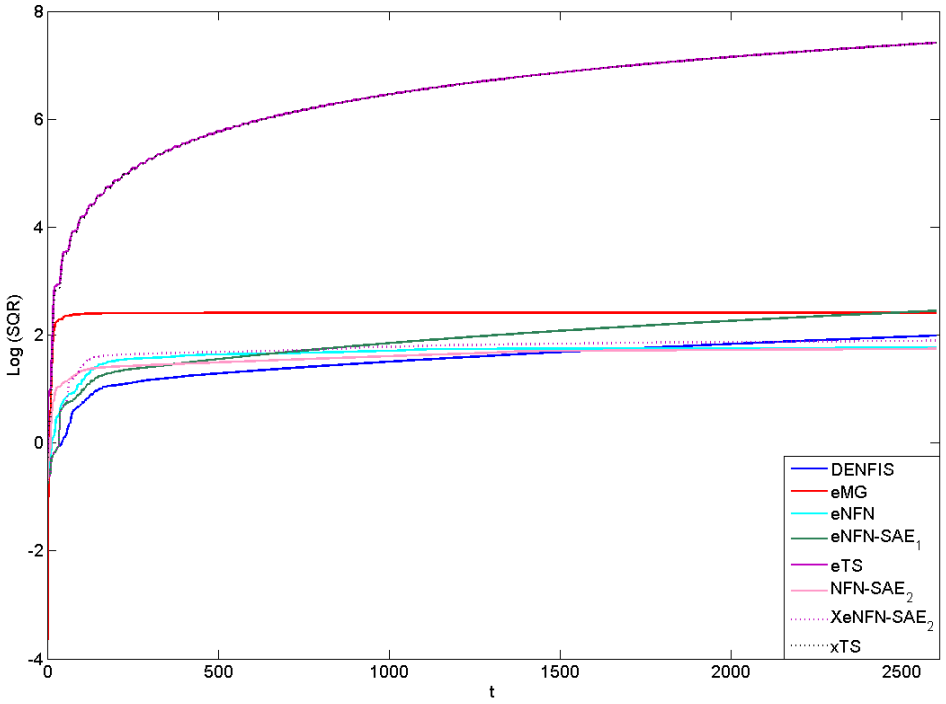

Fig. 7.3: Logaritmo da soma dos quadrados residuais na identificação de processo não linear.

DENFIS, eMG, eNFN-SAE$_2$, eTS, X-eNFN-SAE e xTS. Os resultados também mostram que o desempenho da NFN-SAE$_2$ é comparável ao da eNFN. O comparativo entre a X-eNFN-SAE$_2$ e os demais modelos pelo teste MGN é ilustrado na Tabela 7.5. Dado um nível de significância de 0,05, os resultados apresentados nesta tabela mostram que a X-eNFN-SAE$_2$ é estatisticamente superior ao DENFIS, eMG, eNFN-SAE$_1$, eTS e xTS, e inferior a eNFN e NFN-SAE$_2$.

Tab. 7.3: Avaliação da eNFN-SAE$_1$ pelo teste MGN na identificação de processo não linear.

Modelo	MGN	p-valor
DENFIS vs eNFN-SAE$_1$	15.4702	0.0000
eMG vs eNFN-SAE$_1$	0.9741	0.1650
eNFN vs eNFN-SAE$_1$	19.5082	0.0000
eNFN-SAE$_1$ vs eTS	229.7587	0.0000
NFN-SAE$_2$ vs eNFN-SAE$_1$	19.6652	0.0000
X-eNFN-SAE$_2$ vs eNFN-SAE$_1$	17.3040	0.0000
eNFN-SAE$_1$ vs xTS	229.5175	0.0000

Tab. 7.4: Avaliação da NFN-SAE$_2$ pelo teste MGN na identificação de processo não linear.

Modelo	MGN	p-valor
NFN-SAE$_2$ vs DENFIS	7.5719	0.0000
NFN-SAE$_2$ vs eMG	18.9387	0.0000
NFN-SAE$_2$ vs eNFN	0.5590	0.2881
NFN-SAE$_2$ vs eNFN-SAE$_1$	19.6652	0.0000
NFN-SAE$_2$ vs eTS	327.4922	0.0000
NFN-SAE$_2$ vs X-eNFN-SAE$_2$	3.4184	0.0003
NFN-SAE$_2$ vs xTS	327.0980	0.0000

7.3 Previsão da Série Temporal de Mackey-Glass

Nesta seção, avaliam-se os modelos na previsão da série temporal clássica de Mackey-Glass. O conjunto de dados é os mesmo empregado na Seção 5.2.3, o qual consiste de 3500 amostras, 1750 para estimar os parâmetros e 1750 para avaliar o desempenho pelo RMSE, SQR e MGN.

Os parâmetros dos algoritmos são definidos como se segue: DENFIS ($dthr = 0.1$, $mofn = 3$); eMG ($\lambda = 0.05$, $w = 25$, $\sum_{init} = 10^{-2}.I_4$, $\alpha = 0.01$); eNFN ($\beta = 0.01$, $w = 100$, $\eta = 5$); eNFN-SAE$_1$ ($\beta = 0.01$, $w = 100$, $\lambda = 0.01$, $\eta = 10$); eTS ($r = 0.06$, $\Omega = 500$); NFN-SAE$_1$ ($m = 4$, $\beta = 0.01$, $\lambda = 0.01$); X-eNFN-SAE$_1$ ($\beta = 0.01$, $w = 100$, $\lambda = 0.01$, $\eta = 10$); xTS ($\Omega = 500$). As redes eNFN-SAE$_1$, NFN-SAE$_1$ e X-eNFN-SAE$_1$ iniciam com x_t, enquanto DENFIS, eMG, eNFN, eTS e xTS iniciam e mantêm todas as quatro variáveis de entrada.

Tab. 7.5: Avaliação da X-eNFN-SAE$_2$ pelo teste MGN na identificação de processo não linear.

Modelo	MGN	p-valor
X-eNFN-SAE$_2$ vs DENFIS	4.0030	0.0000
X-eNFN-SAE$_2$ vs eMG	14.9537	0.0000
eNFN vs X-eNFN-SAE$_2$	4.4553	0.0000
X-eNFN-SAE$_2$ vs eNFN-SAE$_1$	17.3040	0.0000
X-eNFN-SAE$_2$ vs eTS	310.2632	0.0000
NFN-SAE$_2$ vs X-eNFN-SAE$_2$	3.4184	0.0003
X-eNFN-SAE$_2$ vs xTS	309.9409	0.0000

A seleção adaptativa de variáveis de entrada e a evolução da estrutura das redes eNFN-SAE$_1$, NFN-SAE$_1$ e X-eNFN-SAE$_1$ é ilustrada na Figura 7.4. O mecanismo de seleção de entradas das redes eNFN-SAE$_1$ e NFN-SAE$_1$ selecionou x_t e x_{t-12}, enquanto o mecanismo da X-eNFN-SAE$_1$ selecionou x_{t-6}, x_{t-12} e x_{t-18} para compor a estrutura final da rede. A saída desejada e a saída estimada pela eNFN-SAE$_1$ na previsão da série de Mackey-Glass é apresentada na Figura 7.5.

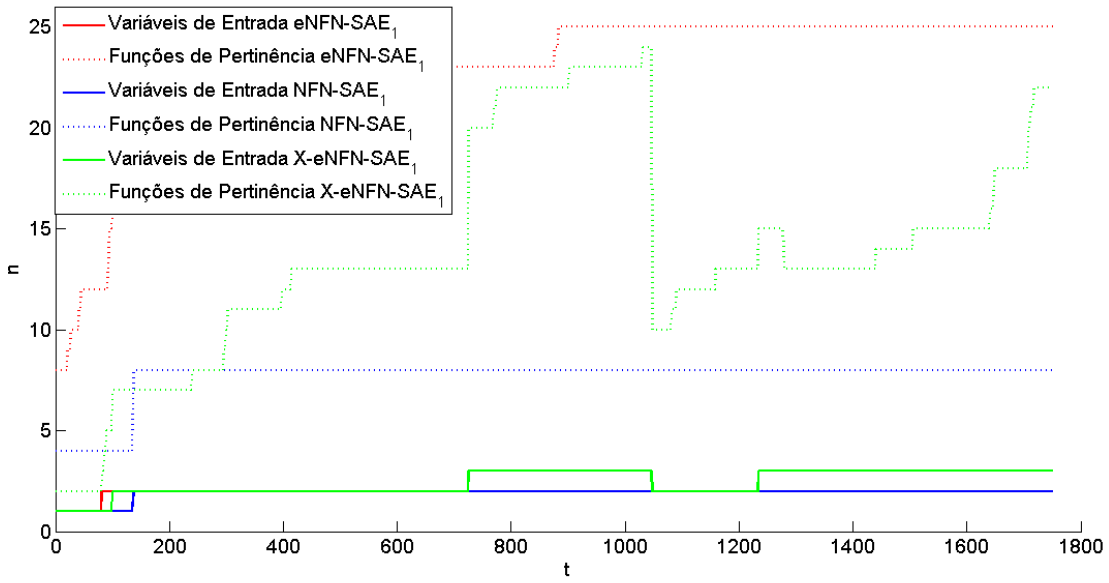

Fig. 7.4: Evolução da estrutura das redes com seleção adaptativa de entradas na previsão da série temporal de Mackey-Glass.

A Tabela 7.6 ilustra pelo RMSE os resultados obtidos. Estes sugerem que o melhor desempenho foi obtido pela NFN-SAE$_1$, seguida pelo DENFIS, eNFN-SAE$_1$, eNFN, X-eNFN-SAE$_1$ e eMG. Os resultados alcançados pela NFN-SAE$_1$, DENFIS, eNFN-SAE$_1$, eNFN, X-eNFN-SAE$_1$ e eMG são comparáveis e superam os obtidos pelo eTS e xTS em uma ordem de grandeza.

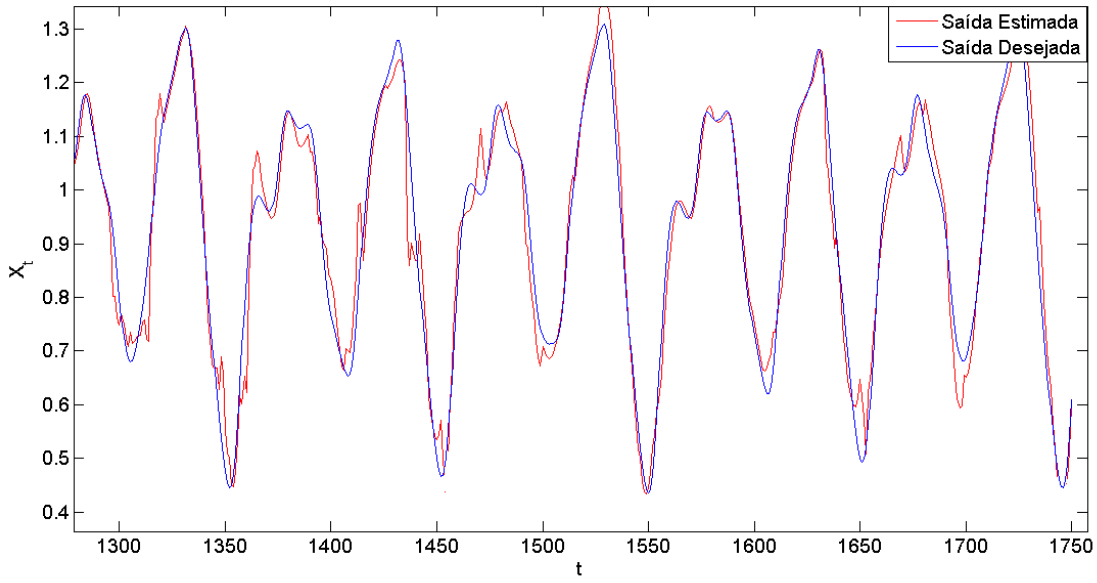

Fig. 7.5: Previsão da série temporal de Mackey-Glass.

O método de seleção de entradas possibilitou a construção de modelos compactos com duas e três variáveis de entrada e com um bom desempenho. Nesta simulação as redes com seleção adaptativa de entradas iniciadas com uma única variável de entrada superam as redes iniciadas com todas as quatro variáveis de entrada.

Tab. 7.6: Desempenho na previsão da série temporal de Mackey-Glass.

Modelo	N. Regras	RMSE
DENFIS	23	0.0425
eMG	07	0.0871
eNFN	21	0.0745
eNFN-SAE$_1$	11	0.0604
eTS	05	0.3739
NFN-SAE$_1$	08	0.0387
X-eNFN-SAE$_1$	22	0.0819
xTS	04	0.3750

O logaritmo da soma dos quadrados residuais ($Log\ (SQR)$) para cada iteração t é apresentado na Figura 7.6. Analisando a Figura 7.6, pode-se observar que o melhor desempenho foi obtido pela NFN-SAE$_1$, seguida pelo DENFIS e eNFN-SAE$_1$. A curva de erro da eNFN é comparável a da X-eNFN-SAE$_1$ e superior a do eMG. Comparando a curva de erro desses seis, percebe-se que estes tiveram um desempenho superior ao eTS e xTS.

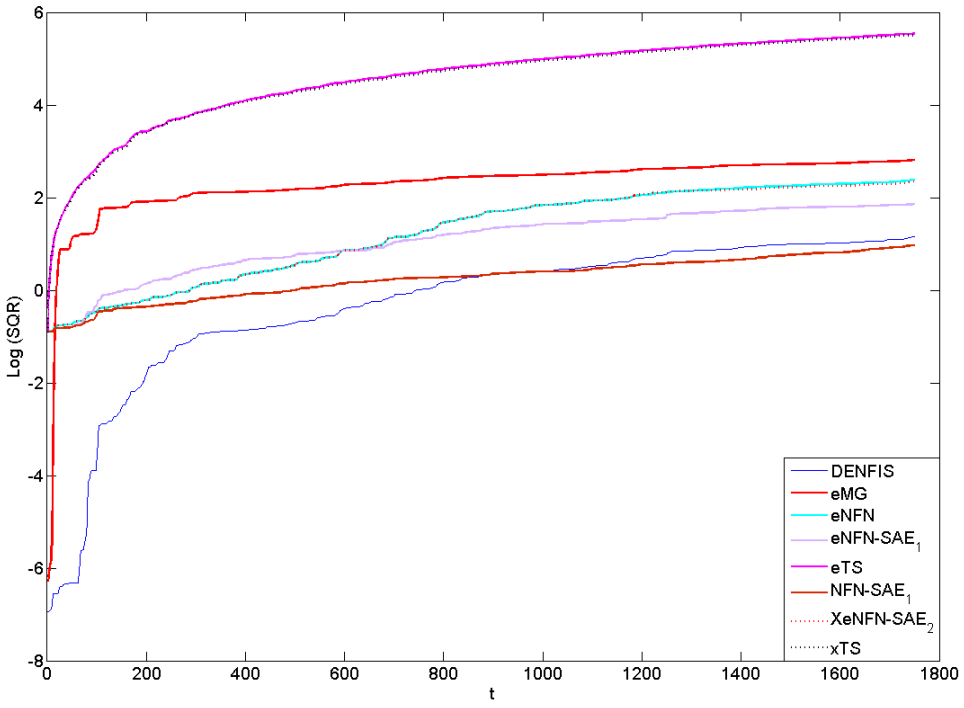

Fig. 7.6: Logaritmo da soma dos quadrados residuais na previsão da série temporal de Mackey-Glass.

O teste estatístico MGN para a rede eNFN-SAE$_1$ é ilustrado na Tabela 7.7. Os resultados do teste MGN sugerem, com um nível de significância de 0,05, que a eNFN-SAE$_1$ é estatisticamente superior ao eMG, eNFN, eTS, X-eNFN-SAE$_1$ e xTS. A comparação também mostra que o desempenho do DENFIS e da NFN-SAE$_1$ é superior ao da eNFN-SAE$_1$. A Tabela 7.8 ilustra as comparações entre o NFN-SAE$_1$ e os demais modelos analisados utilizando-se o teste MGN. Dado um nível de significância de 0,05, os resultados apresentados mostram que a NFN-SAE$_1$ foi estatisticamente superior aos demais modelos. Os resultados do comparativo da X-eNFN-SAE$_1$ são ilustrados na Tabela 7.9. Estes sugerem que a X-eNFN-SAE$_1$ é estatisticamente superior ao eMG, eTS e xTS, com um nível de significância de 0,05.

Tab. 7.7: Avaliação da eNFN-SAE$_1$ pelo teste MGN na previsão da série temporal de Mackey-Glass.

Modelo	MGN	p-valor
DENFIS vs eNFN-SAE$_1$	15.4772	0.0000
eNFN-SAE$_1$ vs eMG	21.2624	0.0000
eNFN-SAE$_1$ vs eNFN	12.2078	0.0000
eNFN-SAE$_1$ vs eTS	18.8215	0.0000
NFN-SAE$_1$ vs eNFN-SAE$_1$	21.3820	0.0000
eNFN-SAE$_1$ vs X-eNFN-SAE$_1$	13.6740	0.0000
eNFN-SAE$_1$ vs xTS	17.8088	0.0000

Tab. 7.8: Avaliação da NFN-SAE$_1$ pelo teste MGN na previsão da série temporal de Mackey-Glass.

Modelo	MGN	p-valor
NFN-SAE$_1$ vs DENFIS	3.9974	0.0000
NFN-SAE$_1$ vs eMG	45.8040	0.0000
NFN-SAE$_1$ vs eNFN	33.9626	0.0000
NFN-SAE$_1$ vs eNFN-SAE$_1$	21.3820	0.0000
NFN-SAE$_1$ vs eTS	42.0571	0.0000
NFN-SAE$_1$ vs X-eNFN-SAE$_1$	36.4279	0.0000
NFN-SAE$_1$ vs xTS	40.4698	0.0000

7.4 Previsão de Temperatura

Nesta seção, avaliam-se os modelos na previsão de temperatura. O problema de previsão de temperatura considera o fluxo de dados derivado das temperaturas médias mensais de três regiões geográficas com padrões climáticos distintos. Uma região com clima extremamente seco

Tab. 7.9: Avaliação da X-eNFN-SAE$_1$ pelo teste MGN na previsão da série temporal de Mackey-Glass.

Modelo	MGN	p-valor
DENFIS vs X-eNFN-SAE$_1$	31.4925	0.0000
X-eNFN-SAE$_1$ vs eMG	7.5024	0.0000
eNFN vs X-eNFN-SAE$_1$	1.8448	0.0326
eNFN-SAE$_1$ vs X-eNFN-SAE$_1$	13.6740	0.0000
X-eNFN-SAE$_1$ vs eTS	5.1681	0.0000
NFN-SAE$_1$ vs X-eNFN-SAE$_1$	36.4279	0.0000
X-eNFN-SAE$_1$ vs xTS	4.2367	0.0000

e com elevadas temperaturas (Vale da Morte), uma região com clima frio (Ottawa) e uma região em que é observada uma vasta gama de temperaturas durante o ano, variando de um inverno muito frio e com neve a um verão quente (Lisboa). Os dados obtidos compreendem as temperaturas médias mensais registradas no período de janeiro de 1901, 1895 e 1910, respectivamente, até dezembro de 2009. O modelo visa à previsão da temperatura média mensal um passo à frente. Trabalhos anteriores sugerem o uso de cinco primeiros valores defasados da série como entradas (Leite et al., 2012; Silva et al., 2012a). O modelo é descrito por:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-5}). \tag{7.3}$$

O conjunto de dados consiste de 3870 observações[4], 1302 do Vale da Morte, 1374 de Ottawa, e 1994 de Lisboa. O conjunto de dados é dividido em dois subconjuntos com 1935 observações cada. O primeiro é utilizado para estimar os parâmetros e o segundo, para avaliar o desempenho. Os modelos são avaliados pelo RMSE, SQR e MGN.

Os parâmetros dos algoritmos são definidos como se segue: DENFIS ($dthr = 0.07$, $mofn = 3$); eMG ($\lambda = 0.05$, $w = 10$, $\sum_{init} = 10^{-2}.I_5$, $\alpha = 0.01$); eNFN ($\beta = 0.01$, $w = 100$, $\eta = 10$); eNFN-SAE$_2$ ($\beta = 0.01$, $w = 100$, $\lambda = 0.01$, $\eta = 10$); eTS ($r = 0.06$, $\Omega = 550$); NFN-SAE$_2$ ($m = 4$, $\beta = 0.01$, $\lambda = 0.01$); X-eNFN-SAE$_2$ ($\beta = 0.01$, $w = 100$, $\lambda = 0.01$, $\eta = 10$); xTS ($\Omega = 650$). As redes eNFN-SAE$_2$, NFN-SAE$_2$ e X-eNFN-SAE$_2$ iniciam com todas as cinco variáveis de entrada, enquanto DENFIS, eMG, eNFN, eTS e xTS iniciam e mantêm todas as cinco variáveis de entrada.

A Figura 7.7 mostra como a estrutura das redes com seleção adaptativa de entradas evolui a cada amostra t do conjunto de dados. O mecanismo de seleção de entradas manteve as cinco variáveis de entrada para as redes eNFN-SAE$_2$, NFN-SAE$_2$ e X-eNFN-SAE$_2$. Aproximadamente entre $t = 835$ e $t = 850$ (Figura 7.7), o número de funções de pertinência (regras) das redes eNFN-SAE$_2$ e X-eNFN-SAE$_2$ é reduzido drasticamente. Isto mostra que a rede adapta-se rapidamente diante de mudanças nos dados de entrada. Nota-se que as redes com seleção adaptativa de entradas iniciaram e mantiveram as cinco variáveis de entrada. Neste caso, a evolução

[4]O conjunto de dados assume as observações das temperatura das três séries (Vale da Morte, Ottawa, e Lisboa) em sequência.

da estrutura das redes eNFN-SAE$_2$ e X-eNFN-SAE$_2$ ocorreu pela inclusão/exclusão de funções de pertinência. A Figura 7.8 mostra a saída desejada e a saída da rede X-eNFN-SAE$_2$.

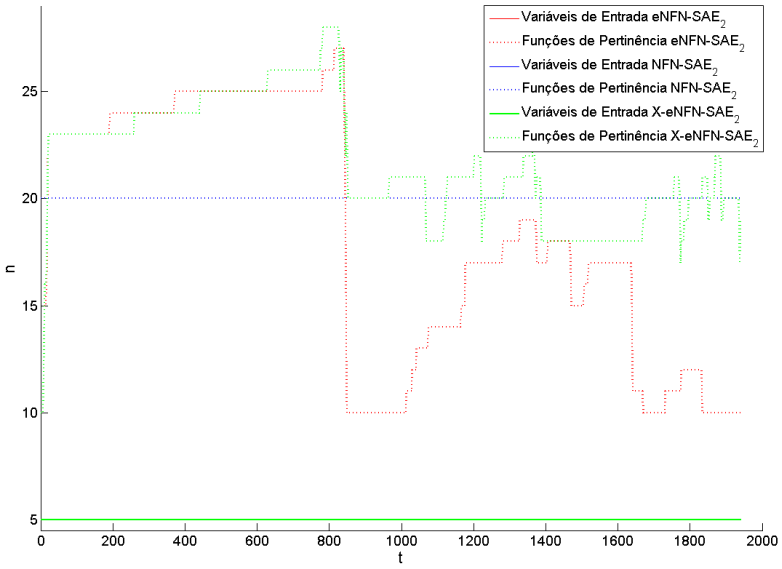

Fig. 7.7: Evolução da estrutura das redes com seleção adaptativa de entradas na previsão de temperatura.

A Tabela 7.10 sumariza pelo RMSE o desempenho dos modelos. Ela sugere que as redes eNFN e X-eNFN-SAE$_2$ alcançaram o mesmo desempenho, superando eNFN-SAE$_2$, NFN-SAE$_2$, xTS e eTS. O melhores desempenhos foram obtidos pelo DENFIS e eMG. Na previsão de temperatura, as redes com seleção de entradas obtiveram um melhor desempenho quando iniciadas com todas as variáveis de entrada.

O logaritmo da soma dos quadrados residuais ($Log\ (SQR)$) na previsão de temperatura é apresentado na Figura 7.9. Os resultados visualizados na Figura 7.9 sugerem que as redes propostas (X-eNFN-SAE$_2$, eNFN, eNFN-SAE$_2$, NFN-SAE$_2$) são comparáveis e superam o desempenho do eTS e xTS. Os melhores resultados foram obtidos pelo DENFIS e eMG.

Os resultados da comparação entre as redes com seleção adaptativa de entradas e os outros modelos evolutivos pelo teste MGN na previsão de temperatura são apresentados nas tabelas 7.11, 7.12 e 7.13. As tabelas mostram os valores da estatística do teste MGN e o p-valor correspondente. Os resultados apresentados pelo teste MGN corroboram os resultados obtidos pelas medidas de erro. A Tabela 7.11 mostra que a rede eNFN-SAE$_2$ é estatisticamente superior ao eTS e xTS. A Tabela 7.12 sugere que a NFN-SAE$_2$ possui um desempenho estatístico similar a eNFN e X-eNFN-SAE$_2$, superior ao eNFN-SAE$_2$, eTS e xTS. A Tabela 7.13 mostra que as redes X-eNFN-SAE$_2$ e eNFN obtiveram o mesmo desempenho e que os resultados dessas duas redes superam os obtidos por eNFN-SAE$_2$, eTS e xTS. Os resultados apresentados consideram um nível de significância de 0,05.

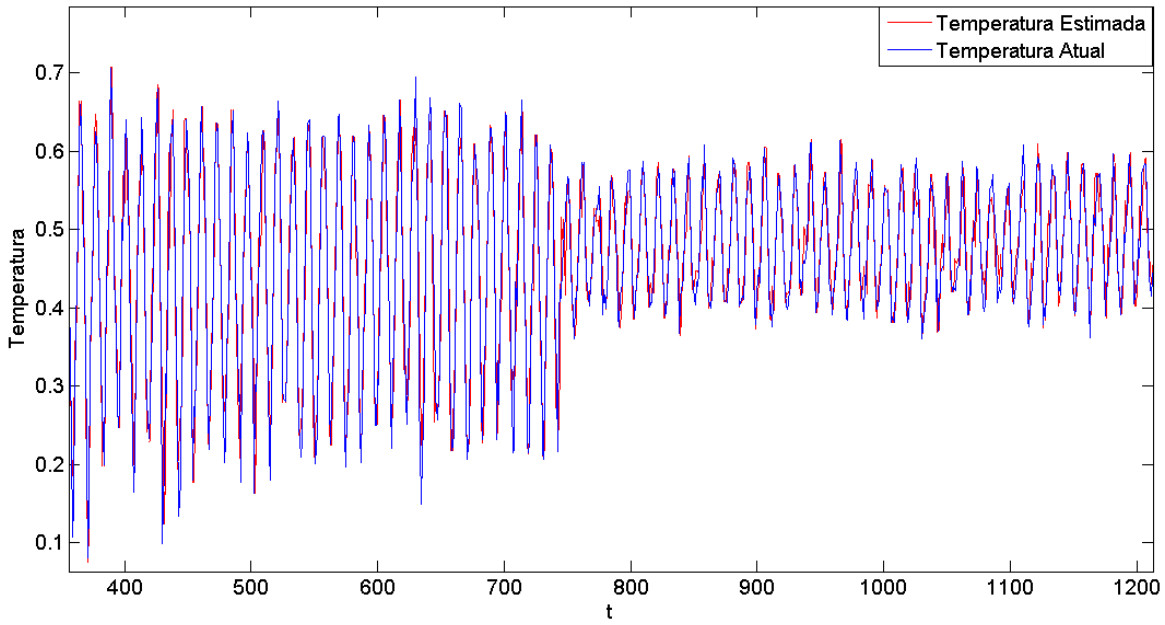

Fig. 7.8: Previsão de temperatura.

Tab. 7.10: Desempenho na previsão de temperatura.

Modelo	N. Regras	RMSE
DENFIS	52	0.0327
eMG	03	0.0377
eNFN	19	0.0438
eNFN-SAE$_2$	10	0.0461
eTS	15	0.1675
NFN-SAE$_2$	20	0.0501
X-eNFN-SAE$_2$	19	0.0438
xTS	17	0.1687

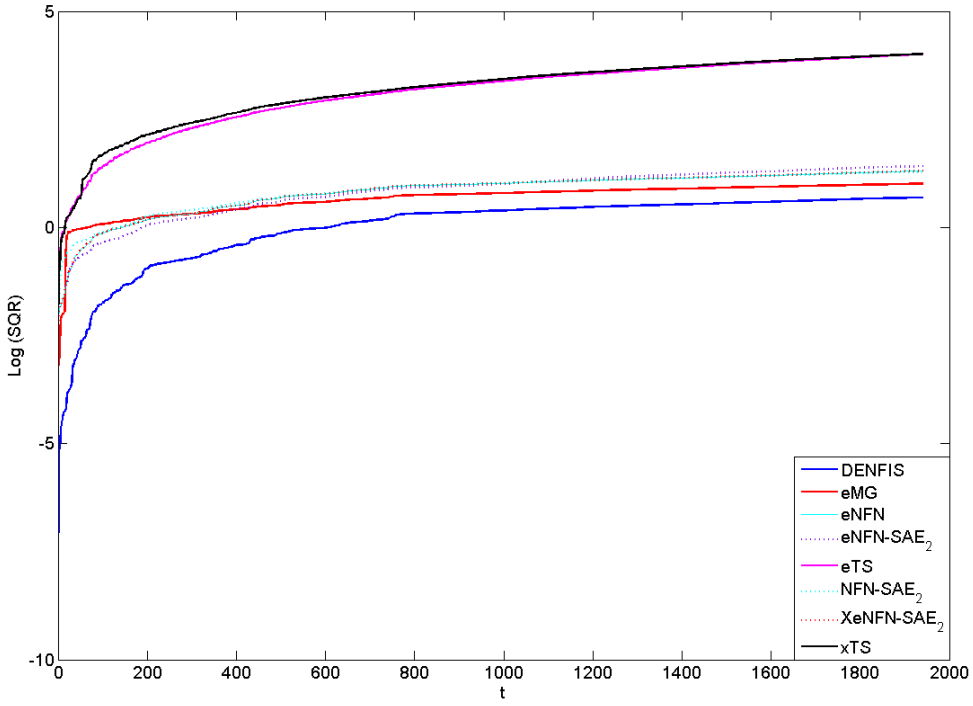

Fig. 7.9: Logaritmo da soma dos quadrados residuais na previsão de temperatura.

Tab. 7.11: Avaliação da eNFN-SAE$_2$ pelo teste MGN na previsão de temperatura.

Modelo	MGN	p-valor
DENFIS vs eNFN-SAE$_2$	20.5161	0.0000
eMG vs eNFN-SAE$_2$	11.2902	0.0000
eNFN vs eNFN-SAE$_2$	4.1097	0.0000
eNFN-SAE$_2$ vs eTS	25.6088	0.0000
NFN-SAE$_2$ vs eNFN-SAE$_2$	4.2693	0.0000
X-eNFN-SAE$_2$ vs eNFN-SAE$_2$	4.1097	0.0000
eNFN-SAE$_2$ vs xTS	28.6404	0.0000

Tab. 7.12: Avaliação da NFN-SAE$_2$ pelo teste MGN na previsão de temperatura.

Modelo	MGN	p-valor
DENFIS vs NFN-SAE$_2$	17.0126	0.0000
eMG vs NFN-SAE$_2$	7.9588	0.0000
NFN-SAE$_2$ vs eNFN	0.9220	0.1783
NFN-SAE$_2$ vs eNFN-SAE$_2$	4.2693	0.0000
NFN-SAE$_2$ vs eTS	29.5316	0.0000
X-eNFN-SAE$_2$ vs NFN-SAE$_2$	0.9220	0.1783
NFN-SAE$_2$ vs xTS	32.2996	0.0000

Tab. 7.13: Avaliação da X-eNFN-SAE$_2$ pelo teste MGN na previsão de temperatura.

Modelo	MGN	p-valor
DENFIS vc X-eNFN-SAE$_2$	17.9565	0.0000
eMG vs X-eNFN-SAE$_2$	8.6082	0.0000
X-eNFN-SAE$_2$ vs eNFN	0.0000	0.0000
X-eNFN-SAE$_2$ vs eNFN-SAE$_2$	4.1097	0.0000
X-eNFN-SAE$_2$ vs eTS	28.7017	0.0000
X-eNFN-SAE$_2$ vs NFN-SAE$_2$	0.9220	0.1783
X-eNFN-SAE$_2$ vs xTS	31.8708	0.0000

7.5 Análise do Tempo de Processamento

Esta seção apresenta a avaliação do tempo de processamento das redes com seleção adaptativa de entradas e dos modelos alternativos abordados neste capítulo. O tempo de processamento se refere às simulações computacionais descritas nas seções 7.2, 7.3 e 7.4. Os algoritmos são inicializados com os mesmos parâmetros definidos nessas seções.

Tabela 7.14 ilustra o tempo de processamento na identificação de processo não linear, a Tabela 7.15 na previsão da série temporal de Mackey-Glass, e a Tabela 7.16 na previsão de temperatura. Os tempos de processamento são expressos em milissegundos (ms) e se referem ao tempo médio e ao desvio padrão para processamento de cada amostra do conjunto de dados, calculados a partir de 10 simulações. Nos resultados apresentados nas seções 7.2, 7.3 e 7.4, optou-se por apresentar somente as configurações de rede que obtiveram melhor desempenho. Nesta seção, apresenta-se tanto as redes iniciadas com uma única variável de entrada quanto as iniciadas com todas as variáveis de entrada. As redes eNFN-SAE$_1$, NFN-SAE$_1$ e X-eNFN-SAE$_1$ foram iniciadas com uma variável de entrada, enquanto as redes eNFN-SAE$_2$, NFN-SAE$_2$ e X-eNFN-SAE$_2$ foram iniciadas com todas as variáveis de entrada disponíveis.

Tab. 7.14: Tempo de processamento na identificação de processo não linear.

Modelo	Tempo (ms)	Desvio Padrão
DENFIS	2.0330	0.0265
eMG	3.5779	0.0117
eNFN	0.4751	0.0275
eNFN-SAE$_1$	1.0499	0.0246
eNFN-SAE$_2$	1.7630	0.0252
eTS	1.1055	0.0386
NFN-SAE$_1$	0.9035	0.0245
NFN-SAE$_2$	1.0078	0.0305
X-eNFN-SAE$_1$	3.5398	0.0317
X-eNFN-SAE$_2$	3.2176	0.0237
xTS	1.1798	0.0325

Analisando-se o tempo de execução dos algoritmos apresentados nas tabelas 7.14, 7.15 e 7.16, verifica-se que:

- A rede eNFN possui o menor tempo de processamento.
- As redes com seleção adaptativa de entradas tem um menor tempo de processamento quando iniciadas com uma única variável de entrada.
- O tempo de processamento das redes NFN-SAE foram o segundo e terceiro melhor em dois dos experimentos. No experimento de previsão da série temporal de Mackey-Glass, as redes NFN-SAE obtiveram o quarto e o quinto melhor desempenho.

Tab. 7.15: Tempo de processamento na previsão da série temporal de Mackey-Glass.

Modelo	Tempo (ms)	Desvio Padrão
DENFIS	6.0275	0.0245
eMG	3.3908	0.0458
eNFN	0.5607	0.0365
eNFN-SAE$_1$	2.7250	0.0335
eNFN-SAE$_2$	3.1270	0.0242
eTS	1.8574	0.0218
NFN-SAE$_1$	1.9411	0.0258
NFN-SAE$_2$	1.9778	0.0298
X-eNFN-SAE$_1$	4.7130	0.0330
X-eNFN-SAE$_2$	6.8302	0.0322
xTS	1.5416	0.0167

Tab. 7.16: Tempo de processamento na previsão de temperatura.

Modelo	Tempo (ms)	Desvio Padrão
DENFIS	1.9456	0.0232
eMG	1.5793	0.0136
eNFN	0.2936	0.0147
eNFN-SAE$_1$	1.9984	0.0241
eNFN-SAE$_2$	2.2542	0.0230
eTS	1.1640	0.0118
NFN-SAE$_1$	1.0412	0.0175
NFN-SAE$_2$	1.1474	0.0232
X-eNFN-SAE$_1$	2.9630	0.0242
X-eNFN-SAE$_2$	3.2017	0.0248
xTS	1.2202	0.0300

- As redes eNFN-SAE obtiveram tempo de processamento compatível com o dos modelos evolutivos alternativos.

- As redes X-eNFN-SAE obtiveram um dos piores desempenhos no tempo de processamento. O elevado custo de processamento desta rede se deve à necessidade de se calcular o grau de ativação das funções de pertinência, as saídas, e atualizar a estrutura da rede de forma independente para cada um dos modelos candidatos e para o modelo atual.

É importante ressaltar que a redes NFN-SAE, eNFN-SAE e X-eNFN-SAE são as únicas que incorporam métodos de seleção adaptativa de entradas em seu algoritmo de aprendizado. Portanto, é aceitável que estas possuam tempo de processamento maior do que o dos outros modelos evolutivos. Mesmo neste cenário desfavorável, as redes NFN-SAE e eNFN-SAE obtiveram tempo de processamento comparável ao do DENFIS, eMG, eTS e xTS.

7.6 Relação entre Dimensão do Espaço de Entrada/Saída e Tempo de Processamento

Esta seção avalia a escalabilidade das redes com seleção adaptativa de entradas. Avalia-se a relação entre tempo de processamento e o aumento no número de funções de pertinência e variáveis de entrada. Espera-se que o aumento no número de funções de pertinência não tenha impacto no tempo de processamento das redes. Outro fator a ser avaliado é o crescimento do tempo de processamento das redes em função do aumento da dimensão do espaço de entrada. É desejado que as redes tenham um crescimento linear no tempo de processamento com relação ao aumento no número de variáveis de entrada. Nos experimentos, utilizaram-se os mesmos conjuntos de dados descritos na Seção 5.4. Os conjuntos de dados possuem 3500 amostras e o número de variáveis de entrada é n, $n = 2, \ldots, 15$.

A Figura 7.10 ilustra a escalabilidade[5] das redes NFN-SAE. Para as redes NFN-SAE utilizaram-se quatro configurações como se segue: NFN-SAE$_1$ uma variável de entrada e $m = 7$; NFN-SAE$_2$ uma variável de entrada e $m = 14$; NFN-SAE$_3$ todas as variáveis de entrada e $m = 7$; NFN-SAE$_4$ todas as variáveis de entrada e $m = 14$. A Tabela 7.17 ilustra o número de variáveis de entrada selecionadas por cada uma das redes NFN-SAE para cada dimensão do espaço de entrada.

Analisando a escalabilidade das redes NFN-SAE temos duas situações distintas. As redes NFN-SAE$_1$ e NFN-SAE$_2$ que foram iniciadas com uma única variável de entrada se comportaram conforme o esperado, isto é, o crescimento do tempo de processamento é linear. Porém, nas redes NFN-SAE$_3$ e NFN-SAE$_4$ (iniciadas com todas as variáveis de entrada) o crescimento do tempo tende para o quadrático. Nota-se também que o número de funções de pertinência não impacta no tempo de processamento.

As figuras 7.11 e 7.12 comparam o número de regras e o tempo de processamento das redes eNFN-SAE e X-eNFN-SAE, respectivamente. As redes foram iniciadas com as seguintes configurações: eNFN-SAE$_1$ e X-eNFN-SAE$_1$ uma variável de entrada e $\eta = 10$; eNFN-SAE$_2$

[5]Nas figuras 7.10, 7.11 e 7.12 as retas indicam o crescimento linear do tempo de processamento baseado no tempo de processamento das redes com duas variáveis de entrada.

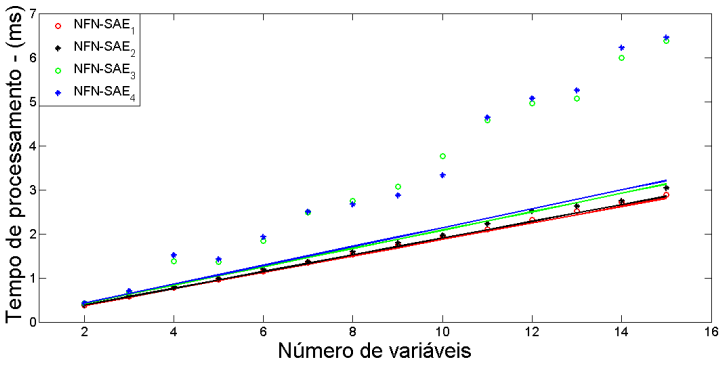

Fig. 7.10: Escalabilidade das redes NFN-SAE.

e X-eNFN-SAE$_2$ uma variável de entrada e $\eta = 20$; eNFN-SAE$_3$ e X-eNFN-SAE$_3$ todas as variáveis de entrada e $\eta = 10$; eNFN-SAE$_4$ e X-eNFN-SAE$_4$ todas as variáveis de entrada e $\eta = 20$. As Tabelas 7.18 e 7.19 mostram o número de regras e número de variáveis de entrada selecionados para cada uma das redes.

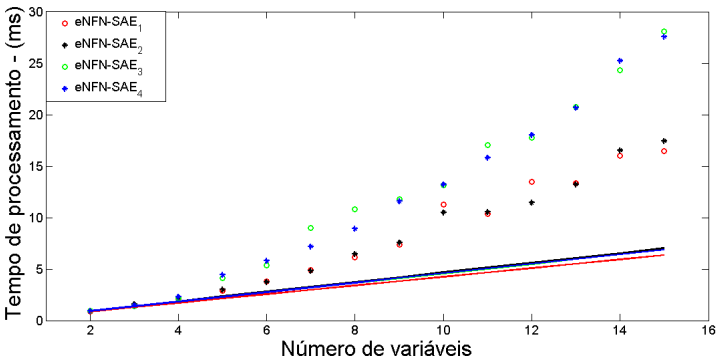

Fig. 7.11: Escalabilidade das redes eNFN-SAE.

A Figura 7.11 ilustra o crescimento do tempo de processamento para a rede eNFN-SAE com relação ao número de variáveis de entrada e o número de regras. Nesta figura pode-se perceber que as redes iniciadas com uma única variável de entrada possuem um menor tempo de processamento. Percebe-se também que o número de regras não tem impacto significativo no tempo de processamento da rede e que o maior impacto no tempo de processamento é o aumento no número de variáveis de entrada.

Analisando os resultados apresentados pela rede X-eNFN-SAE (Figura 7.12) pode-se perceber que as redes iniciadas com uma única variável de entrada possuem um crescimento linear em relação ao número de variáveis de entrada, o que não acontece com as redes iniciadas com todas as variáveis de entrada disponíveis. Observa-se que o número de funções de pertinência não tem impacto significativo no tempo de processamento. O maior impacto no tempo de processamento dessas redes se dá com o aumento no número de variáveis de entrada, o que já era

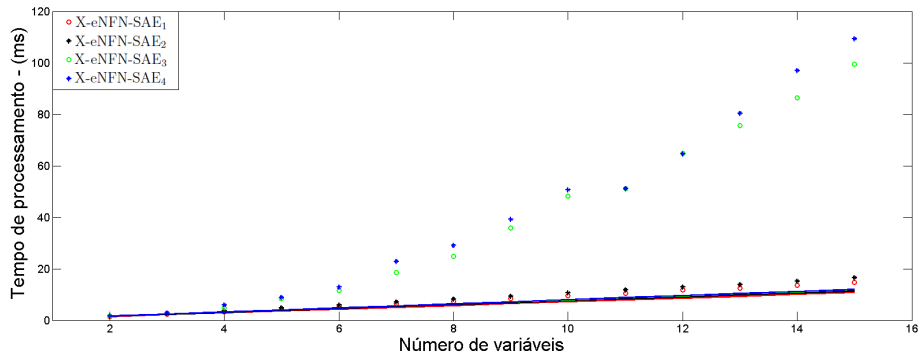

Fig. 7.12: Escalabilidade das redes X-eNFN-SAE.

esperado, pois com o aumento no número de variáveis de entrada ocorre também ao aumento no número de modelos candidatos, e a rede deve calcular o grau de ativação das funções de pertinência, evoluir a estrutura, e calcular a saída de maneira independente para o modelo atual e para cada um dos modelos candidatos.

Tab. 7.17: Regras e variáveis de entrada por dimensão do espaço de entrada - NFN-SAE.

N. Var.	NFN-SAE$_1$ Var. - Regras	NFN-SAE$_2$ Var. - Regras	NFN-SAE$_3$ Var. - Regras	NFN-SAE$_4$ Var. - Regras
02	1 - 7	1 - 14	1 - 7	1 - 14
03	1 - 7	1 - 14	1 - 7	1 - 14
04	1 - 7	1 - 14	1 - 7	3 - 42
05	1 - 7	1 - 14	3 - 21	3 - 42
06	1 - 7	1 - 14	4 - 28	4 - 56
07	1 - 7	1 - 14	5 - 35	5 - 70
08	1 - 7	1 - 14	6 - 42	7 - 98
09	1 - 7	1 - 14	8 - 56	8 - 112
10	1 - 7	1 - 14	9 - 63	9 - 126
11	1 - 7	1 - 14	10 - 70	10 - 140
12	1 - 7	1 - 14	10 - 70	11 - 154
13	1 - 7	1 - 14	11 - 77	12 - 168
14	1 - 7	1 - 14	12 - 84	13 - 182
15	1 - 7	1 - 14	13 - 91	14 - 196

Tab. 7.18: Regras e variáveis de entrada por dimensão do espaço de entrada - eNFN-SAE.

N. Var.	eNFN-SAE$_1$ Var. - Regras	eNFN-SAE$_2$ Var. - Regras	eNFN-SAE$_3$ Var. - Regras	eNFN-SAE$_4$ Var. - Regras
02	1 - 10	1 - 16	1 - 10	1 - 16
03	1 - 9	1 - 16	1 - 10	1 - 14
04	1 - 9	1 - 15	1 - 8	1 - 10
05	1 - 9	1 - 15	3 - 30	3 - 42
06	1 - 9	1 - 15	4 - 40	4 - 59
07	1 - 10	1 - 9	5 - 49	6 - 93
08	1 - 10	1 - 9	7 - 67	7 - 106
09	1 - 10	1 - 9	8 - 80	9 - 139
10	1 - 10	1 - 9	10 - 98	10 - 138
11	2 - 20	2 - 27	10 - 100	10 - 141
12	1 - 9	1 - 16	12 - 118	11 - 173
13	1 - 9	1 - 16	13 - 132	13 - 209
14	1 - 9	1 - 16	14 - 136	14 - 212
15	1 - 9	1 - 16	15 - 147	15 - 218

Tab. 7.19: Regras e variáveis de entrada por dimensão do espaço de entrada - X-eNFN-SAE.

N. Var.	X-eNFN-SAE$_1$ Var. - Regras	X-eNFN-SAE$_2$ Var. - Regras	X-eNFN-SAE$_3$ Var. - Regras	X-eNFN-SAE$_4$ Var. - Regras
02	1 - 9	1 - 7	1 - 9	1 - 15
03	1 - 8	1 - 9	1 - 5	1 - 6
04	1 - 5	1 - 12	1 - 5	1 - 6
05	1 - 5	1 - 7	2 - 16	2 - 21
06	1 - 5	1 - 6	3 - 26	3 - 46
07	1 - 5	1 - 6	5 - 42	5 - 68
08	1 - 5	1 - 6	6 - 55	6 - 68
09	1 - 5	1 - 6	7 - 64	7 - 92
10	3 - 27	3 - 44	8 - 73	8 - 106
11	1 - 8	1 - 13	9 - 84	9 - 127
12	1 - 5	1 - 6	10 - 82	10 - 125
13	1 - 5	1 - 11	11 - 103	11 - 120
14	1 - 5	1 - 8	12 - 110	12 - 130
15	1 - 5	1 - 6	13 - 116	13 - 154

7.7 Resumo

Neste capítulo, foram realizadas simulações computacionais na identificação de sistema não linear e em problemas de previsão para avaliar o desempenho das redes com seleção adaptativa de entradas. Os resultados obtidos pelas redes com seleção de entradas são comparáveis aos dos demais modelos evolutivos. Na análise do tempo de processamento dos algoritmos, a rede eNFN obteve o menor tempo de processamento. As redes com seleção de entradas e os demais modelos possuem um tempo de processamento comparável.

Conclui-se que as redes com seleção adaptativa de entradas possuem uma acurácia e um tempo de processamento comparáveis aos modelos evolutivos. Os resultados obtidos sugerem as redes com seleção adaptativa como promissoras abordagens no desenvolvimento de sistemas *fuzzy* com alto grau de adaptabilidade.

Capítulo 8

Conclusão

O desafio na modelagem de sistemas evolutivos está na construção de modelos com alto grau de adaptabilidade, autonomia e flexibilidade. Estes são alcançados pela modificação da estrutura e dos parâmetros livres dos modelos de acordo com alterações nos dados de entrada. Diante deste cenário, este trabalho propôs novos algoritmos de aprendizado para sistemas *fuzzy* evolutivos com o objetivo de desenvolver modelos com baixo custo computacional, altamente flexíveis, parcimoniosos, precisos e que possam ser implementados em sistemas de tempo real. Os algoritmos evolutivos propostos neste trabalho foram construídos sob a estrutura da rede *Neo-Fuzzy-Neuron* (NFN).

Inicialmente foi introduzido um algoritmo evolutivo para o NFN. O algoritmo evolutivo proposto (eNFN) difere-se do original (NFN) por possibilitar a modificação de sua estrutura durante o processo de aprendizagem. A estrutura do eNFN evolui iterativamente de acordo com o fluxo de dados, permitindo, a cada nova amostra apresentada ao modelo, a alteração dos parâmetros, a inserção, a exclusão, ou a permanência de funções de pertinência. As funcionalidades propostas têm por objetivo manter um erro mínimo e uniforme para todas as regiões do espaço de entrada, além de permitir a adaptação dos parâmetros e estrutura do modelo, a fim de lidar com as mudanças nas condições de operação.

O desempenho do eNFN foi avaliado e comparado com outros modelos evolutivos na identificação de sistema não linear e em problemas de previsão de séries temporais. Os resultados alcançados pelo eNFN são comparáveis aos dos modelos evolutivos alternativos. Avaliou-se a complexidade computacional dos modelos por meio da ordem de complexidade e do tempo de processamento. O eNFN possui uma baixa complexidade computacional se comparado com os modelos evolutivos analisados. Além de possuir uma menor ordem de complexidade, o eNFN possui um tempo de processamento cerca de dois terços menor do que o dos outros modelos evolutivos. A boa precisão, as equações simples e a baixa complexidade credenciam a rede eNFN para aplicações em sistemas de tempo real com dinâmica rápida.

Foram realizados experimentos computacionais com o eNFN em problemas práticos em tempo real com alta taxa de amostragem. Os experimentos foram realizados na estimação de variáveis de processos de um sistema de levitação magnética (MagLev) e de um sistema MIMO de duplo rotor (TRMS). Os resultados experimentais sugerem que a eNFN é efetiva para estimar variáveis de estado de processos com dinâmica rápida em tempo real. Evidentemente, a rede também pode ser utilizada para processos lentos. Além disso, os experimentos mostram que

a estrutura da eNFN evolui de acordo com mudanças no modo de operação dos sistemas, adequando a rede para manter uma estrutura compacta e um erro pequeno.

Este trabalho ainda propôs uma nova abordagem para o desenvolvimento de modelos *fuzzy* evolutivos com seleção adaptativa de entradas. A abordagem foi desenvolvida considerando-se a estrutura do NFN e um teste estatístico para selecionar adaptativamente as variáveis de entrada. O teste estatístico empregado considera a precisão e o número de parâmetros dos modelos. Com base na abordagem para seleção adaptativa de entradas foram propostos três algoritmos.

O primeiro, chamado de NFN-SAE, utiliza a estrutura da rede NFN e o método de seleção adaptativa de entradas. Neste algoritmo, o número de funções de pertinência é fixo e determinado *a priori*, e a evolução da estrutura se dá pela inclusão e/ou exclusão de variáveis de entrada. O segundo (eNFN-SAE) e o terceiro (X-eNFN-SAE) foram desenvolvidos utilizando-se o método para evolução da estrutura proposto no eNFN e o procedimento para seleção adaptativa de entradas. Nestes dois algoritmos, a estrutura evolui pela inclusão/exclusão de variáveis de entrada e também pela inserção/eliminação de funções de pertinência. No algoritmo do eNFN-SAE, o cálculo do grau de ativação das funções de pertinência e a evolução da estrutura são realizados somente uma vez para todos os modelos (candidatos e atual), enquanto no algoritmo do X-eNFN-SAE esse processo é realizado de maneira independente para cada modelo.

Os algoritmos com seleção adaptativa de entradas foram avaliados em problemas de identificação e previsão. Simulações computacionais mostram que as abordagens propostas e os modelos resultantes atingem um desempenho comparável ao de modelos evolutivos que estão no estado da arte. A análise da complexidade computacional mostra que as redes com seleção adaptativa de entradas e os modelos evolutivos alternativos possuem um custo computacional comparável, sendo superados pelo custo computacional da rede eNFN. Os resultados sugerem as redes com seleção adaptativa como promissoras abordagens no desenvolvimento de sistemas *fuzzy* com alto grau de adaptabilidade e autonomia.

Sugere-se utilizar o eNFN em problemas nos quais as variáveis de entradas são conhecidas e o tempo de processamento é uma prioridade, como em aplicações em tempo real com dinâmica rápida. A rede NFN-SAE possui o menor tempo de processamento entre as redes com seleção adaptativa de entradas, portanto recomenda-se seu uso em aplicações em que o tempo de processamento é um fator relevante e que não se tenha conhecimento *a priori* do sistema para definição das variáveis de entrada. Em aplicações nas quais não se tem nenhum conhecimento *a priori* do sistema e a taxa de amostragem é baixa, sugere-se utilizar a eNFN-SAE ou X-eNFN-SAE. A eNFN-SAE possui um menor custo computacional que a X-eNFN-SAE, mas em compensação a X-eNFN-SAE tem uma maior independência na evolução da estrutura da rede.

Como propostas de continuidade para este trabalho sugere-se:

- Reavaliar o critério de refinamento do espaço de entrada (inclusão e exclusão de funções de pertinência) para o eNFN (proposto no Capítulo 4). Adotou-se, como critério para inclusão de funções de pertinência, a soma erro médio do modelo com a variância do modelo. Outros métodos foram avaliados como, por exemplo, o uso da estatística de uma distribuição t de *Student* para amostras de tamanhos e variâncias diferentes. Com o uso dessa estatística, o modelo obteve boa precisão, porém, seu custo computacional tornou

seu uso proibitivo. Portanto, as estatísticas para esta análise, além de proporcionarem uma boa precisão ao modelo, devem ter um baixo custo computacional. Uma opção para granularização do domínio de entrada é a abordagem proposta por Wang et al. (2010) (descrita sucintamente na Seção 2.2.1).

- Avaliar métodos de análise da dependência entre as variáveis de entrada e verificar sua aplicabilidade nos algoritmos propostos. A dependência entre as variáveis pode ser implementada por meio do produto tensor entre os conjuntos *fuzzy* de duas ou mais variáveis de entrada (Bossley, 1995; Harris et al., 1996; Shahin et al., 2003).

- Reavaliar o método para seleção adaptativa de entradas a fim de reduzir seu custo computacional. Uma possível abordagem seria a atribuição de pesos para indicar a relevância de cada uma das variáveis de entrada, por exemplo, como proposto em Katakis et al. (2006) e Lughofer (2011).

- Avaliar o uso da abordagem proposta para evolução da estrutura do NFN em outras redes. Uma possibilidade seria o uso da abordagem proposta para construir uma versão adaptativa/evolutiva do controlador baseado em *Look-up Table* (Filev and Ying, 2013).

- Desenvolver novas técnicas de controle em tempo real para os módulos de levitação magnética e duplo rotor, ou para outros processos não lineares com parâmetros variantes no tempo, a partir dos algoritmos propostos neste trabalho. Esse controlador pode ser implementado por meio de um controle preditivo ou controle baseado na dinâmica inversa.

- Analisar os parâmetros iniciais dos algoritmos propostos que são escolhidos manualmente e propor mecanismos para ajuste automático destes parâmetros. Assim, proporcionando uma maior autonomia aos algoritmos.

Referências Bibliográficas

Allen, M. (1997). *Understanding Regression Analysis*. Springer, 1 edition.

Angelov, P. (2011). Fuzzily connected multimodel systems evolving autonomously from data streams. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(4):898–910.

Angelov, P. and Buswell, R. (2001). Evolving rule-based models: A tool for intelligent adaptation. In *Proceedings of the Joint IFSA World Congress and NAFIPS International Conference*, volume 2, pages 1062 – 1067.

Angelov, P. and Buswell, R. (2002). Identification of evolving fuzzy rule-based models. *IEEE Transactions on Fuzzy Systems*, 10(5):667 – 677.

Angelov, P. and Filev, D. (2004). An approach to online identification of Takagi-Sugeno fuzzy models. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 34(1):484 – 498.

Angelov, P. and Filev, D. (2005). Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '05*, pages 1068 – 1073.

Angelov, P., Filev, D., and Kasabov, N. (2008a). Guest editorial evolving fuzzy systems: Preface to the special section. *IEEE Transactions on Fuzzy Systems*, 16(6):1390 – 1392.

Angelov, P., Filev, D., and Kasabov, N. (2010). *Evolving Intelligent Systems: Methodology and Applications*. Wiley.

Angelov, P., Kordon, A., and Zhou, X. (2008b). Evolving fuzzy inferential sensors for process industry. In *Proceedings of the International Workshop on Genetic and Evolving Systems, GEFS '08*, pages 41 – 46.

Angelov, P., Ramezani, R., and Zhou, X. (2008c). Autonomous novelty detection and object tracking in video streams using evolving clustering and Takagi-Sugeno type neuro-fuzzy system. In *Proceedings of the IEEE International Joint Conference on Neural Networks, IJCNN '08*, pages 1456–1463.

Angelov, P., Victor, J., Dourado, A., and Filev, D. (2004a). On-line evolution of Takagi-Sugeno fuzzy models. In *Proceedings of the 2nd IFAC Workshop on Advanced Fuzzy/Neural Control, AFNC '04*, pages 67 – 72.

Angelov, P., Xydeas, C., and Filev, D. (2004b). On-line identification of MIMO evolving Takagi-Sugeno fuzzy models. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '04*, pages 55 – 60.

Angelov, P. and Yager, R. (2012). A new type of simplified fuzzy rule-based system. *International Journal of General Systems*, 41(2):163–185.

Angelov, P. and Zhou, X. (2006). Evolving fuzzy systems from data streams in real-time. In *Proceedings of the International Symposium on Evolving Fuzzy Systems*, pages 29 – 35.

Angelov, P. and Zhou, X. (2008a). Evolving fuzzy-rule-based classifiers from data streams. *IEEE Transactions on Fuzzy Systems*, 16(6):1462 – 1475.

Angelov, P. and Zhou, X. (2008b). On line learning fuzzy rule-based system structure from data streams. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '08*, pages 915 – 922.

Angelov, P., Zhou, X., Filev, D., and Lughofer, E. (2007a). Architectures for evolving fuzzy rule-based classifiers. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 2050–2055.

Angelov, P., Zhou, X., and Klawonn, F. (2007b). Evolving fuzzy rule-based classifiers. In *Proceedings of the IEEE Symposium on Computational Intelligence in Image and Signal Processing, CIISP '07*, pages 220 –225.

Bacelar, A., Filho, E., Neves, F., and Landim, R. (2003). On-line linear system parameter estimation using the neo-fuzzy-neuron algorithm. In *Proceedings of the IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pages 115 –118.

Bacelar, A., Neves, F., Filho, E., and Landim, R. (2004). Uma adaptação do algoritmo neo-fuzzy-neuron-generalizado para estimação on-line de parâmetros de sistemas MIMO: Aplicação em uma máquina CC. In *Anais do Congresso Brasileiro de Automática, CBA '04*, volume 1, pages 1 – 6.

Barros, J. and Dexter, A. (2007). Evolving fuzzy model-based adaptive control. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '07*, pages 1 – 5.

Baruah, R., Angelov, P., and Andreu, J. (2011). Simpl_eClass: Simplified potential-free evolving fuzzy rule-based classifiers. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 2249–2254.

Bazaraa, M., Sherali, H., and Shetty, C. (1993). *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, 3 edition.

Birek, L., Petrovic, D., and Boylan, J. (2014). Water leakage forecasting: the application of a modified fuzzy evolving algorithm. *Applied Soft Computing*, 14, Part B(0):305 – 315.

Bodyanskiy, Y., Kokshenev, I., and Kolodyazhniy, V. (2003). An adaptive learning algorithm for a neo fuzzy neuron. In *Proceedings of the 3rd Conference of the European Society for Fuzzy Logic and Technology*, pages 375–379.

Bossley, K. (1995). Neurofuzzy construction algorithms. Technical report, University of Southampton, University of Southampton - Department of Eletronics and Computer Science.

Bouchachia, A., Lughofer, E., and Sayed-Mouchaweh, M. (2014). Special Issue: Evolving soft computing techniques and applications. *Applied Soft Computing*, 14, Part B(0):141 – 143.

Brown, M. and Harris, C. (1994). *Neurofuzzy Adaptive Modeling and Crontrol*. International Series in Systems and Control Engineering. Prentice Hall, 1 edition.

Caminhas, W. (1997). *Estratégias de Detecção e Diagnósticos de Falhas em Sistemas Dinâmicos*. PhD thesis, Programa de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas.

Caminhas, W. and Gomide, F. (2000). A fast learning algorithm for neofuzzy networks. In *Proceedings of the Information Processing and Management of Uncertainty in Knowledge Based Systems, IPMU '00*, volume 1, pages 1784 – 1790.

Caminhas, W., Pereira, G., Tavares, H., and Gomide, F. (1998). Identificação de sistemas dinâmicos: Abordagem baseada em neurônio nebuloso. In *Anais do Simpósio Brasileiro de Redes Neurais, SBRN '98*, pages 105 – 110.

Cao, F. and Wang, Y. (2009). Design of a single-phase grid-connected photovoltaic systems based on Fuzzy-PID controller. In *Proceedings of the Intelligent Computing International Conference on Emerging Intelligent Computing Technology and Applications, ICIC '09*, pages 912–919.

Cernuda, C., Lughofer, E., Marzinger, W., and Kasberger, J. (2011). NIR-based quantification of process parameters in polyetheracrylat (PEA) production using flexible non-linear fuzzy systems. *Chemometrics and Intelligent Laboratory Systems*, 109(1):22 – 33.

Cernuda, C., Lughofer, E., Suppan, L., Roder, T., Schmuch, R., Hintenaus, P., Marzinger, W., and Kasberger, J. (2012). Evolving chemometric models for predicting dynamic process parameters in viscose production. *Analytica Chimica Acta*, (0):22 – 38.

Chang, P., Fan, C., and Hsieh, J. (2009). A weighted evolving fuzzy neural network for electricity demand forecasting. In *Proceedings of the Asian Conference on Intelligent Information and Database Systems, ACIIDS '09*, pages 330 – 335.

Cormen, T., Rivest, R., Stein, C., and Leiserson, C. (2002). *Algoritmos - Teoria e Prática*. Campus Elsevier, 1 edition.

Diebold, F. and Mariano, R. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):253–63.

Dovzan, D., Logar, V., and Skrjanc, I. (2012). Solving the sales prediction problem with fuzzy evolving methods. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC '12*, pages 1–8.

Feedback, I. (2006a). Magnetic levitation control experiments, 33-942s. *UK*.

Feedback, I. (2006b). Twin rotor MIMO system control experiments, 33-942s. *UK*.

Filev, D. and Ying, H. (2013). The look-up table controllers and a particular class of Mamdani fuzzy controllers are equivalent - implications to real-world applications. In *Proceedings of the Joint IFSA World Congress and NAFIPS Annual Meeting, IFSA/NAFIPS '13*, pages 902–907.

Gomez, J. and Dasgupta, D. (2002). Evolving fuzzy classifiers for intrusion detection. In *Proceedings of the IEEE Workshop on Information Assurance*, pages 1 – 7.

Gomez, J. and Leon, E. (2006). A fuzzy set/rule distance for evolving fuzzy anomaly detectors. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '06*, pages 2286 – 2292.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182.

Harris, C., Brown, M., Bossley, K., Mills, D., and Ming, F. (1996). Advances in neurofuzzy algorithms for real-time modelling and control. *Engineering Applications of Artificial Intelligence*, 9(1):1 – 16.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foudation*. Prentice Hall, 2 edition.

Iglesias, J., Angelov, P., Ledezma, A., and Sanchis, A. (2009). Modelling evolving user behaviours. In *Proceedings of the IEEE Workshop on Evolving and Self-Developing Intelligent Systems, ESDIS '09*, pages 16 – 23.

Jamsa, K. and Klander, L. (1999). *Programando em C/C++ - A Bíblia*. Pearson, 2 edition.

Jang, J. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernetics*, 23(3):665 – 685.

Kasabov, N. (2001). Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 31(6):902 – 918.

Kasabov, N. and Filev, D. (1998). Evolving fuzzy neural networks - algorithms, applications and biological motivation. *World Scientific Methodologies for the Conception, Design and Application of Soft Computing*, pages 271–274.

Kasabov, N. and Filev, D. (2006). Evolving intelligent systems: Methods, learning & applications. In *Proceedings of the International Symposium on Evolving Fuzzy Systems*, pages 8 – 18.

Kasabov, N. and Song, Q. (1999). Dynamic evolving fuzzy neural networks with "m-out-of-n" activation nodes for on-line adaptive systems. Technical report, Department of Information Science - University of Otago, Dunedin, New Zealand.

Kasabov, N. and Song, Q. (2002). DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems*, 10(2):144 – 154.

Katakis, I., Tsoumakas, G., and Vlahavas, I. (2005). On the utility of incremental feature selection for the classification of textual data streams. In *Proceedings of the Panhellenic Conference on Advances in Informatics, PCI '05*, pages 338–348.

Katakis, I., Tsoumakas, G., and Vlahavas, I. (2006). Dynamic feature space and incremental feature selection for the classification of textual data streams. In *Proceedings of the International Workshop on Knowledge Discovery from Data Streams, ECML/PKDD '06*, pages 107 –116.

Komijani, M., Lucas, C., Araabi, B., and Kalhor, A. (2012). Introducing evolving Takagi-Sugeno method based on local least squares support vector machine models. *Evolving Systems*, 3(2):81–93.

Leite, D., Ballini, R., Costa, P., and Gomide, F. (2012). Evolving fuzzy granular modeling from nonstationary fuzzy data streams. *Evolving Systems*, 3:65–79.

Leite, D., Costa, P., and Gomide, F. (2009). Evolving granular classification neural network. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN '09*, pages 2204 – 2211. IEEE Press.

Leite, D., Costa, P., and Gomide, F. (2010). Evolving granular neural network for semi-supervised data stream classification. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN '10*, pages 1 – 8.

Leite, D. and Gomide, F. (2012). Evolving linguistic fuzzy models from data streams. In Trillas, E., Bonissone, P. P., Magdalena, L., and Kacprzyk, J., editors, *Combining Experimentation and Theory*, volume 271 of *Studies in Fuzziness and Soft Computing*, pages 209–223. Springer Berlin Heidelberg.

Leite, D., Gomide, F., Ballini, R., and Costa, P. (2011). Fuzzy granular evolving modeling for time series prediction. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '11*, pages 2794–2801.

Lemos, A. (2011). *Modelagem Nebulosa Evolutiva: Novas Topologias e Algoritmos de Aprendizagem*. PhD thesis, Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais.

Lemos, A., Ballini, R., Caminhas, W., and Gomide, F. (2013). System modeling and forecasting with evolving fuzzy algorithms. In *Soft Computing: State of the Art Theory and Novel Applications*, volume 291 of *Studies in Fuzziness and Soft Computing*, pages 255–268.

Lemos, A., Caminhas, W., and Gomide, F. (2010). Fuzzy multivariable gaussian evolving approach for fault detection and diagnosis. In *Computational Intelligence for Knowledge-Based Systems Design*, volume 6178 of *Lecture Notes in Computer Science*, pages 360–369.

Lemos, A., Caminhas, W., and Gomide, F. (2011a). Evolving fuzzy linear regression trees with feature selection. In *Proceedings of the IEEE Workshop on Evolving and Adaptive Intelligent Systems, EAIS '11*, volume 1, pages 31–38.

Lemos, A., Caminhas, W., and Gomide, F. (2011b). Fuzzy evolving linear regression trees. *Evolving Systems*, 2:1–14.

Lemos, A., Caminhas, W., and Gomide, F. (2011c). Multivariable gaussian evolving fuzzy modeling system. *IEEE Transactions on Fuzzy Systems*, 19(1):91 – 104.

Lemos, A., Leite, D., Maciel, L., Ballini, R., Caminhas, W., and Gomide, F. (2012). Evolving fuzzy linear regression tree approach for forecasting sales volume of petroleum products. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '12*, pages 1–8.

Lemos, A., Maia, R., Inácio, M., and Caminhas, W. (2009). Uma metodologia para geração automática de redes neurais nebulosas a partir de árvores de decisão. In *Anais do Congresso Brasileiro de Redes Neurais, SBRN '09*, pages 1 – 5.

Li, Y. (2004). On incremental and robust subspace learning. *Pattern Recognition*, 37:1509–1518.

Lima, E. (2008). Modelagem fuzzy funcional evolutiva participativa. Master's thesis, Programa de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas.

Lima, E., Hell, M., Ballini, R., and Gomide, F. (2010). *Evolving Fuzzy Modeling Using Participatory Learning*, chapter 4, pages 67 – 86. Evolving Intelligent Systems: Methodology and Applications. John Wiley & Sons, 1 edition.

Lin, Y., Chang, J., and Lin, C. (2013a). Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 24(2):310–321.

Lin, Y., Chang, J., and Lin, C. (2014). A TSK-Type-Based self-evolving compensatory interval Type-2 fuzzy neural network (TSCIT2FNN) and its applications. *IEEE Transactions on Industrial Electronics*, 61(1):447–459.

Lin, Y., Chang, J., Pal, N., and Lin, C. (2013b). A mutually recurrent interval Type-2 neural fuzzy system (MRIT2NFS) with self-evolving structure and parameters. *IEEE Transactions on Fuzzy Systems*, 21(3):492–509.

Linden, R. and Bhaya, A. (2007). Evolving fuzzy rules to model gene expression. *Biosystems*, 88(1-2):76 – 91.

Lughofer, E. (2008a). Extensions of vector quantization for incremental clustering. *Pattern Recognition*, 41(3):995 – 1011.

Lughofer, E. (2008b). FLEXFIS: A robust incremental learning approach for evolving Takagi-Sugeno fuzzy models. *IEEE Transactions on Fuzzy Systems*, 16(6):1393 – 1410.

Lughofer, E. (2011). On-line incremental feature weighting in evolving fuzzy classifiers. *Fuzzy Sets Systems*, 163(1):1–23.

Lughofer, E. (2012). Flexible evolving fuzzy inference systems from data streams (FLEXFIS++). In *Learning in Non-Stationary Environments*, pages 205–245.

Lughofer, E. (2013). On-line assurance of interpretability criteria in evolving fuzzy systems - achievements, new concepts and open issues. *Information Sciences*, 251(0):22 – 46.

Lughofer, E. and Angelov, P. (2011). Handling drifts and shifts in on-line data streams with evolving fuzzy systems. *Applied Soft Computing*, 11(2):2057–2068.

Lughofer, E., Angelov, P., and Zhou, X. (2007). Evolving single- and multi-model fuzzy classifiers with FLEXFIS-Class. In *Proceedings of the IEEE International Fuzzy Systems Conference, FUZZ-IEEE '07*, pages 1 –6.

Lughofer, E. and Guardiola, C. (2008). Applying evolving fuzzy models with adaptive local error bars to on-line fault detection. In *Proceedings of the International Workshop on Genetic and Evolving Systems, GEFS '08*, pages 35 – 40.

Lughofer, E. and Kindermann, S. (2010). SparseFIS: Data-driven learning of fuzzy systems with sparsity constraints. *IEEE Transactions on Fuzzy Systems*, 18(2):396 –411.

Lughofer, E., Macian, V., Guardiola, C., and Klement, E. (2011a). Identifying static and dynamic prediction models for NOx emissions with evolving fuzzy systems. *Applied Soft Computing*, 11(2):2487 – 2500.

Lughofer, E., Trawinski, B., Trawinski, K., Kempa, O., and Lasota, T. (2011b). On employing fuzzy modeling algorithms for the valuation of residential premises. *Information Sciences*, 181(23):5123 – 5142.

Maciel, L., Gomide, F., and Ballini, R. (2012a). An enhanced approach for evolving participatory learning fuzzy modeling. In *Proceedings of the IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS '12*, pages 23–28.

Maciel, L., Gomide, F., and Ballini, R. (2012b). MIMO evolving functional fuzzy models for interest rate forecasting. In *Proceedings of the IEEE Conference on Computational Intelligence for Financial Engineering Economics, CIFEr '12*, pages 1–8.

Maciel, L., Gomide, F., and Ballini, R. (2012c). MIMO evolving participatory learning fuzzy modeling. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '12*, pages 1–8.

Maciel, L., Gomide, F., and Ballini, R. (2013a). Enhanced evolving participatory learning fuzzy modeling: an application for asset returns volatility forecasting. *Evolving Systems*, pages 1–14.

Maciel, L., Gomide, F., Ballini, R., and Yager, R. (2013b). Simplified evolving rule-based fuzzy modeling of realized volatility forecasting with jumps. In *Proceedings of the IEEE Conference on Computational Intelligence for Financial Engineering Economics, CIFEr '13*, pages 82–89.

Maciel, L., Lemos, A., Gomide, F., and Ballini, R. (2012d). Evolving fuzzy systems for pricing fixed income options. *Evolving Systems*, 3(1):5–18.

Mackey, M. and Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science*, 197(4300):287 – 289.

Mallinson, H. and Bentley, P. (1999). Evolving fuzzy rules for pattern classification. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation, CIMCA '99*, volume 55, pages 184–191.

Mamdani, E. and Assilian, S. (1999). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Human-Computer Studies*, 51(2):135–147.

Marsala, C. (2013). Fuzzy decision trees for dynamic data. In *Proceedings of the IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS '13*, pages 17–24.

MathWorks, I. (2009). Real-time workshop 7 users guide. *Natick, MA, USA*.

McDonald, S. and Angelov, P. (2010). Evolving Takagi-Sugeno modelling with memory for slow processes. *KES Journal: Innovation in Knowledge-Based & Intelligent Engineering Systems*, 14(1):11 – 19.

Nayak, P. and Sudheer, K. (2008). Fuzzy model identification based on cluster estimation for reservoir inflow forecasting. *Hydrological Processes*, 22(6):827 – 841.

Nejjari, F., Rotondo, D., Puig, V., and Innocenti, M. (2012). Quasi-LPV modelling and non-linear identification of a twin rotor system. In *Proceedings of the 20th Mediterranean Conference on Control Automation*, pages 229–234.

Nguyen, M., Guo, J., and Shi, D. (2006). ESOFCMAC: Evolving self-organizing fuzzy cerebellar model articulation controller. In *Proceedings of the IEEE International Joint Conference on Neural Networks, IJCNN '06*, pages 3694 – 3699.

Pedrycz, W. (1991). Neurocomputations in relational systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):289 – 297.

Pedrycz, W. (2005). *Knowledge-based clustering - from data to information granules*. Wiley.

Pedrycz, W. and Gomide, F. (2007). *Fuzzy Systems Engineering: Toward Human-Centric Computing*. John Wiley & Sons, 1 edition.

Potts, D. and Sammut, C. (2004). Incremental learning of linear model trees. *Machine Learning*, 61(1):5 – 48.

Pouzols, F. M. and Lendasse, A. (2010). Evolving fuzzy optimally pruned extreme learning machine for regression problems. *Evolving Systems*, 1(1):43–58.

Pratama, M., Anavatti, S., and Lughofer, E. (2013a). Evolving fuzzy rule-based classifier based on GENEFIS. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '13*, pages 1–8.

Pratama, M., Anavatti, S., and Lughofer, E. (2013b). GENEFIS: Towards an effective localist network. *IEEE Transactions on Fuzzy Systems*, (99):1–17.

Rahideh, A. and Shaheed, M. (2008). Dynamic modelling of a twin rotor MIMO system using grey box approach. In *Proceedings of the 5th International Symposium on Mechatronics and Its Applications, ISMA '08*, pages 1–6.

Rong, H., Han, S., and Zhao, G. (2014). Adaptive fuzzy control of aircraft wing-rock motion. *Applied Soft Computing*, 14, Part B(0):181 – 193.

Rong, H., Sundararajan, N., Huang, G., and Saratchandran, P. (2006). Sequential adaptive fuzzy inference system (SAFIS) for non-linear system identification and prediction. *Fuzzy Sets Systems*, 157(9):1260 – 1275.

Rong, H., Sundararajan, N., Huang, G., and Zhao, G. (2011). Extended sequential adaptive fuzzy inference system for classification problems. *Evolving Systems*, 2(2):71–82.

Rubio, J. (2009). SOFMLS: Online self-organizing fuzzy modified least-squares network. *IEEE Transactions on Fuzzy Systems*, 17(6):484 – 498.

Shahin, M., Maier, H., and Jaksa, M. (2003). Neural and neurofuzzy techniques applied to modelling settlement of shallow foundations on granular soils. In *Proceedings of the International Congress on Modelling and Simulation, MODSIM '03*, volume 4, pages 1886–1891.

Shaker, A., Senge, R., and Hulermeier, E. (2013). Evolving fuzzy pattern trees for binary classification on data streams. *Information Sciences*, 220(0):34 – 45.

Silva, A., Caminhas, W., and Lemos, A. (2010). Uma breve revisão de sistemas nebulosos evolutivos. In *I Congresso Brasileiro de Sistemas Fuzzy, I CBSF*, pages 1–8.

Silva, A., Caminhas, W., Lemos, A., and Gomide, F. (2012a). Evolving neural fuzzy network with adaptive feature selection. In *Proceedings of the 11th International Conference on Machine Learning and Applications, ICMLA '12*, volume 2, pages 440 –445.

Silva, A., Caminhas, W., Lemos, A., and Gomide, F. (2012b). Modelo nebuloso evolutivo com seleção adaptiva de entradas. In *II Congresso Brasileiro de Sistemas Fuzzy, II CBSF*, pages 1–8.

Silva, A., Caminhas, W., Lemos, A., and Gomide, F. (2012c). Modelo nebuloso evolutivo para sistemas de tempo real com dinâmica rápida. In *XIX Congresso Brasileiro de Automática, XIX CBA*, pages 1–8.

Silva, A., Caminhas, W., Lemos, A., and Gomide, F. (2013a). Evolving neo-fuzzy neural network with adaptive feature selection. In *1st BRICS Countries, BRICS-CCI '13*, pages 1–8.

Silva, A., Caminhas, W., Lemos, A., and Gomide, F. (2013b). Rede neuro-fuzzy evolutiva para estimação em tempo real de posição de um sistema de levitação magnética. In *XI Simpósio Brasileiro de Automação Inteligente, XI SBAI*, pages 1–8.

Silva, A., Caminhas, W., Lemos, A., and Gomide, F. (2014a). Estimação em tempo real de variáveis de estado do TRMS com rede neurofuzzy evolutiva. In *XX Congresso Brasileiro de Automática, XX CBA*, pages 1–8.

Silva, A., Caminhas, W., Lemos, A., and Gomide, F. (2014b). Extended approach for evolving neo-fuzzy neural with adaptive feature selection. In *Proceedings of the 11th International FLINS Conference on Decision Making and Soft Computing, FLINS '14*, pages 1–6.

Silva, A., Caminhas, W., Lemos, A., and Gomide, F. (2014c). A fast learning algorithm for evolving neo-fuzzy neuron. *Applied Soft Computing*, 14, Part B(0):194 – 209.

Silva, A., Rosa, Raul, C. W., Lemos, A., and Gomide, F. (2014d). Rede neuro-fuzzy evolutiva com seleção de entradas aplicada na modelagem de sistemas. In *III Congresso Brasileiro de Sistemas Fuzzy, III CBSF*, pages 1–8.

Silva, L., Gomide, F., and Yager, R. (2005). Participatory learning in fuzzy clustering. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '05*, pages 857 – 861.

Smith, F. and Tighe, A. (2005). Adapting in an uncertain world. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 6, pages 5958 – 5963.

Song, Q. and Kasabov, N. (2001). ECM - A novel on-line, evolving clustering method and its applications. In *Proceedings of the Biannual Conference on Artificial Neural Networks and Expert Systems, ANNES '01*, pages 87–92.

Subudhi, B. and Jena, D. (2009). Nonlinear system identification of a twin rotor MIMO system. In *Proceedings of the IEEE Region 10 Conference, TENCON '09*, pages 1–6.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15(1):116 – 132.

Toha, S. and Tokhi, M. (2009). Dynamic nonlinear inverse-model based control of a twin rotor system using adaptive neuro-fuzzy inference system. In *Proceedings of the Third UKSim European Symposium on Computer Modeling and Simulation, EMS '09*, pages 107–111.

Toscani, L. and Veloso, P. (2012). *Complexidade de Algoritmos*, volume 13. Bookman, 3 edition.

Tung, S. and Quek, C. (2010). eFSM - A novel online neural-fuzzy semantic memory model. *IEEE Transactions on Neural Networks*, 21(1):136–157.

Tung, S., Quek, C., and Guan, C. (2013). eT2FIS: An evolving Type-2 neural fuzzy inference system. *Information Sciences*, 220(0):124 – 148.

Uchino, E. and Yamakawa, T. (1994). Neo-fuzzy-neuron based new approach to system modeling with application to actual system. In *Proceedings of the International Conference on Tools with Artificial Intelligence*, pages 564 – 570.

Wang, D., Zeng, X., and Keane, J. (2010). A structure evolving learning method for fuzzy systems. *Evolving Systems*, 1:83–95.

Wang, D., Zeng, X., and Keane, J. (2013). A simplified structure evolving method for Mamdani fuzzy system identification and its application to high-dimensional problems. *Information Sciences*, 220(0):110 – 123.

Wang, W. and Vrbanek, J. (2008). An evolving fuzzy predictor for industrial applications. *IEEE Transactions on Fuzzy Systems*, 16(6):1439 – 1449.

Xydeas, C., Angelov, P., Chiao, S., and Reoullas, M. (2006). Advances in classification of EEG signals via evolving fuzzy classifiers and dependant multiple HMMs. *Computers in Biology and Medicine*, 36(10):1064 – 1083.

Yager, R. (1990). A model of participatory learning. *IEEE Transactions on Systems, Man and Cybernetics*, 20(5):1229 – 1234.

Yager, R. (2004). Participatory learning: a paradigm for more human like learning. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '04*, pages 79 – 84.

Yamakawa, T., Uchino, E., Miki, T., and Kusabagi, H. (1992). A neo fuzzy neuron and its applications to system identification and predictions to system behavior. In *Proceedings of the International Conference on Fuzzy Logic and Neural Networks*, volume 1, pages 477 – 484.

Zaychenko, Y. and Gasanov, A. (2012). Investigations of cascade neo-fuzzy neural networks in the problem of forecasting at the stock exchange. In *Proceedings of the IV International Conference Problems of Cybernetics and Informatics, PCI '12*, pages 1–3.

Zhu, J., Lao, N., and Xing, E. (2010). Grafting-light: fast, incremental feature selection and structure learning of Markov random fields. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '10*, pages 303–312.

Ziviani, N. (2006). *Projeto de Algoritmos com Implementações em Java e C++*. Thomson Learning, 1 edition.