CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

PROGRAMAÇÃO GENÉTICA APLICADA NA CONSTRUÇÃO DA BASE DE REGRAS EM REDES *Neuro-Fuzzy* TIPO *Neo-Fuzzy-Neuron*

GLENDER BRÁS DE MEDEIROS

Orientador: Alisson Marques da Silva
CEFET-MG

BELO HORIZONTE
DEZEMBRO DE 2019

GLENDER BRÁS DE MEDEIROS

PROGRAMAÇÃO GENÉTICA APLICADA NA CONSTRUÇÃO DA BASE DE REGRAS EM REDES *Neuro-Fuzzy* TIPO *Neo-Fuzzy-Neuron*

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Matemática e Computacional do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito parcial para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Área de concentração: Modelagem Matemática e Computacional

Linha de pesquisa: Sistemas Inteligentes

Orientador: Alisson Marques da Silva
CEFET-MG

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL
BELO HORIZONTE
DEZEMBRO DE 2019

Medeiros, Glender Brás de
M488p Programação genética aplicada na construção da base de regras em redes Neuro-Fuzzy tipo Neo-Fuzzy-Neuron / Glender Brás de Medeiros. – 2019.
75 f.

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional.
Orientador: Alisson Marques da Silva.
Dissertação (mestrado) – Centro Federal de Educação Tecnológica de Minas Gerais.

1. Sistemas difusos – Teses. 2. Redes neurais (Computação) – Teses. 3. Programação genética (Computação) – Teses. I. Silva, Alisson Marques da. II. Centro Federal de Educação Tecnológica de Minas Gerais. III. Título.

CDD 001.64404

Elaboração da ficha catalográfica pela Biblioteca-Campus II / CEFET-MG

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

"PROGRAMAÇÃO GENÉTICA APLICADA NA CONSTRUÇÃO DA BASE DE REGRAS EM REDES NEURO-FUZZY TIPO NEO-FUZZY-NEURON"

Dissertação de Mestrado apresentada por **Glender Brás de Medeiros**, em 12 de dezembro de 2019, ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, e aprovada pela banca examinadora constituída pelos professores:

Prof. Dr. Alisson Marques da Silva (Orientador)
Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Walmir Matos Caminhas
Universidade Federal de Minas Gerais

Prof. Dr. Paulo Eduardo Maciel de Almeida
Centro Federal de Educação Tecnológica de Minas Gerais

Visto e permitida a impressão,

Prof. Dr. Thiago de Souza Rodrigues
Coordenador do Programa de Pós-Graduação em
Modelagem Matemática e Computacional

Agradecimentos

A Deus, que possibilitou que eu chegasse até aqui, que guiou cada passo, que esteve comigo em cada momento, que me sustentou em cada dificuldade, que proporcionou cada pequena vitória, que foi responsável por cada alegria, que nunca deixa de mostrar o quanto me ama, que transforma cada sonho em realidade, que não permite que eu desista e, por fim, que é a razão de toda a minha existência.

Ao meu orientador, Prof. Alisson Marques da Silva, por todo apoio e direcionamento, pela ajuda nos momentos que precisei, pela paciência durante todo este percurso e pela grande contribuição para minha formação.

Aos professores Flávio Cruzeiro e Sérgio Ricardo que foram essenciais principalmente no início do meu caminho no MMC.

À Profa. Gisele Pappa do DCC/UFMG por todo apoio desde o fim da minha graduação, pela disponibilidade em me ajudar e contribuir, seja com ideias, conselhos ou questões técnicas sempre que precisei.

Aos colegas e amigos do MMC pelas discussões e partilhas de conhecimento, por compartilhar das alegrias e dos momentos difíceis ao longo do mestrado e também pela distração e descontração nos momentos necessários.

A todos os meus amigos que me acompanharam, torceram e estiveram comigo direta ou indiretamente neste tempo.

À minha família por ser minha base e sempre torcer, acreditar em mim e apoiar meus sonhos.

À família EJC, família que Deus me permitiu escolher, que me ajuda a estar sempre mais próximo de Deus e com quem eu vivi momentos incríveis, em especial neste último ano.

Ao CEFET-MG pela estrutura e por ser a raiz da minha formação profissional.

À CAPES pela apoio financeiro.

“Conheça todas as teorias, domine todas as técnicas, mas ao tocar uma alma humana, seja apenas outra alma humana.”
(Carl Jung)

Resumo

Este trabalho propõe a utilização de Programação Genética para construção da base de regras em Redes *Neuro-Fuzzy* do tipo *Neo-Fuzzy-Neuron* (NFN). Redes *Neuro-Fuzzy* são sistemas híbridos que combinam características dos Sistemas *Fuzzy*, como lidar com dados linguísticos e imprecisos, e das Redes Neurais Artificiais, como a capacidade de incorporar técnicas de aprendizado. A rede NFN é uma rede *Neuro-Fuzzy* composta por n modelos Takagi-Sugeno de ordem $zero$, um para cada variável de entrada. Programação Genética (PG) é uma técnica de Computação Evolucionária que utiliza analogias aos conceitos de evolução natural dos seres vivos para gerar e evoluir programas computacionais automaticamente. É um método populacional, isto é, trabalha com uma população de indivíduos (soluções) que buscam gerar indivíduos melhores a partir do processo evolucionário da PG. Neste trabalho são propostos três novos algoritmos para redes *Neuro-Fuzzy* do tipo *Neo-Fuzzy-Neuron* com aprendizado por Programação Genética para construção da base de regras em cooperação com um método do gradiente para ajuste dos parâmetros do consequente. A escolha da PG se deve principalmente pela capacidade desta de, como qualquer outro método evolucionário, explorar um amplo espaço de busca com menos passos que algoritmos convencionais e fugir de ótimos locais. Por fim, os modelos foram avaliados e comparados com modelos alternativos do estado da arte que utilizam métodos convencionais de aprendizado em problemas de previsão e identificação de sistemas. Os resultados obtidos mostram que a Programação Genética é uma alternativa eficaz para a construção da base de regras em redes *Neo-Fuzzy-Neuron*, possibilitando obter modelos com boa acurácia quando aplicados em problemas de previsão e identificação de sistemas.

Palavras-chave: Sistemas *Fuzzy*, Redes *Neuro-Fuzzy*, Sistemas *Fuzzy*-Genéticos, Programação Genética.

Abstract

This work proposes the use of Genetic Programming to building rule bases in Neuro-Fuzzy Networks Neo-Fuzzy-Neuron(NFN) type. Neuro-Fuzzy Networks are hybrid systems that combine characteristics of Fuzzy Systems, such as handling imprecise linguistic data, and Artificial Neural Networks, such as the ability to incorporate learning techniques. The NFN network is a Neuro-Fuzzy network consisting of n Takagi-Sugeno models of order $zero$, one for each input variable. Genetic Programming (GP) is an Evolutionary Computing technique that uses analogies of natural evolution concepts of living beings to automatically generate and evolve computer programs. It is a population method, that is, it works with a population of individuals (solutions) that seek to generate better individuals through the evolutionary process of GP. In this work, we proposed three new approaches to Neo-Fuzzy-Neuron Network with Genetic Programming learning to build the rule base in cooperation with a gradient method to adjust the consequent parameters. The choice of PG is mainly due to its ability to, like any other evolutionary method, explore a large search space with fewer steps than conventional algorithms and avoid local optimal. Finally, the models were evaluated for forecasting and system identification problems and compared with alternative state-of-the-art models using conventional learning methods. The results show that Genetic Programming is an effective alternative for building the rule base in Neo-Fuzzy-Neuron networks, making it possible to obtain models with good accuracy when applied to forecast and system identification problems.

Keywords: Fuzzy Systems, Neuro-Fuzzy Networks, Genetic Fuzzy Systems, Genetic Programming.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

ANFIS	*Adaptative Neuro-Fuzzy Inference System*
ELM	*Extreme Learning Machine*
EQM	Erro Quadrático Médio
GENFIS	*Genetic Fuzzy Inference System*
IRL	*Iterative Rule Learning*
MLP	*Multilayer Perceptron*
NFN	*Neo-Fuzzy-Neuron*
NFN-PG-MG	*Neo-Fuzzy-Neuron* com Programação Genética Multi-Gene
NSGA-II	*Non-dominated Sorting Genetic Algorithm II*
PG	Programação Genética
PG-MG	Programação Genética Multi-Gene
RMSE	*Root Mean Square Error*
SVM	*Support Vector Machine*
TS	Takagi-Sugeno

Lista de Símbolos

c	Centro da função
d	Derivada do erro
σ	Desvio padrão
Δ	Distância entre o centro de duas funções
e	Erro retornado
s	Espalhamento da função
w	Grau de ativação
μ	Grau de pertinência
max_x	Máximo da variável x
z	Média amostral do grupo
$\hat{z}$	Média amostral global
min_x	Mínimo da variável x
l	Número de épocas de treinamento
g	Número de gerações
m_ini	Número inicial de funções de pertinência
m_max	Número máximo de funções de pertinência
q	Peso de uma regra *fuzzy*
t	Quantidade de amostras
m	Quantidade de funções de pertinência
n	Quantidade de variáveis de entrada
φ	Relação *fuzzy*
y	Saída desejada
$\hat{y}$	Saída prevista pelo modelo

y_i	Saída individual das regras
tam_pop	Tamanho da população
Trn	Tamanho do torneio
α	Taxa de Aprendizado
tx_c	Taxa de cruzamento
tx_m	Taxa de mutação
S	Variância
x	Variável de entrada

Sumário

Capítulo 1

Introdução

Neste capítulo inicial apresenta-se uma contextualização sobre Sistemas *Fuzzy*, Redes *Neuro-Fuzzy* e os principais desafios no desenvolvimento destes. A motivação para a realização deste trabalho e sua relevância são apresentados na Seção 1.1. A Seção 1.2 mostra o objetivo geral e os objetivos específicos a serem alcançados. As principais contribuições e as publicações geradas a partir deste trabalho são descritas na Seção 1.3. Por fim, na Seção 1.4, é sumarizado o conteúdo de cada um dos próximos capítulos.

1.1 Motivação e Relevância

Os Sistemas *Fuzzy*[1] (ZADEH, 1965) e suas variações são amplamente utilizados na resolução de problemas nos quais são necessárias capacidades de raciocínio semelhantes às dos seres humanos (MAMDANI; ASSILIAN, 1999). Dentre esses problemas pode-se citar modelagem (TIBIRIÇÁ et al., 2005; MODELLING. . . , 2015; KARYOTIS et al., 2016), controle (JUANG, 2002; CHEN; LIN; LIN, 2009; HAN et al., 2015), classificação (YANG; LIU; CHEN, 2006; MELIN et al., 2013; LI; WANG; ZHANG, 2017), identificação, detecção e diagnóstico de falhas (GALLOVA, 2009; WU, 2011; HANG; ZHANG; CHENG, 2016), previsão e predição (ANDRADE; FLAUZINO; SILVA, 2010; MIRANIAN; ABDOLLAHZADE, 2013; THOMAS et al., 2016), saúde (CARMONA et al., 2015; OMISORE; SAMUEL; ATAJEROMAVWO, 2017; ASHISH et al., 2018), finanças (AZNARTE et al., 2012; SHEN; TZENG, 2014; DASH; DASH, 2016), mineração de dados e reconhecimento de padrões (CARMONA et al., 2013b; RADHAKRISHNA et al., 2018), reconhecimento de imagens (LIU; CHEN, 2007; MYLONAS; STAVRAKOUDIS; THEOCHARIS, 2013), redes de computadores e segurança da informação (VICTOIRE; SAKTHIVEL, 2011; SHAMSHIRBAND et al., 2014; ELHAG et al., 2015).

[1] Neste trabalho será adotado o termo original em Inglês devido à nomenclatura em Português ainda não ser bem defina, embora alguns autores utilizem o termo Nebuloso ou Difuso como tradução para *Fuzzy*.

Inicialmente, a modelagem dos Sistemas *Fuzzy*, isto é, a criação da base de regras e ajustes dos parâmetros, era realizada por conhecimento especialista. A partir dos anos 90, com aumento da complexidade dos sistemas e com a disponibilidade de dados sobre esses sistemas, a modelagem passou a ser realizada com base em dados e o conhecimento especialista, se necessário, utilizado como informação complementar (MALLINSON; BENTLEY, 1999; KASABOV; FILEV, 2006). A construção da base de regras e ajuste dos parâmetros em Sistemas *Fuzzy* pode ser realizada utilizando técnicas de aprendizado máquina e um conjunto de dados de treinamento (MATÍA; MARICHAL; JIMÉNEZ, 2014). Geralmente, a construção da base de regras é realizada por algoritmos de agrupamento *fuzzy*, utilizando informações sobre a organização espacial das variáveis de entrada. Outras abordagens utilizam o erro de modelagem calculado recursivamente para construção do conjunto de regras (LEE, 1990). Os parâmetros do antecedente e consequente são ajustados, por exemplo, pelo método do gradiente (GUÉLY; SIARRY, 1993) ou por mínimos quadrados recursivos (WANG; LEE, 2002). A busca por novos algoritmos de aprendizado para construção da base de regras e ajuste de parâmetros é um desafio contínuo. Neste contexto, destaca-se o uso de abordagens híbridas que buscam unir as capacidades de diferentes técnicas de modelagem na construção sistemas mais eficientes e robustos. Uma dessas abordagens híbridas são as Redes *Neuro-Fuzzy*.

Redes *Neuro-Fuzzy* são Redes Neurais formadas por neurônios *fuzzy*. Essas possuem a capacidade de treinamento e aprendizado das Redes Neurais Artificiais e de processamento de informações linguísticas e imprecisas dos Sistemas *Fuzzy* (MITRA; HAYASHI, 2000). O desafio no desenvolvimento de abordagens híbridas, mais especificamente em Redes *Neuro-Fuzzy*, está na obtenção de um sistema que seja capaz de construir corretamente seu processo de raciocínio e que tenha capacidade de generalização. Para isso, são empregadas diversas técnicas de aprendizado de máquina, visando tornar essa modelagem automática e eficiente. A cooperação entre duas técnicas tem se demonstrado bastante eficiente para suprir diversas dificuldades na modelagem dos Redes *Neuro-Fuzzy*, principalmente na construção base de regras e no ajuste dos parâmetros (ABRAHAM, 2001). Dentre as abordagens utilizadas para aprendizado em Redes *Neuro-Fuzzy* têm-se as técnicas de Computação Evolucionária que se destacam pela sua capacidade de adaptação e aprendizado, baseado em busca global (HERRERA, 2008).

Computação Evolucionária é a área da computação inspirada nos processos de evolução natural dos seres vivos, em que o indivíduo mais forte tende a sobreviver. Utiliza analogias a estes processos como seleção natural, cruzamento e mutação para gerar e evoluir uma população de soluções, na qual cada indivíduo (solução) compete pela sobrevivência (KELLER DERONG LIU, 2016). As principais abordagens utilizadas

nessa área são Algoritmos Genéticos (GOLDBERG; HOLLAND, 1988), Evolução Diferencial (STORN; PRICE, 1997), Estratégias Evolutivas (BEYER; SCHWEFEL, 2002) e Programação Genética (KOZA, 1992). O principal diferencial da Programação Genética para as outras técnicas evolucionárias é a capacidade de gerar programas computacionais interpretáveis, o que facilita, por exemplo, a representação e regressão de funções matemáticas. A aplicação desses métodos em Redes *Neuro-Fuzzy* tem sido amplamente estudada para tarefas como construção da base de regras e ajuste dos parâmetros (CORDÓN et al., 2004).

Neste contexto, este trabalho propõe o uso de técnicas de Programação Genética (PG) para construção da base de regras em Redes *Neuro-Fuzzy*, mais especificamente em Redes *Neuro-Fuzzy* do tipo *Neo-Fuzzy-Neuron* (YAMAKAWA et al., 1992). Por ser um método populacional e evolucionário, a PG pode encontrar boas soluções em um amplo espaço de busca com menos passos que outros algoritmos.

1.2 Objetivos: geral e específicos

O objetivo geral deste trabalho é propor e implementar algoritmos que utilizem a Programação Genética para criação da base de regras em Redes *Neuro-Fuzzy* do tipo *Neo-Fuzzy-Neuron*.

Os objetivos específicos são:

- Propor algoritmos de aprendizado baseados em Programação Genética para Redes *Neuro-Fuzzy* do tipo *Neo-Fuzzy-Neuron*.
- Validar as abordagens propostas em problemas de previsão e identificação de sistemas.

1.3 Contribuições

A principal contribuição deste trabalho está no desenvolvimento de modelos *Neuro-Fuzzy* do tipo *Neo-Fuzzy-Neuron* com aprendizado por Programação Genética para modelagem da base de regras em cooperação com um método do Gradiente para ajuste automático dos parâmetros do consequente. A abordagem proposta proporciona a obtenção de sistemas com aprendizado a partir de dados com boa acurácia. A utilização de um algoritmo evolucionário favorece o treinamento do sistema por permitir uma exploração ampla e global do espaço de busca que permite chegar a uma boa solução com um número reduzido de passos, já que esta exploração é guiada pelos processos evolucionários do algoritmo. Em outras palavras, a utilização de um método populacional e evolucionário possibilita um treinamento mais rápido e eficiente, pois

esta técnica dispensa a exploração exaustiva de todo o espaço de busca. As evoluções na estrutura genética da população de indivíduos (soluções) permitem fugir de ótimos locais, o que é um problema nos métodos clássicos de aprendizado e permite encontrar uma melhor solução com menos iterações no algoritmo de treinamento.

1.3.1 Publicações

Durante a realização deste trabalho foram publicados os seguintes artigos:

- Brás, G.; Silva, A. M.. **Neo-Fuzzy-Neuron Network with Genetic Programming Learning for Non Linear Regression Problems**. ENIAC - XVI Encontro Nacional de Inteligência Artificial e Computacional. 2019 - Este artigo apresenta a primeira versão dos algoritmos descritos nas Seções 3.2 e 3.3 aplicados na previsão de vazão semanal em usinas hidrelétricas e identificação de sistemas não-lineares.
- Brás, G.; Silva, A. M.. **Non Linear Systems Identification and Forecasting Based on New Fuzzy System with Genetic Programming**. CILAMCE - XL Ibero-Latin-American Congress on Computational Methods in Engineering. 2019 - Este artigo apresenta a primeira versão do algoritmo descrito na Seção 3.4 aplicado à previsão de temperatura e séries temporais.
- Brás, G.; Silva, A. M.. **Uma Nova Abordagem para Construção de Regras em Redes *Neo-Fuzzy-Neuron* Utilizando Programação Genética Multi-Gene**. CBIC - XIV Congresso Brasileiro de Inteligência Computacional. 2019 - Este artigo apresenta a segunda versão do algoritmo descrito na Seção 3.4 aplicado à previsão de temperatura e séries temporais.

1.4 Organização do trabalho

O restante deste trabalho está dividido da seguinte maneira:

- No Capítulo 2 é apresentada uma revisão bibliográfica, discorrendo sobre os conceitos básicos tratados neste trabalho: Sistemas *Fuzzy*, Redes *Neuro-Fuzzy*, Computação Evolucionária, Programação Genética e Sistemas *Fuzzy* Genéticos; e uma revisão da literatura, posicionando sobre o estado da arte a cerca de Redes *Neo-Fuzzy-Neuron* e Sistemas *Fuzzy* Genéticos.
- O Capítulo 3 apresenta os algoritmos propostos neste trabalho.
- O Capítulo 4 descreve os experimentos realizados para validação dos algoritmos e discute os resultados alcançados.
- O Capítulo 5 traz as considerações finais e perspectivas de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Este capítulo apresenta uma revisão bibliográfica acerca dos principais temas tratados nesse trabalho: conceitos básicos e revisão da literatura. Na Seção 2.1 são discutidos os conceitos e a estrutura dos Sistemas *Fuzzy*, Redes *Neuro-Fuzzy* e a rede *Neo-Fuzzy-Neuron* (NFN), que será utilizada para implementação dos algoritmos apresentados no Capítulo 3. Ao final desta seção é apresentada uma breve revisão da literatura sobre trabalhos que exploram algoritmos de aprendizado para redes *Neo-Fuzzy-Neuron*. A Seção 2.2 discorre sobre os conceitos e técnicas de Computação Evolucionária com foco em Programação Genética, Programação Genética Multi-Gene e Sistemas *Fuzzy* Genéticos. Ao final desta seção é apresentada uma breve revisão da literatura sobre trabalhos que exploram técnicas de Computação Evolucionária para aprendizado em Sistemas *Fuzzy* e Redes *Neuro-Fuzzy*.

2.1 Sistemas *Fuzzy*

Os Sistemas *Fuzzy* utilizam a lógica *fuzzy* (ZADEH, 1965) para construir um processo de raciocínio aproximado que tenha a capacidade de lidar com informações linguísticas, incertas e imprecisas, o que se assemelha à linguagem natural (KACPRZYK; PEDRYCZ, 2015). A representação do conhecimento nos Sistemas *Fuzzy* é realizada pelo uso de variáveis linguísticas e seus valores também linguísticos, que são definidos por conjuntos *fuzzy*. Um exemplo de representação de conjuntos *fuzzy* pode ser visto na Figura 1. Esses valores são especificados conforme o contexto do problema e são definidos, para cada variável, por funções pertinência que representam os conjuntos *fuzzy*.

Os Sistemas *Fuzzy* baseados em regras são compostos de três etapas: fuzzificação, inferência e defuzzificação, como mostrado na Figura 2. A fuzzificação consiste em ler os valores de entrada do sistema, que normalmente são precisos (valores *crisp*), e dividi-los em conjuntos *fuzzy* para obter o antecedente das regras. Os conjuntos *fuzzy*

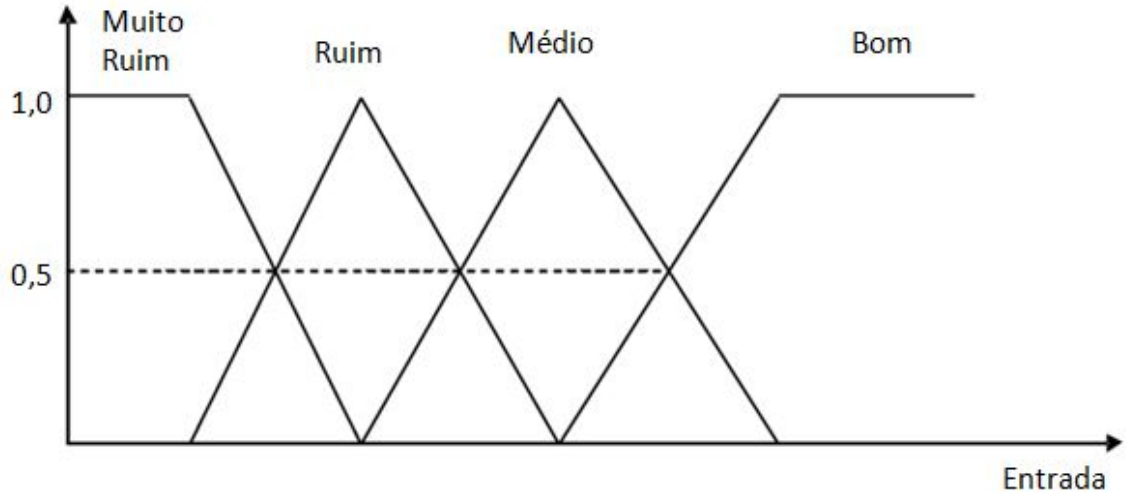

Figura 1 – Exemplo de representação de conjuntos *fuzzy*.

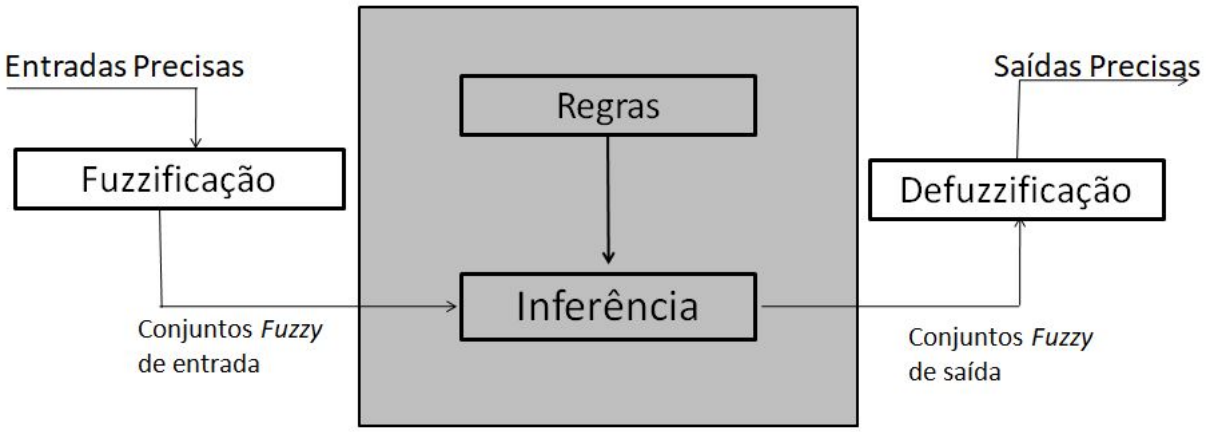

Figura 2 – Estrutura básica de um Sistema *Fuzzy*.

consideram valores linguísticos qualitativos, atribuindo um conceito à variável (GANGA; CARPINETTI; POLITANO, 2011), por exemplo: muito ruim, ruim, médio, bom, com mostrado na Figura 1. Convertidos em conjuntos *fuzzy*, os dados são encaminhados para o mecanismo de inferência. Este contém a base de regras e mapeia as entradas fuzzificadas às regras para gerar saídas também fuzzificadas (FULCHER, 2008).

O mecanismo de inferência é composto por quatro etapas (JANG; SUN; MIZUTANI, 1997). A primeira etapa define o grau de pertinência das variáveis ao conjunto a que pertencem. Ou seja, será atribuído um grau de pertencimento a cada um dos conjuntos existentes entre 0 e 1. A função de pertinência que define uma relação *fuzzy* φ entre a

variável x e o conjunto *fuzzy* u para uma regra i pode ser dada pela Equação (1):

$$\varphi(x,y) = \max(\varphi_i(\text{x,u})). \tag{1}$$

Após obter os graus de pertinência é necessário verificar o grau de ativação de cada regra. A partir do grau de pertinência, verifica-se em que nível a relação obtida na etapa anterior satisfaz a parte antecedente de cada regra. Considerando o operador min, o grau de ativação de uma regra pode ser dado por:

$$\min\{\mu A(x_1), \mu C(x_2)\}, \tag{2}$$

sendo x_1 e x_2 as varáveis de entrada, A e C os conjuntos e μ o grau de pertinência das variáveis em seus respectivos conjuntos. A terceira etapa é a implicação do consequente, que é a segunda parte da regra $SE-ENTAO$. Nesta etapa define-se o consequente de acordo com o grau de ativação encontrado no antecedente. A Figura 3 exemplifica o grau de ativação de regras *fuzzy* com seus respectivos antecedentes e consequentes. A última etapa da inferência é a agregação, em que o resultado final será dado pela combinação do resultado de todas as regras. Essa agregação normalmente é dada pelo operador máximo ($\vee$).

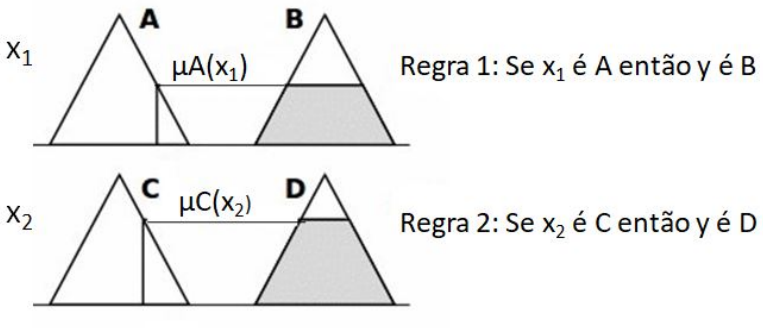

Figura 3 – Exemplo de regras *fuzzy* com seus respectivos graus de ativação.

Após o processo de inferência, é necessário fazer a defuzzificação do resultado, que transforma a saída *fuzzy* novamente em valores precisos para ser retornado para o usuário. A defuzzificação pode ser realizada por meio de vários métodos, como por exemplo:

- Método min-max: a regra com maior força de disparo é selecionada e determina qual função de consequente é ativada.
- Método da média: é calculada a média da força de ativação de cada regra de forma que todas as regras ativadas tenham influência na saída.

- Método raiz-soma-quadrado: cada função de pertinência é dimensionada de modo que o pico da função seja igual à força máxima de disparo que corresponde a essa função.
- Método do centro de gravidade recortado: cada função de pertinência é recortada nas forças de ativação da regra correspondentes.

Além dos modelos linguísticos, propostos inicialmente por Mamdani e Assilian (1975), posteriormente, foram propostos por Takagi e Sugeno (1985) os modelos funcionais, nos quais a parte do consequente das regras é uma função $f(x)$ das variáveis dos antecedentes. A função $f(x)$ é, em geral, um polinômio cujo o grau define a ordem do modelo, ou seja, se for uma constante o modelo é de ordem $zero$ ou se for uma função linear o modelo é de ordem um. Segundo Johansen, Shorten e Murray-Smith (2000) o modelo funcional de Takagi-Sugeno permite tratar dados não-lineares a partir das relações lineares locais apresentadas por cada regra, o que o torna mais simples que outros modelos de inferência *fuzzy*, facilitando assim a modelagem. Este modelo, que é base para os modelos *Neuro-Fuzzy* que são objeto deste trabalho, será apresentado na Seção 2.1.1.

2.1.1 O Modelo de Takagi-Sugeno

O modelo Takagi-Sugeno (TS) é uma das ferramentas mais úteis de inferência *fuzzy* para modelagem de sistemas dinâmicos e não-lineares (MATÍA; MARICHAL; JIMÉNEZ, 2014). Como em qualquer outro Sistema *Fuzzy*, as variáveis de entrada devem ser divididas em conjuntos *fuzzy* dentro do universo de discurso de cada variável. O universo de discurso representa o intervalo de valores que essas variáveis podem assumir. Neste modelo, as regras são funções matemáticas que relacionam os antecedentes e os consequentes. Desta forma, os antecedentes das regras são valores *fuzzy*, já os consequentes são valores funcionais. O exemplo a seguir apresenta a estrutura de regras de um modelo TS com n entradas e m regras, em que X é a entrada, A_{ij} é o conjunto *fuzzy* j ao qual a entrada i pertence e y é o consequente definido pela função $f(x)$.

$$
\begin{array}{llll}
R_1 & Se & (x_1 \text{ é } A_{1,1}) e (x_2 \text{ é } A_{2,1}) e \ldots (x_n \text{ é } A_{n,1}) & Então \quad y_1 \text{ é } f(x_1, x_2, \ldots, x_n) \\
R_2 & Se & (x_1 \text{ é } A_{1,2}) e (x_2 \text{ é } A_{2,2}) e \ldots (x_n \text{ é } A_{n,2}) & Então \quad y_2 \text{ é } f(x_1, x_2, \ldots, x_n) \\
 & & . & \\
 & & . & \\
 & & . & \\
R_m & Se & (x_1 \text{ é } A_{1,m}) e (x_2 \text{ é } A_{2,m}) e \ldots (x_n \text{ é } A_{n,m}) & Então \quad y_n \text{ é } f(x_1, x_2, \ldots, x_n)
\end{array}
$$

A partir do grau de compatibilidade das variáveis de entrada com o conjunto *fuzzy* correspondente, obtêm-se o seu grau de pertinência, dado por $\mu_{A_{ij}}(x_i)$, onde $\mu_{A_{ij}}$ é o grau de pertinência da entrada i no conjunto j e x_i é a variável de entrada. Já o grau de ativação do antecedente da regra é dado pela Equação (3):

$$w = \prod \mu_{A_{ij}}(x_i), \tag{3}$$

em que $\mu_{A_{ij}}$ é o grau de pertinência da entrada i no conjunto j e x é a variável de entrada.

Os consequentes são dados por funções dos antecedentes. O grau dessas funções define a ordem do modelo. Em modelos de ordem $zero$ os consequentes são constantes, em modelos de ordem um os consequentes são lineares. A partir das relações lineares locais representadas por cada regra, é possível estabelecer relações não-lineares globais, o que torna esse modelo eficiente para tratar funções não-lineares. O fato de os consequentes serem dados por funções matemáticas faz com que este modelo apresente um controle mais dinâmico que o modelo de Mamdani (FULCHER, 2008).

No modelo de Takagi-Sugeno, a saída final (y) é dada pela média ponderada das saídas individuais e seus respectivos graus de ativação, tendo, assim, uma combinação não-linear dos modelos lineares locais de cada regra (TANAKA; WANG, 2004)

$$y = \frac{\sum w_i y_i}{\sum w_i} \tag{4}$$

em que y é a saída final do sistema, y_i é a saída individual da regra i e w_i é o grau de ativação da regra i. Como o resultado de cada regra é um valor numérico, preciso e não um conjunto *fuzzy*, este modelo dispensa o processo de defuzzificação.

Este modelo permite extrair conhecimento de uma base de dados onde, a partir de um conjunto de dados de entrada e saída, os conjuntos *fuzzy* e os parâmetros do consequente são estimados conforme uma medida de desempenho definida pelo usuário. Normalmente, para definição das regras utiliza-se algoritmos para aproximação de funções, que utilizam índices de desempenho definidos pelo usuário para encontrar as relações entre as variáveis e definir as funções dos consequentes. Os pesos e demais parâmetros do sistema são obtidos e otimizados por um processo de minimização do erro entre a saída retornada pelo modelo e a saída esperada conforme a base de dados. Várias técnicas podem ser utilizadas para realizar esta minimização como, por exemplo, o método dos mínimos quadrados que consiste em tentar minimizar a soma do quadrado do erro gerado pela saída do modelo, ou algoritmos de agrupamento que

buscam agrupar dados baseados em seu grau similaridade. Outras técnicas, tanto exatas quanto heurísticas, podem ser utilizadas, principalmente ao se criar sistemas híbridos baseados no modelo Takagi-Sugeno, o que será discutido na Seção 2.1.2.

2.1.2 Redes *Neuro-Fuzzy*

Baseados no modelo de Takagi-Sugeno (TS) (TAKAGI; SUGENO, 1985), Redes *Neuro-Fuzzy* combinam características de Redes Neurais Artificiais (RNA) e Sistemas *Fuzzy* no intuito de melhorar o raciocínio e a inferência em uma máquina de aprendizado. São capazes de lidar a com imprecisão inerente aos Sistemas *Fuzzy* e incorporar a capacidade de aprendizagem e robustez das Redes Neurais (SHIHABUDHEEN; PILLAI, 2018). Uma Rede *Neuro-Fuzzy* eficiente deve obter as seguintes características:

- aprendizado rápido;
- adaptabilidade *on-line*;
- auto ajuste para obter o menor erro global possível;
- baixa complexidade computacional.

Ao construir Sistemas *Fuzzy* é necessário ter o conhecimento das regras para especificação dos conjuntos *fuzzy*. Porém quando há dados relacionados a um problema pode-se utilizar uma RNA, sendo necessário que o usuário apenas especifique a arquitetura e algoritmo de aprendizado a ser utilizado. Como as desvantagens das duas técnicas parecem complementares, torna-se natural utilizar a integração entre os dois para obter sistemas mais eficientes (ABRAHAM, 2001). Ao realizar essa hibridização pode-se considerar, de forma simples, que o aprendizado provido pela Rede Neural determina as funções e regras do Sistema de Inferência *Fuzzy*.

Para Fullér (1995) as Redes Neurais são eficientes para reconhecer padrões em dados, mas não para explicitar como alcançaram o resultado final ou quais fatores o influenciaram. Já os Sistemas *Fuzzy* conseguem lidar com informações imprecisas e permitem ter o conhecimento de como obtiveram o resultado final e que o influenciou, porém não têm a capacidade de construir automaticamente sua base de regras por meio da qual atingem este resultado.

O processo computacional inerente as Redes *Neuro-Fuzzy* inicia-se com o desenvolvimento de neurônios *fuzzy* com base nas características dos neurônios biológicos e em seguida incorporando técnicas de aprendizado (FULLÉR, 1995). Esse processo segue três etapas:

- desenvolvimento de modelos *fuzzy* neurais com base em neurônios biológicos;
- modelos de conexões sinápticas que incorporam imprecisão na rede;

- desenvolvimento do algoritmo de aprendizado (para ajuste dos pesos sinápticos)

Basicamente, existem duas possíveis formas de modelagem para Redes *Neuro-Fuzzy*. Na primeira o módulo *fuzzy* recebe varáveis linguísticas e fornece um vetor de valores numéricos para o módulo neural, que processará esses valores, executará processo de aprendizado e retornará o resultado final. Já no segundo modelo possível, o módulo neural recebe e processa os valores de entrada e aciona o mecanismo de inferência *fuzzy* que contém a base de regras.

O conjunto de regras e demais parâmetros do sistema podem ser obtidos com a utilização métodos de aprendizado, que extraem conhecimento a partir dos dados. Os principais encontrados na literatura são: métodos de aprendizado do tipo gradiente; técnicas híbridas; técnicas populacionais; técnicas de aprendizado extremo; máquina de vetor de suporte.

- **Métodos de aprendizado do tipo gradiente**: técnica baseada na descida em direção à região mais íngreme da vizinhança, ou seja, que proporciona melhor resultado.
- **Técnicas híbridas**: combinam duas ou mais técnicas de aprendizado como *back-propagation*, mínimos quadrados, algoritmos de agrupamento, entre outros.
- **Técnicas populacionais**: envolvem Algoritmos Genéticos, Evolução Diferencial, Algoritmos de Enxame, Colônias de Formigas, entre outros.
- **ELM (*Extreme Learning Machine* - Máquina de Aprendizagem Extrema)**: técnica de rede neural onde os parâmetros dos neurônios ocultos são definidos aleatoriamente enquanto os dos neurônios de saída são definidos analiticamente baseado no método dos mínimos quadrados.
- **SVM (*Support Vector Machine* - Máquina de Vetor de Suporte)**: algoritmos de classificação que tem sua decisão guiado por pontos vetoriais de suporte. É uma das principais técnicas de classificação na área de aprendizado de máquina.

Essas técnicas e os principais trabalhos que as utilizam para aprendizado em Redes *Neuro-Fuzzy* são apresentados em Shihabudheen e Pillai (2018). Os principais modelos conhecidos na literatura são discutidos em Abraham (2001) que os classifica em três categorias: modelos concorrentes, cooperativos e totalmente fundidos. Já o modelo *Neo-Fuzzy-Neuron*, utilizado neste trabalho, será apresentado no Seção 2.1.3.

2.1.3 *Neo-Fuzzy-Neuron* (NFN)

Esta Seção detalha a rede *Neo-Fuzzy-Neuron* (NFN) (YAMAKAWA et al., 1992) que será utilizada para implementação dos algoritmos apresentados no Capítulo 3. O NFN é capaz de tratar a não linearidade entre dados por meio de uma conjunção

de funções lineares (MIKI; YAMAKAWA, 1999). A vantagem deste tipo de rede é o rápido aprendizado, que pode ser até 100 vezes menor do que o de uma rede neural multicamada convencional e a garantida de convergência ao ótimo global (YAMAKAWA et al., 1993). A estrutura da rede NFN é apresentada na Seção 2.1.3.1. A Seção 2.1.3.2 trata das funções de pertinência utilizadas no NFN. Já o processo de aprendizado e ajuste de pesos é detalhado na Seção 2.1.3.3.

2.1.3.1 Estrutura da rede NFN

Neo-Fuzzy-Neuron é uma Rede *Neuro-Fuzzy* composta por um conjunto de n modelos do tipo Takagi-Sugeno (TS) (TAKAGI; SUGENO, 1985) de ordem $zero$, um para cada variável de entrada. No NFN a saída de cada modelo individual y_{ti} é dada por um conjunto de m_j regras, e cada regra é representada por uma função de pertinência. A estrutura básica da rede NFN é ilustrada na Figura 4, na qual x_{ti} são as variáveis de entrada no instante t, q_{ij} são os parâmetros do consequente associados a cada função de pertinência que define cada regra, y_{ti} é a saída individual de cada modelo e $\hat{y}_t$ é a saída da rede.

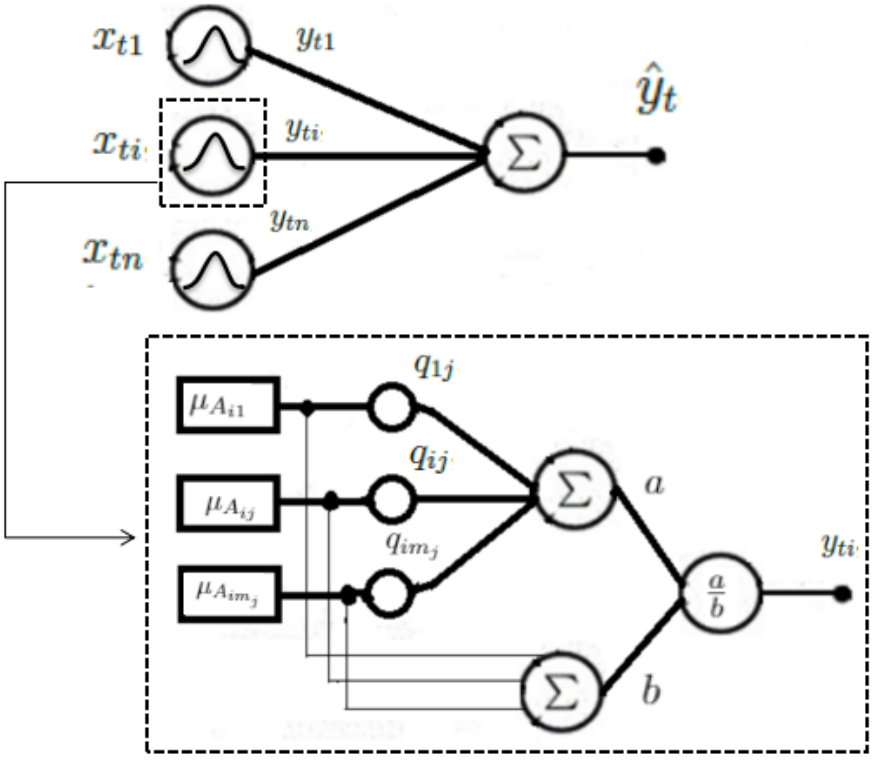

Figura 4 – Estrutura da Rede *Neuro-Fuzzy Neo-Fuzzy-Neuron*. Adaptado de (SILVA et al., 2014)

A saída do NFN é a soma dos modelos individuais e é obtida no instante t $(\hat{y}_t)$ por:

$$\hat{y}_t = \sum_{i=1}^{n} y_{ti}. \tag{5}$$

Para cada variável x_{ti} do universo de discurso, a partição do espaço de entrada é realizada por m_j funções de pertinência. A partir dessa partição do espaço de entrada tem-se as seguintes m_j regras:

$$\begin{array}{llll} R_1 & Se & x_{ti}\,\acute{e}\,A_{i1} & Então \quad y_{ti}\,\acute{e}\,q_{i1} \\ R_2 & Se & x_{ti}\,\acute{e}\,A_{i2} & Então \quad y_{ti}\,\acute{e}\,q_{i2} \\ & & . & \\ & & . & \\ & & . & \\ R_j & Se & x_{ti}\,\acute{e}\,A_{ij} & Então \quad y_{ti}\,\acute{e}\,q_{ij} \end{array}$$

Os modelos são desacoplados e as saídas individuais y_{ti} são obtidas pelo grau de ativação de cada regra $\mu_{A_{ij}}(x_i)$ multiplicado pelo seu respectivo parâmetro q_{ij} (a), ponderados pelo somatório do grau de ativação de todas as regras (b), conforme apresentado na Equação (6):

$$y_{ti} = \frac{a}{b} = \frac{\sum_{j=1}^{m_i} \mu_{A_{ij}}(x_i) q_{ij}}{\sum_{j=1}^{m_i} \mu_{A_{ij}}(x_i)}, \tag{6}$$

onde, i indexa as variáveis de entrada, j as funções de pertinência e t é o instante de tempo.

2.1.3.2 Funções de Pertinência

As funções de pertinência utilizadas na NFN proposta por Yamakawa et al. (1992) são Triangulares. Outros tipos de funções de pertinência também podem ser utilizados, como funções Gaussianas ou Trapezoidais (BODYANSKIY; KOKSHENEV; KOLODYAZHNIY, 2003; ZAYCHENKO; GASANOV, 2012). Neste trabalho optou-se pelo uso de funções Gaussianas por ser o tipo de função mais adequada para representar dados incertos (KREINOVICH; QUINTANA; REZNIK, 1992). Uma função de pertinência Gaussiana é definida pelo centro (c) e pelo espalhamento (s), como pode ser visto na Figura 5.

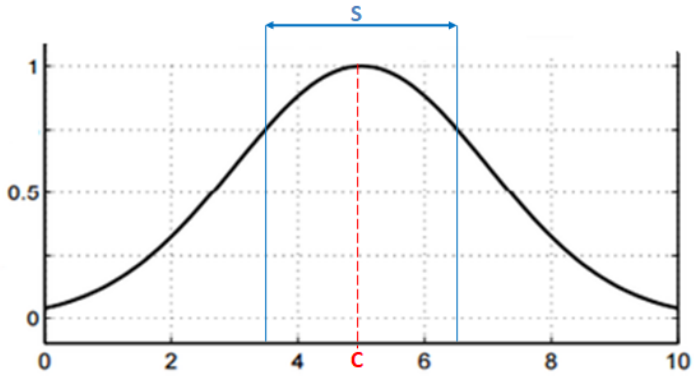

Figura 5 – Função de pertinência Gaussiana.

O grau de ativação $\mu_{A_{ij}}(x_i)$ de uma regra j para uma variável x_i retornado por uma função de pertinência Gaussiana é dado por:

$$\mu_{A_{ij}}(x_i) = exp\left(-\frac{1}{2}\left(\frac{x_i - c_i}{s_i}\right)^2\right). \tag{7}$$

As funções de pertinência Gaussianas podem ser uniformemente espaçadas ou não-uniformemente espaçadas. Funções uniformemente espaçadas consideram que todos as regras abrangem o mesmo intervalo de valores para o universo de discurso da variável em questão. Para isso é necessário particioná-lo em m_i intervalos iguais, sendo m_i o número de funções de pertinência para a variável x_i. Desta forma a distância entre o centro de cada função (Δ) é a mesma e o valor de s para cada função também é o mesmo, conforme mostrado na Figura 6, onde o universo de discurso é particionado em 5 funções.

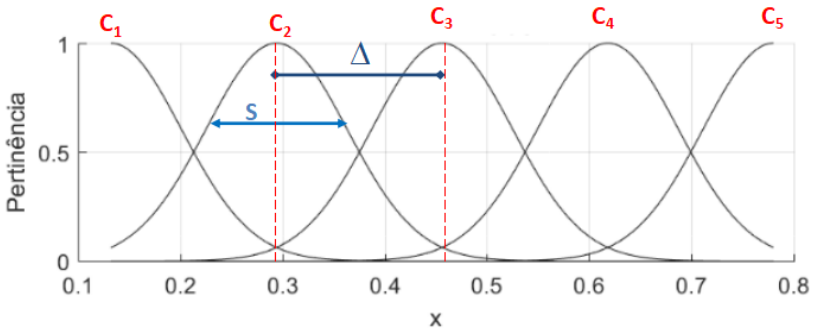

Figura 6 – Funções de pertinência Gaussianas uniformemente espaçadas.

Nota-se que o centro da primeira função de pertinência coincide com o limite inferior da variável x_i e o centro da última função com o limite superior. O valor de Δ é dado pela

Equação (8) e o valor de c para cada função pela Equação (9):

$$\Delta_i = \frac{(max_{x_i} - min_{x_i})}{m_i - 1}, \tag{8}$$

$$c_{ij} = min_{x_i} + (j-1)\Delta_i, \tag{9}$$

em que, i indexa a variável de entrada, m_i é o número de funções de pertinência para a variável i, j indexa a função de pertinência, max_{x_i} é o limite superior da variável x_i, min_{x_i} é o limite inferior da variável x_i e m_i é o número de funções de pertinência para a variável i.

A definição das funções uniformemente espaçadas é mais simples, porém os valores de c e s utilizados para essa definição são muito específicos, o que pode prejudicar a estimativa de saída, pois o nível de informação do sistema pode não ser adequado para utilizar valores tão exatos (ROSS, 2005). Por este motivo, são utilizadas também funções não-uniformemente espaçadas, como pode ser visto na Figura 7.

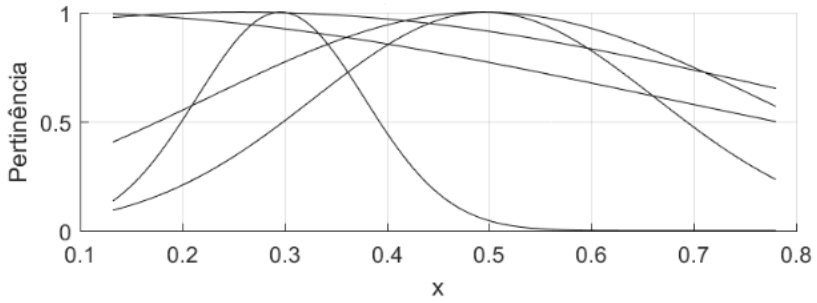

Figura 7 – Funções de pertinência Gaussianas não-uniformemente espaçadas.

Como pode ser observado na Figura 7, não existe uma definição rígida para os valores de c e s, sendo que o valor de c pode estar entre quaisquer valor dentro do universo de discurso e s pode assumir qualquer valor. Não há necessidade de calcular Δ neste caso, já que não existe uma distância específica definida para cada função. As funções não-uniformemente espaçadas podem ser consideradas funções generalizadas pois possibilitam sua otimização para o conjunto de informação disponível (KLIR; YUAN, 1996).

2.1.3.3 Ajuste de Parâmetros

O procedimento para adaptação dos parâmetros dos consequentes da rede NFN, isto é, o treinamento, é realizado por um método baseado no Gradiente. O aprendizado é supervisionado e tem o propósito de ajustar os parâmetros q_{ij}. Outros procedimentos,

tais como, atualização em lote e atualização incremental com termo de *Momentum*, foram apresentados em (YAMAKAWA et al., 1992). O procedimento de atualização dos parâmetros descrito a seguir consiste em atualizar seus respectivos valores a cada época de treinamento a partir de uma taxa de aprendizado (α) previamente definida. A fórmula para atualização dos parâmetros pode ser vista na Equação (10):

$$q_{ij} = q_{ij} - \alpha(e_t)(x_{ti})(d_{ij}), \tag{10}$$

na qual α é a taxa de aprendizado, e_t é o erro de modelagem obtido por y_t - $\hat{y}_t$, x_{ti} é a amostra de dados no instante t para a variável i e d_{ij} é obtido por:

$$d_{ij} = \frac{\mu_{A_{ij}}}{\sum_{j=1}^{m_i} \mu_{A_{ij}}}, \tag{11}$$

em que m_i é o número de funções de pertinência para a variável i.

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

Algoritmo 1: Algoritmo do NFN

```
Entrada x_t, y_t, n;
Saida ŷ_t;
for epoca = 1 : l do
    for t = 1,2,... do
        Ler x_t, y_t;
        for j = 1 : m do
            for i = 1 : n do
                Calcular d_ij;
                Atualizar q_ij;
            end
        end
        Calcular ŷ_t;
        Calcular e_t;
    end
end
```

Diversas técnicas baseadas na proposta inicial de Yamakawa et al. (1992) foram propostas e utilizadas para aprendizado em redes NFN, sendo os métodos do Gradiente e dos mínimos quadrados os mais encontrados na literatura. A próxima Seção apresenta uma revisão da literatura abordando trabalhos relevantes que exploram diferentes algoritmos de aprendizado para redes NFN.

2.1.4 Revisão da literatura sobre *Neo-Fuzzy-Neuron*

Uma rede *Neo-Fuzzy-Neuron* é utilizada em Soualhi et al. (2013) para previsão de degradação de rolamentos. O modelo utiliza funções de pertinência Triangulares criadas pelo particionamento uniforme do universo de discurso. O aprendizado consiste na adaptação e ajuste dos pesos de conexão entre os neurônios a partir de um algoritmo de aprendizagem incremental. Já Bodyanskiy, Pliss e Vynokurova (2013) apresenta uma modificação na NFN que o torna mais flexível, utilizando aprendizagem adaptativa para ajuste dos parâmetros e do tipo das funções de pertinência. O autor apresenta uma nova fórmula para definição das funções de pertinência, onde existem parâmetros nesta fórmula que definem o tipo da função (Gaussiana, Trapezoidal ou Triangular). Desta forma o algoritmo define não só os parâmetros das funções mas também o seu tipo. Além disso é utilizado um algoritmo de particionamento para evitar lacunas no universo de discurso. Os pesos são ajustados por um método do Gradiente Descendente.

O trabalho de Bodyanskiy et al. (2015) apresenta um Sistema híbrido *Neuro-Neo-Fuzzy*, que combina uma rede Takagi-Sugeno-Kang (SUGENO; KANG, 1988) e uma rede do tipo *Neo-Fuzzy-Neuron* com funções de base radial para resolução de problemas de mineração de dados que envolvem dados não-lineares, estocásticos, estacionários e caóticos com processamento em tempo real. O modelo possui três camadas para processamento das informações. A camada de entrada processa as variáveis de entrada que serão fuzzificadas pela primeira camada oculta, que contém as funções de pertinência. A segunda camada oculta realiza a agregação dos níveis de pertinência calculados na camada anterior e, por fim, a terceira camada é composta por uma rede *Neo-Fuzzy-Neuron* que retorna a saída do sistema. As funções de pertinência da primeira camada são Gaussianas, já as da camada NFN são Triangulares, complementares e uniformemente espaçadas. Para ajuste dos pesos é utilizado um método de mínimos quadrados recorrentes exponencialmente ponderados para entradas lineares e um método baseado no Gradiente com taxa de aprendizado ótima para entradas caóticas e não-estacionárias. Já Todorov e Terziyska (2015) descreve uma abordagem simples de *Backpropagation* heurístico para redes NFN em cascata aplicadas a séries temporais. O modelo utiliza funções Gaussianas uniformemente espaçadas e o aprendizado consiste em ajustar os pesos da rede por *Backpropagation* por meio de uma heurística baseada no método do Gradiente clássico. A abordagem utiliza uma taxa de aprendizado local para cada peso que varia conforme o sinal do Gradiente.

Zurita et al. (2016) apresenta uma rede *Neo-Fuzzy-Neuron* aplicada à modelagem de séries temporais industriais. As funções de pertinência utilizadas são Triangulares e complementares obtidas pelo particionamento uniforme do universo de discurso. O processo de aprendizado consiste em adaptar os pesos da rede, utilizando uma

abordagem de aprendizado incremental utilizando o método do Gradiente Descendente com *Backpropagation*. No trabalho de Bodyanskiy, Tyshchenko e Kopaliani (2016) é proposta uma rede *Neo-Fuzzy-Neuron* (NFN) estendida. Essa rede consiste em uma rede NFN com aprendizado adaptativo por *Backpropagation* que possui capacidade de rastreamento e suavização. Nesta abordagem o universo de discurso é particionado uniformemente por funções Triangulares e complementares. O ajuste dos pesos é realizado por um algoritmo adaptativo baseado no Gradiente. Já Hu, Bodyanskiy e Tyshchenko (2017) utilizam a mesma estrutura e apresenta uma arquitetura e métodos para redes NFN profundas com ajuste de pesos *online*. As funções de pertinência são Triangulares e complementares, criadas a partir de partições uniformes. Os pesos são exponenciais e o ajuste é realizado por um método de mínimos quadrados recorrentes ponderados com aprendizado adaptativo e em cascata.

Uma abordagem simples e automática para projetar redes *Neo-Fuzzy-Neuron* para identificação é proposta por Mendes et al. (2017). O autor utiliza um algoritmo de adequação para aprender múltiplos modelos Takagi-Sugeno de ordem $zero$, sendo cada modelo um neurônio da NFN. A rede é composta por funções Triangulares e complementares criadas com o particionamento uniforme do universo de discurso. Já os parâmetros do consequente são obtidos a partir de um algoritmo de mínimos quadrados com *Backfitting*. Em Setlak et al. (2018) é proposta uma rede NFN em cascata convexa. A arquitetura em cascata permite treinar cada pilha de neurônios de forma independente. Além disso o modelo considera o *Neo-Fuzzy-Neuron* generalizado, uma variação multidimensional do NFN. As funções de pertinência são criadas pelo particionamento uniforme do universo de discurso na forma Triangular e complementar. O treinamento consiste em ajustar os pesos sinápticos da rede e é realizado por um método baseado no Gradiente. Este método permite treinar cada cascata de neurônios de forma independente. A velocidade de aprendizado pode ser aumentada utilizando-se o método dos mínimos quadrados recursivos padrão para adaptar a taxa de aprendizado.

Uma estrutura do *Neo-Fuzzy-Neuron* estendida é utilizada no trabalho de Bodyanskiy, Kulishova e Malysheva (2018) para reconhecimento de expressões faciais. As funções de pertinência utilizadas são do tipo *B-Spline* (WANG et al., 1994), que são funções contínuas com curvaturas mais suaves. Estas funções foram criadas por partição uniforme. É utilizado um algoritmo baseado no Gradiente para adaptação dos pesos que são exponenciais. Para tornar o aprendizado mais rápido utilizou-se um critério de aprendizado de entropia que se demonstra mais eficiente quando utilizada para esse tipo de função de pertinência. Em Bodyanskiy et al. (2018) é proposto um codificador automático baseado em uma rede *Neo-Fuzzy-Neuron* generalizada com o intuito de reduzir o número de funções de pertinência e, consequentemente, o tempo de

processamento. São utilizadas funções Triangulares e complementares obtidas pelo particionamento uniforme do universo de discurso. O modelo é alimentado por n pesos não-lineares multidimensionais que são ajustados por um método baseado no Gradiente com *Backpropagation*. A escolha da taxa de aprendizagem se dá por um método de mínimos quadrados recorrentes na forma ótima (com apenas um passo) com propriedades de rastreamento e suavização e permite a minimização do número de funções de pertinência.

2.2 Computação Evolucionária

Computação Evolucionária é a área da computação inspirada nos processos de evolução natural dos seres vivos, onde o indivíduo mais forte tende a sobreviver. A Computação Evolucionária utiliza analogias ao processo de evolução como seleção natural, cruzamento e mutação para gerar e evoluir uma população de soluções, onde cada indivíduo (solução) compete pela sobrevivência (KELLER DERONG LIU, 2016). As definições e princípios básicos de um algoritmo evolucionário são:

- **Indivíduo**: é representação de uma solução para o problema. Pode mudar de acordo com o problema e é de extrema importância escolher a representação adequada para o que está se tratando.
- **Gene**: são as informações que compõe um indivíduo. Por exemplo, se um indivíduo é representado por um vetor binário, cada índice do vetor representa um gene.
- **População**: é um conjunto de n soluções para o problema. O tamanho n da população (por quantas soluções será composta) é pré-definido e é interessante existirem soluções boas e ruins em uma mesma população para favorecer o processo evolutivo.
- **Reprodução**: replicação de uma solução para a próxima população, sem alterações.
- ***Fitness***: é a aptidão de uma solução, ela é uma função que determina o quanto uma solução é boa. Quanto maior a *fitness*, melhor é a solução.
- **Cruzamento**: combinação de duas soluções (soluções pai) para formar uma nova solução (solução filha). Essencial para o processo de melhoria de soluções já que os pais com melhor aptidão (*fitness*) terão maior probabilidade de se reproduzir.
- **Mutação**: é o processo que altera algum gene do indivíduo aleatoriamente. Este processo tem como objetivo inserir diversidade na população. Quanto maior a diversidade, maior o espaço de busca que está sendo explorado.
- **Seleção natural**: é o processo de escolha dos indivíduos que passarão para a próxima população. Este processo é probabilístico e os indivíduos com maior

fitness possuem maior chance de sobreviver na próxima população. Acontece após o cruzamento (geração de filhos) e a mutação.

As principais técnicas utilizadas nessa área são Algoritmos Genéticos, Programação Genética, Estratégias Evolutivas e Evolução Diferencial. Todas elas seguem estes mesmos princípios, com algumas diferenças na representação dos indivíduos ou em algum detalhe no processo de evolutivo do algoritmo. Enquanto no Algoritmo Genético os indivíduos são normalmente vetores simples em que cada um representa diretamente uma solução para dado problema, na Programação Genética trabalha-se com uma população de programas computacionais que levam à solução final. Por outro lado, as Estratégias Evolutivas e a Evolução Diferencial tem sua principal diferença no processo de evolução das soluções. Enquanto no primeiro os pais sobrevivem até que gerem filhos necessariamente melhores, no segundo as operações de cruzamento e mutação são realizadas a partir de perturbações nos indivíduos, onde cada casal gerará um novo filho em um local diferente do espaço de busca. Dentre essas técnicas a Programação Genética tem como vantagem a capacidade de trabalhar com programas computacionais interpretáveis, o que proporciona a criação de algoritmos generalistas que sejam capazes de resolver problemas genéricos, necessitando de poucas adaptações. A Seção 2.2.1 apresenta mais detalhes sobre a Programação Genética e sua modelagem.

2.2.1 Programação Genética

Programação Genética (PG) é uma técnica para gerar e evoluir programas computacionais automaticamente (KOZA, 1992). Surgiu com o objetivo de fazer com que computadores pudessem aprender a resolver problemas sem serem explicitamente programados para isso, o que torna o método bastante genérico e sem necessidade de grandes adaptações para trabalhar com problemas distintos. Se difere de outras técnicas evolucionárias e de inteligência computacional pelo fato de apresentar soluções em forma de programas computacionais interpretáveis (KOZA, 1992).

A PG é um método populacional, ou seja, cria e evolui uma população de indivíduos (programas), que normalmente são representados por árvores de sintaxe (POLI; LANGDON; MCPHEE, 2008) compostas por funções (nós internos) e terminais (nós folha), conforme mostrado na Figura 8. O termo programa, neste caso, deve ser entendido de forma mais ampla, podendo se referir, por exemplo, a uma expressão matemática, onde as funções são operações aritméticas e os terminais são as variáveis e constantes do problema, ou uma sequência de instruções, onde as funções são os operadores e os terminais são os atributos. Cada função e cada terminal representam um *gene* do indivíduo. Cada indivíduo representa uma possível solução para o problema.

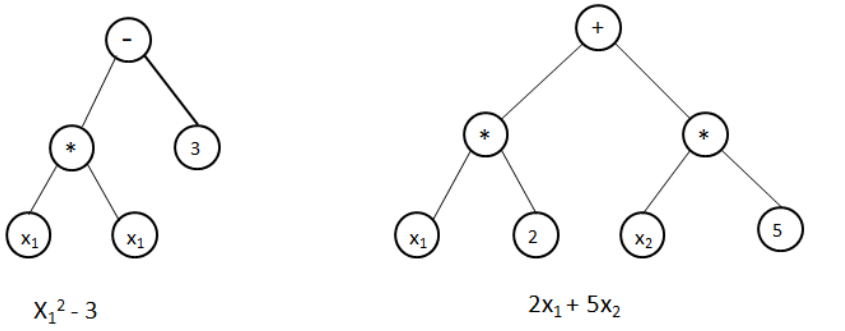

Figura 8 – Exemplo de indivíduos representados por árvores de sintaxe.

O processo evolucionário de uma PG inicia-se com a geração de uma população inicial, normalmente aleatória, na qual são selecionadas funções e terminais para compor cada indivíduo. A evolução ocorre por meio de processos análogos aos processos de evolução natural de organismos vivos, como qualquer outra técnica evolucionária. A cada geração os indivíduos são avaliados conforme uma função de aptidão (*fitness*) e os melhores têm mais chances de serem selecionados para as operações de cruzamento, reprodução e mutação. A cada iteração uma nova população é gerada pelo processo de seleção natural. Na geração da nova população pode-se substituir todos os indivíduos da última geração pelos filhos gerados (modo geracional), ou pode-se utilizar um método de seleção que considere a *fitness* para selecionar quais indivíduos devem sobreviver para a próxima geração. Nos dois casos corre-se o risco de perder o melhor indivíduo, já que os processos são probabilísticos. Para resolver este problema é possível utilizar o elitismo, mecanismo que garante que o melhor indivíduo seja mantido para a próxima população. Todos esses passos são repetidos até atender a um critério de parada, que pode ser um valor pré-definido de *fitness* para o melhor indivíduo ou um número determinado de gerações (BANZHAF et al., 1998). A Figura 9 ilustra este processo.

Devido a facilidade em representar funções matemáticas conforme citado anteriormente, a PG tem se demonstrado uma poderosa ferramenta para regressão simbólica de funções como discutido em Azad e Ryan (2014) e Oliveira et al. (2015). Segundo Koza (1994), em problemas de Aprendizado de Máquina e Inteligência Artificial, a representação mais natural para uma solução é um programa de computador, isto é, uma composição hierárquica de funções primitivas e terminais com tamanho variável, ao contrário de representações em cadeias simples de tamanho fixo, utilizadas por outras abordagens, que torna a representação mais difícil, não natural e excessivamente restritiva.

Para obter uma representação eficiente das soluções, a modelagem de uma PG deve ser feita com cuidado e exige três propriedades básicas para os indivíduos gerados

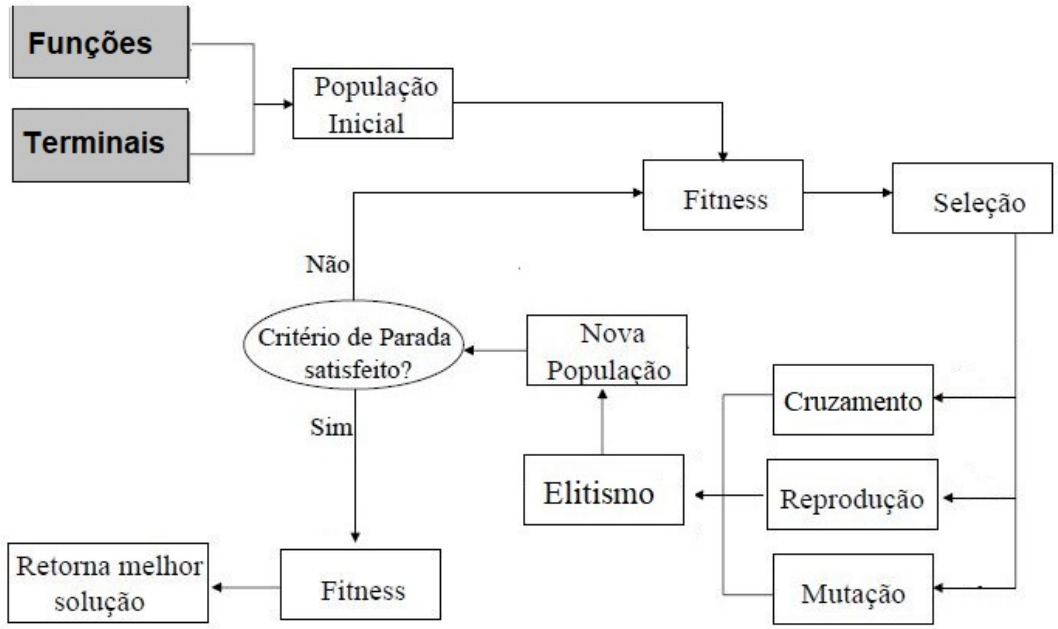

Figura 9 – Fluxo de execução de um algoritmo de Programação Genética.

(KOZA, 1992):

- **Suficiência**: a expressão representada pelo indivíduo deve ser suficiente para representar uma solução candidata para o problema em questão. Ou seja, todos os indivíduos devem representar uma solução candidata completa para o problema.
- **Fechamento**: uma função deve aceitar como entrada qualquer saída produzida por qualquer outro elemento do conjunto de funções ou do conjunto de terminais.
- **Parcimônia**: conter apenas funções realmente necessárias para resolver o problema em questão. Esta função é desejada, mas não obrigatória.

2.2.1.1 Geração da População Inicial

A população inicial de uma PG normalmente é gerada de forma aleatória. A geração das árvores de representação dos indivíduos requer cuidados devido à sua estrutura complexa. É importante que haja diversidade para representar uma boa amostra do espaço de busca, o que favorece a convergência do algoritmo. Por outro lado, também é importante que se imponha limites para o tamanho das árvores, para que a população não possua indivíduos de grande complexidade. Normalmente, ao se criar os indivíduos, impõe-se esse limite, que é geralmente especificado por uma profundidade máxima, podendo-se também especificar uma profundidade mínima. Uma outra abordagem consiste em limitar o tamanho ao invés da profundidade das árvores (POLI; LANGDON; MCPHEE, 2008). Os principais métodos para geração da população inicial da PG são:

- **Método Grow**: cria árvores de profundidade variável, respeitando uma profundidade máxima definida. Os nós são gerados aleatoriamente entre funções e terminais. O processo termina quando é gerado um terminal ou atingida a profundidade máxima.
- **Método Full**: cria árvores completas de mesma profundidade. Determina-se uma profundidade máxima e são escolhidas funções enquanto a profundidade do indivíduo for inferior à máxima estipulada. Ao atingir a profundidade máxima, são escolhidos terminais e encerra-se o processo.
- **Método Ramped-Half-and-Half**: é o mais recomendado por permitir maior diversidade dos indivíduos. Metade da população é gerada pelo método *Full* e a outro metade pelo método *Grow*, seguindo uma faixa de valores para profundidade máxima. Desta forma são criados indivíduos de formas e tamanhos diferentes.

Mesmo após a geração da população inicial, o controle do tamanho das árvores também é essencial, pois ao longo do processo evolutivo há um crescimento inevitável neste tamanho o que pode trazer prejuízos ao algoritmo principalmente com relação à performance (LANGDON, 2000). Esse controle pode ser feito impondo-se limites na profundidade ou crescimento dos programas para as próximas populações ou aplicando-se uma penalidade na medida de *fitness* de programas maiores, fazendo com que tenham menos chance de serem selecionados para as gerações futuras.

2.2.1.2 Métodos de Seleção

Os operadores genéticos em uma PG são aplicados baseado na função *fitness*, ou seja, indivíduos mais aptos tem mais chance de serem selecionados. A seleção ocorre em dois momentos: seleção de pais e seleção de indivíduos para a próxima geração (seleção natural). A seleção de pais escolhe dois indivíduos para se reproduzir e gerar soluções filhas. Este processo se repete até gerar uma quantidade estipulada de soluções filhas. Depois disso é necessário que se selecione os indivíduos que sobreviverão para a próxima geração. Nos dois casos indivíduos com maior aptidão têm mais chance de serem escolhidos. Porém é necessário que também sejam escolhidos indivíduos piores para manter a diversidade da população, o que é um fator importante para o processo evolutivo.

Os dois métodos mais comuns são seleção por torneio e roleta (EIBEN; SMITH et al., 2003). Na seleção por torneio, n indivíduos são sorteados para o torneio e o mais apto vence, sendo escolhido como primeiro pai. Realiza-se outro torneio e é escolhido o segundo pai. O sorteio é randômico, mas o pai escolhido será sempre o mais apto entre os sorteados. Já na seleção por roleta os indivíduos são dispostos no espaço (roleta) proporcional à sua *fitness*, de forma que os que tiverem maior aptidão ocupem uma fatia maior da roleta. É feito o sorteio do indivíduo selecionado a partir do espaço

definido pelas aptidões.

2.2.1.3 Operadores Genéticos

Para evoluir, ou seja, melhorar a qualidade das soluções geradas, são aplicados os operadores genéticos para modificá-las durante o processo evolutivo. Os operadores normalmente utilizados, segundo Banzhaf et al. (1998), são cruzamento, mutação e reprodução.

O cruzamento é essencial no processo evolutivo. O objetivo desse operador é gerar soluções possivelmente melhores (soluções filhas) a partir de duas outras soluções (soluções pai). Parte-se do pressuposto que ao combinar dois indivíduos existe uma grande probabilidade de o indivíduo gerado ser melhor, pois será composto por informações genéticas, possivelmente boas, dos dois pais. O cruzamento possibilita a convergência do algoritmo para regiões mais promissoras no espaço de busca. Na PG o cruzamento é realizado por meio de trocas de sub-árvores conforme mostrado na Figura 10. Seleciona-se um nó aleatoriamente e as sub-árvores cuja raiz corresponde a este nó no pai 1 é trocada com a sub-árvores selecionada no pai 2.

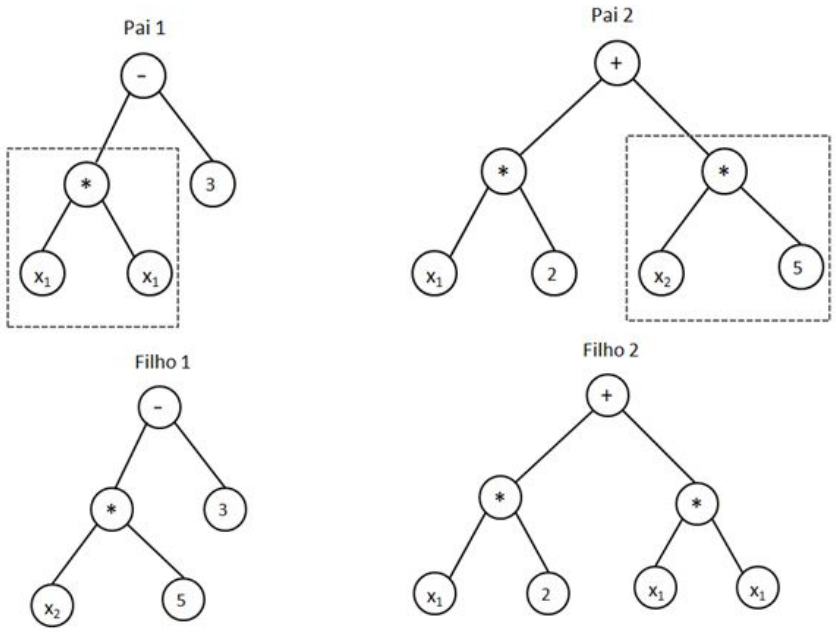

Figura 10 – Exemplo de operação de cruzamento na Programação Genética.

A mutação consiste em alterar alguma informação genética de um indivíduo selecionado. Isso proporciona um maior grau de diversidade na população, fazendo com que os indivíduos gerados não se restrinjam apenas às informações genéticas dos pais. Este processo permite uma maior exploração do espaço de busca. A mutação em uma PG pode ser realizada pela troca de uma sub-árvore selecionada por outra sub-árvore gerada aleatoriamente, pela troca de uma função por outra função ou terminal, ou de

um terminal por outro terminal ou por funções geradas aleatoriamente. A Figura 11 mostra um exemplo de mutação de um ponto.

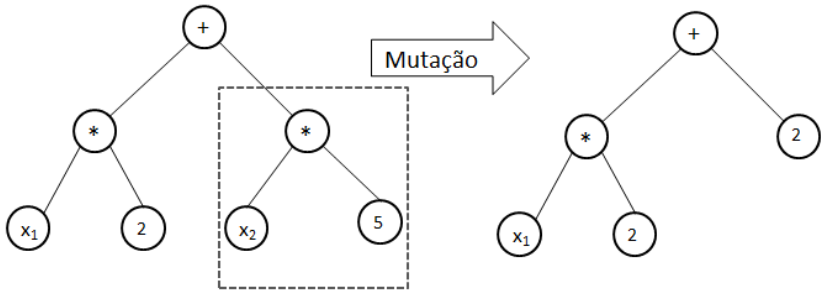

Figura 11 – Exemplo de operação de mutação de um ponto na Programação Genética.

Já o operador de reprodução, diferente dos outros não realiza nenhuma alteração no indivíduo. Consiste em simplesmente replicar os pais selecionados para próxima geração.

Os operadores são aplicados baseados em uma probabilidade pré-definida pelo usuário para cada um deles. Normalmente o operador de cruzamento possui alta probabilidade de ser aplicado (em torno de 90%), já o de mutação tem probabilidade mais baixas (em torno de 1%). A probabilidade restante para completar os 100% corresponde à probabilidade de reprodução (POLI; LANGDON; MCPHEE, 2008).

2.2.2 Programação Genética Muti-Gene

A Programação Genética Muti-Gene (PG-MG) (FERREIRA, 2001) é uma variação da Programação Genética que considera que um indivíduo possui várias árvores em sua estrutura conectadas por uma estrutura linear de genes superiores conforme mostrado na Figura 12.

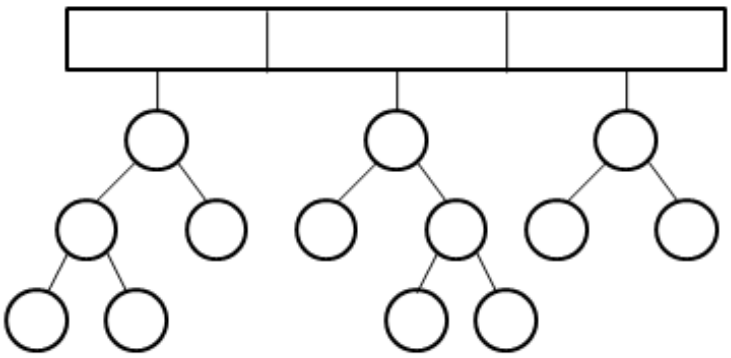

Figura 12 – Exemplo de indivíduo na Programação Genética Multi-Gene.

A estrutura de cada árvore que compõe o indivíduo é a mesma utilizada na Programação Genética tradicional porém o indivíduo como um todo é mais complexo visto que é formado por um conjunto de árvores. A saída final é a agregação das árvores ligadas a cada gene superior que pode ser dada por uma combinação linear, média aritmética, somatório ou outro operador de agregação (KOZA, 1992). Percebe-se que se houver apenas 1 gene superior, a estrutura será exatamente a mesma da PG tradicional.

As definições e operações de seleção, mutação e reprodução na PG-MG são idênticos ao da PG tradicional discutidos na Seção 2.2.1. Já a operação de cruzamento apresenta uma pequena diferença, podendo ser realizada em alto ou baixo nível. O cruzamento em alto nível ocorre quando troca-se um gene superior de um pai com um gene superior do outro. Quando o cruzamento ocorre nas sub-árvores associadas ao gene superior, chama-se cruzamento de baixo nível. O cruzamento de alto nível tende a ter mais influência na saída final do indivíduo, já a de baixo nível permite manipular a estrutura das operações presentes no indivíduo (KOSHIYAMA; VELLASCO; TANSCHEIT, 2015).

Desta forma, pode-se destacar que, além do formato e da codificação dos indivíduos, as principais diferenças entre a PG-MG e a PG tradicional são a definição do máximo de árvores que indivíduo pode possuir e a realização de cruzamento em alto e baixo nível.

2.2.3 Sistemas *Fuzzy* Genéticos

A combinação entre Computação Evolucionária e Sistemas *Fuzzy* tem sido bastante explorada desde os anos 90 no intuito de integrar o gerenciamento de imprecisão e incerteza e a interpretabilidade dos Sistemas *Fuzzy* com a capacidade de adaptação e aprendizado dos algoritmos evolucionários (FERNANDEZ et al., 2015). Os Sistemas *Fuzzy* Genéticos ou *Fuzzy* Evolucionários, podem ser definidos como Sistemas *Fuzzy* que tem seu processo de aprendizado baseado em técnicas de Computação Evolucionária (CE) como Algoritmos Genéticos, Programação Genética, Estratégias Evolutivas. Estas técnicas são utilizadas tanto para definição da base de regras quanto para ajuste de parâmetros em sua estrutura (HERRERA, 2008). O primeiro passo para projetar um Sistema *Fuzzy* Genético é definir quais partes do sistema estão sujeitas à modelagem ou otimização por CE e modelá-las em cromossomos (indivíduos). Pode-se dividir essa abordagem em dois processos: aprendizado, para extrair conhecimento, gerar regras ou definir outros componentes do sistema; e ajuste, para obter a melhor combinação de parâmetros do sistema para melhorar seu desempenho. A estrutura de aprendizado de um Sistema *Fuzzy* Genético baseado em dados é ilustrado na Figura 13.

Considerando a tarefa de aprendizado da base de regras, que é foco deste trabalho, existem quatro possibilidades para realizar o processo de aprendizado genético

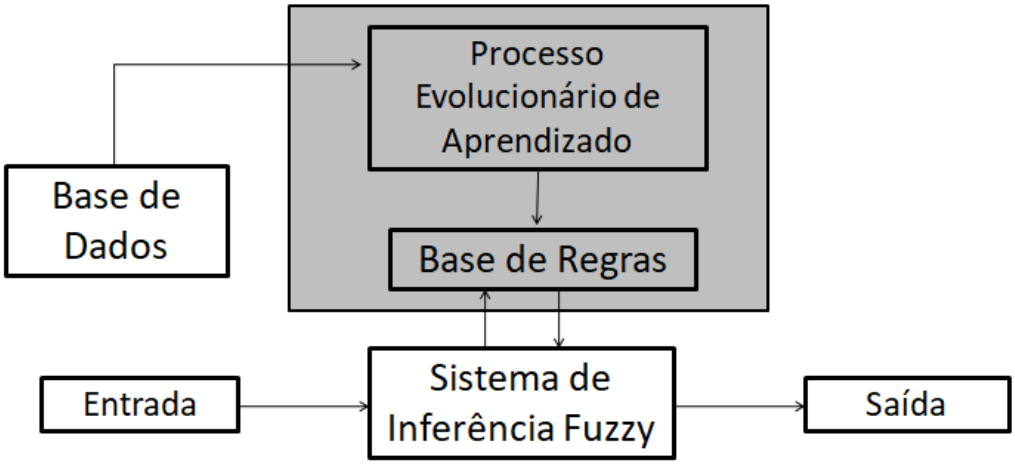

Figura 13 – Sistema *Fuzzy* Genético baseado em dados.

(FERNANDEZ et al., 2015):

- **Seleção evolucionária de regras**: o algoritmo evolucionário é utilizado em uma base de regras já existente para identificar regras redundantes, irrelevantes, errôneas ou conflitantes, de forma a melhorar o desempenho do sistema retornando uma base de regras otimizada;
- **Aprendizado evolucionário de componentes**: o algoritmo evolucionário é utilizado para gerar os melhores componentes para a base de conhecimento, como funções de pertinência ou conjuntos *fuzzy*. Neste caso o espaço de busca pode se tornar maior e o aprendizado mais lento.
- **Aprendizado evolucionário de regras**: o algoritmo evolucionário é utilizado para construir a base de regras utilizando informações, na maioria das vezes numéricas, disponíveis na base de dados.
- **Aprendizado evolucionário do banco de dados**: o algoritmo evolucionário é utilizado no processo de geração do banco de dados que permite aprender a forma das funções, a granularidade dos conjuntos *fuzzy* e outros componentes do banco de dados.

Para Cordón et al. (2001), a definição automática de um Sistema *Fuzzy* baseado em regras pode ser vista como um problema de busca ou de otimização, tendo despertado assim o uso de técnicas de Computação Evolucionária para esta tarefa devido a estas já serem bem conhecidas como uma técnicas globais de pesquisa capazes de trabalhar em grandes espaços de busca. Além disso, a estrutura de código genérico e os recursos independentes de desempenho dos algoritmos evolucionários os tornam boas opções para problemas em que se tem um conhecimento *a priori*, que no caso dos Sistemas *Fuzzy*, pode estar na forma de variáveis linguísticas, parâmetros das funções de pertinência, regras *fuzzy*, número de regras, etc. Ou seja, busca-se obter um conjunto

adequado de valores para os parâmetros de acordo com o critério de otimização. Esses parâmetros constituem o espaço de busca do algoritmo evolucionário.

As três principais abordagens para modelagem genética da base de regras são: *Michigan*, *Pittsburgh* e IRL(*Iterative Rule Learning*) (CORDÓN et al., 2004). Na abordagem de *Michigan* (HOLLAND; REITMAN, 1977) cada indivíduo representa uma regra, o que torna o custo computacional mais baixo, sendo a base de regras final composta por todo o conjunto de indivíduos. Isso torna a modelagem do indivíduo mais simples, porém sua avaliação se torna mais complexa devido à dificuldade em definir a qualidade de uma regra isolada. Já na abordagem de *Pittsburg* (SMITH, 1980) cada indivíduo codifica um conjunto de regras, facilitando assim sua avaliação, visto que toda a base de regras estará representada em um único indivíduo. Por outro lado, o custo computacional e o tempo de treinamento tornam-se maiores. Assim como a abordagem de *Michigan*, a IRL (VENTURINI, 1993) também considera que cada indivíduo representa uma regra porém, diferente da primeira, nesta metodologia o algoritmo evolucionário é executado por uma quantidade determinada de vezes e cada execução retorna uma regra para a base de regras final. Para definir a melhor regra podem ser utilizados os critérios de suporte, cobertura, grau de confiança, entre outros. Além disso também deve-se definir um critério de parada que pode considerar um número específico de regras, ou a abrangência do conjunto de regras obtido com relação ao padrão esperado, classes, consequentes, entre outros (CORDÓN et al., 2001).

Diversos trabalhos já exploraram a utilização de técnicas de Computação Evolucionária no intuito de prover aprendizado em Sistemas *Fuzzy* e Redes *Neuro-Fuzzy*. A próxima Seção apresenta uma revisão da literatura abordando trabalhos relevantes que utilizaram Computação Evolucionárias para construção de regras e ajuste de parâmetros em Sistemas *Fuzzy* e Redes *Neuro-Fuzzy*.

2.2.4 Revisão da literatura sobre Sistemas *Fuzzy* Genéticos

Em sua dissertação de mestrado Maia (2005) propôs um método de Programação Genética (PG) para aprendizado de regras em Sistemas *Fuzzy* do tipo Takagi-Sugeno. Nessa abordagem as funções para cálculo de consequente das regras são geradas iterativamente, regra a regra, por um processo de regressão simbólica realizado por meio da Programação Genética. Inicialmente são definidos *a priori* a quantidade máxima de regras e o erro mínimo, e estes dois parâmetros são utilizados como critério de parada do algoritmo. Os conjuntos *fuzzy* do antecedente são definidos por médias uniformemente espaçadas no ambiente de discurso, de forma que a cada regra gerada é necessário redefinir os conjuntos. Os parâmetros do consequente são definidos pela aplicação de uma *T-norma* sobre o conjunto *fuzzy* de cada regra. Foram utilizadas

funções de pertinência Gaussianas, com médias uniformemente espaçadas no universo de discurso de cada variável. O algoritmo salva todos os modelos encontrados em cada etapa do processo evolutivo da PG e retorna todos eles. Desta forma, permite que o usuário faça a escolha final do modelo que deseja utilizar.

Gallova (2009) propõe um algoritmo genético para aprendizado de regras em um Sistema *Fuzzy* aplicado ao diagnóstico de falhas em ambiente de maquinário baseado na teoria do caos. Neste trabalho foi utilizado um modelo de Mamdani para inicializar a base de regras e depois obter um modelo Takagi-Sugeno (TS), que foi utilizado para o diagnóstico. As regras TS e as funções de pertinência foram geradas e ajustadas por agrupamento auto-organizado. A abordagem possui um mecanismo para seleção de regras mais relevantes, eliminação de regras irrelevantes e combinação de regras. Os pesos são ajustados por *Backpropagation* na camada neural e os conjuntos *fuzzy* obtidos por um algoritmo de agrupamento *Fuzzy C-Means*. O Algoritmo Genético foi utilizado para avaliar e classificar as regras de diagnóstico, associando cada regra ao efeito que causa nos sintomas de falha.

Carmona et al. (2010) apresenta uma hibridização entre Lógica *Fuzzy* e Algoritmos Genéticos em que é utilizado um Algoritmo Evolucionário Multiobjetivo Não-Dominado como processo de aprendizado para extração de regras e descoberta de subgrupos em problemas de mineração de dados. Sua implementação é baseada na versão multiobjetivo dos Algoritmos Genéticos, NSGA-II (*Non-dominated Sorting Genetic Algorithm II*), que retorna um pareto ótimo, ou seja, um conjunto de soluções ótimas não-dominadas. Desta forma, o retorno do algoritmo é um conjunto de regras ótimas que atingem um limiar de confiança pré-determinado. As regras são extraídas de dados, mas usam uma estrutura próxima à estrutura de regras especialistas, associando os antecedentes aos rótulos dos consequentes que podem ser definidos pelo usuário ou por particionamento uniforme do universo de discurso, caso o conhecimento especialista não esteja disponível. Já em Carmona et al. (2011) é descrita a aplicação de um Sistema *Fuzzy* Evolucionário para encontrar subgrupos de pacientes e identificar os que tendem a visitar o departamento de emergência psiquiátrica de um hospital em determinado período. Para isso é utilizado um algoritmo evolucionário multiobjetivo (MESDIF - *Multi-Objective Evolutionary Subgroup Discovery Fuzzy Rules*) para extrair informações sobre a taxa de internação no setor psiquiátrico e gerar regras *fuzzy* que identifiquem estes subgrupos. Cada indivíduo do MESFID é uma regra, porém só há informação do antecedente, pois, em uma mesma execução, todas estão associadas ao mesmo consequente. Desta forma, cada execução gera um conjunto de regras para um mesmo consequente.

Três tipos de Sistemas *Fuzzy* Genéticos para detecção de intrusão em redes de com-

putadores são apresentados em Abadeh, Mohamadi e Habibi (2011). O Algoritmo Genético foi utilizado para criar a base de regras. O grau de certeza de cada regra e o consequente são determinados por um método heurístico. O modelo utilizado foi o de Takagi-Sugeno com funções de pertinência Triangulares e conjuntos *fuzzy* homogeneamente particionados. Considerando o mesmo problema, Victoire e Sakthivel (2011) apresentam um Sistema *Fuzzy* Evolucionário que utiliza um Algoritmo de Evolução Diferencial para gerar regras *fuzzy* de identificação de comportamento intrusivo. Além disso, nesse trabalho é utilizado uma heurística de busca local para refinar as regras encontradas. Estes dois trabalhos utilizam uma abordagem paralela onde a população é dividida em subpopulações que possam ser executadas em processadores diferentes.

No trabalho de Ibánez et al. (2011) é proposto um Sistema *Fuzzy* Evolucionário para o reconhecimento cranio-facial. Utilizando um Algoritmo Genético são levantados diferentes fatores de incerteza e delimitados ambientes de imprecisão para lidar com os objetos analisados para gerar os conjuntos *fuzzy*. Isso é realizado pela combinação de 12 parâmetros de transformação de registro, que permitem estabelecer uma ordem de importância entre diferentes pontos de referência. Aznarte et al. (2012) desenvolveram um modelo *fuzzy* a partir de um algoritmo bio-inspirado para previsão de séries temporais financeiras. Esta abordagem utiliza regras Mamdani baseadas em semântica local para gerar regras Takagi-Sugeno mais precisas. A seleção das melhores regras Mamdani que serão ajustadas localmente para este processo é feita por uma estratégia evolucionária.

Na pesquisa de Williams e Olatunji (2013) é proposta uma abordagem híbrida para diagnóstico de febre tifoide. O sistema apresentado é composto por três componentes, um componente Neural, um Genético e um *Fuzzy*. O componente Neural é composto de atributos da história médica do paciente, exame físico e investigação laboratorial e gera as funções de associação do sistema de inferência *fuzzy* e tem seus pesos de conexão ideais otimizados pelo componente Genético. O componente *fuzzy* lida com dados médicos imprecisos e incompletos para inferência do diagnóstico, do qual os atributos foram definidos após consulta a especialistas e a literatura padrão da área. Carmona et al. (2013a) utiliza um Algoritmo Evolucionário Multiobjetivo para extração de regras *fuzzy* que descrevam subgrupos do vírus da Influenza A. As variáveis contínuas são representadas por rótulos linguísticos na camada *fuzzy*, que podem ser especificados pelo usuário ou definidos por meio de partições uniformes, caso o conhecimento especialista não esteja disponível. O Algoritmo Evolucionário utilizado é baseado no NSGA-II, já mencionado anteriormente. Cada indivíduo representa uma regra e são avaliados por um valor de confiança, de forma que apenas regras com alta confiança sejam consideradas no pareto. As regras devem descrever e prever valores de classe para os vírus de Influenza A. O conjunto de regras final é obtido pela união do conjunto

de regras *fuzzy* no pareto que excedam o limite de confiança. Utilizando a mesma abordagem multiobjetivo e o mesmo tipo de modelagem *fuzzy* do trabalho anterior, Carmona et al. (2013b) apresenta um Sistema *Fuzzy* Evolucionário Multiobjetivo para descoberta de exceções de subgrupos em mineração de dados. Neste trabalho todos os rótulos linguísticos são obtidos por um algoritmo de descoberta de subgrupos, que divide os conjuntos em *fuzzy*, caso o domínio das variáveis seja contínuo, ou *crisp*, caso o domínio seja discreto. O algoritmo evolucionário é utilizado para lidar e detectar exceções nos subgrupos, buscando por conjuntos de exemplos em que o antecedente das regras do subgrupo não coincida com o consequente previamente estabelecido. Cada indivíduo representa uma regra, onde a quantidade de genes é a quantidade de variáveis contidas no conjunto de dados.

Mylonas, Stavrakoudis e Theocharis (2013) propõem um Sistema *Fuzzy* baseado em Algoritmo Genético (AG) para manipular imagens de sensoriamento remoto. A classificação em *pixels* das imagens é realizada por um classificador SVM (*Support Vector Machine*) com saída difusa. O SVM fornece um conjunto de mapas de associação *fuzzy* contendo valores de associação de pixels de cada imagem para cada classe. Após isso é utilizado um componente conectado para obter um mapa de segmentação preliminar. O AG é utilizado para simplificar a segmentação inicial, particionando as imagens em regiões maiores e mais homogêneas para obter uma melhor classificação das imagens dentro dos conjuntos *fuzzy*. Em Fahim et al. (2013) é proposto um modelo de Sistema *Fuzzy* Evolucionário para reconhecimento de atividade em ambientes de saúde e medir as incertezas associadas a essas atividades. Os conjuntos *fuzzy* são obtidos a partir de um algoritmo de extração e reconhecimento de atividades em *smartphones*. O Algoritmo Genético é utilizado para obter um conjunto de regras ótimas a partir da combinação de antecedentes e consequentes, onde cada indivíduo representa uma regra. No trabalho de Kaur, Bhardwaj e Been (2014) é apresentado um sistema *Neuro-Fuzzy* com aprendizado por *Backpropagation* para diagnóstico de risco de hipertensão. Os fatores de risco foram usados como entrada e os rótulos linguísticos, que representam o risco (alto, médio, baixo), foram definidos por especialistas. O Algoritmo Genético neste caso foi utilizado para inicializar a rede *Neuro-Fuzzy* com os pesos e vieses otimizados.

Muitas vezes a quantidade de regras criadas em um Sistema *Fuzzy* pode gerar redundância entre elas, o que faz com que existam uma quantidade excessiva e desnecessária de regras. Neste caso pode-se aplicar um algoritmo de aprendizado para avaliar essas regras, reduzir redundâncias e obter uma base de regras menor e mais eficiente como é apresentado em Elhag et al. (2015). O autor utiliza um Sistema *Fuzzy* Genético dentro de uma estrutura de aprendizagem pareada para detecção de vulnerabilidades em sistemas de informação e redes no intuito de evitar ataques cibernéticos. O AG é

utilizado para seleção das melhores regras no modelo de Mamdani obtendo assim um conjunto compacto delas dentro do contexto de cada problema. Utilizando Programação Genética (PG), Koshiyama, Vellasco e Tanscheit (2015) apresentam um Sistema *Fuzzy* para problemas de classificação. O trabalho utiliza a Programação Genética Multi-Gene para criação e ajuste da base de regras de um Sistema *Fuzzy* Genético de Pittsburg, onde cada indivíduo representa uma base de regras completa. As funções de pertinência utilizadas são Triangulares e complementares e a PG obtém as regras associadas a cada antecedente. O modelo utilizado é o de Takagi-Sugeno.

Já para diagnóstico de Tuberculose, foi utilizado um método de inferência *Neuro-Fuzzy* Genético (GENFIS) em Omisore, Samuel e Atajeromavwo (2017). O método utiliza aprendizado por *Forward Propagation* com sete camadas de neurônio e raciocínio guiado pelo modelo de inferência de Mamdani, com regras definidas por especialistas. O AG foi empregado para seleção dos parâmetros ideais de entrada da rede, que são utilizados para diagnóstico. Os indivíduos modelados carregam a informação da variável de diagnóstico e do peso de conexão, obtendo ao final o conjunto de regras e seus respectivos pesos otimizados. Ashish et al. (2018) apresentou um protótipo de *software* que utiliza modelos *Neuro-Fuzzy* para identificar os principais fatores responsáveis pela depressão. O referido trabalho aborda dois modelos, um com aprendizado por *Backpropagation* e outro por Algoritmo Genético. Abordagem por Algoritmo Genético além de otimizar pesos, fugindo de ótimos locais, ainda possibilita a identificação de regras redundantes e inexpressivas, obtendo assim uma base de regras mais limpa e otimizada. No trabalho de Soliman et al. (2018) é apresentado um Sistema *Neuro-Fuzzy* híbrido para melhorar o desempenho de um sistema de conversão de energia eólica de velocidade variável utilizando o ANFIS-GA (Sistema de inferência ANFIS com aprendizado baseado em Algoritmos Genéticos). O AG é utilizado para adaptar os parâmetros ANFIS em um sistema Takagi-Sugeno. Os antecedentes e consequentes são modelados como variáveis do Algoritmo Genético. Por ser um método global de pesquisa, favorece que se obtenha ao fim um conjunto de parâmetros de entrada e saída mais otimizado, fugindo dos ótimos locais. Esta abordagem trabalhou com dados não-lineares, demonstrando uma boa eficiência do método para trabalhar com esse tipo de dado.

Capítulo 3

NFN-PG-MG - *Neo-Fuzzy-Neuron* com Programação Genética Multi-Gene

A abordagem proposta neste trabalho foi desenvolvida sob a estrutura da rede *Neo-Fuzzy-Neuron* (NFN), apresentada na Seção 2.1.3 com aprendizado por Programação Genética. A Programação Genética será utilizada como ferramenta de regressão para criar e ajustar as funções de pertinência e, consequentemente, gerar regras automaticamente a partir dos dados.

3.1 Introdução

Esta seção introduz a abordagem proposta para construção da base de regras de uma rede *Neo-Fuzzy-Neuron*. A metodologia de *Pittsburg* (SMITH, 1980), na qual um indivíduo representa um conjunto completo de funções, foi utilizada em conjunto com a Programação Genética Multi-Gene (PG-MG) (FERREIRA, 2001) para criação das funções de pertinência que definem as regras da rede NFN e um método baseado no Gradiente para ajuste dos parâmetros do consequente. O modelo apresentado é chamado de NFN-PG-MG (*Neo-Fuzzy-Neuron* com Programação Genética Multi-Gene).

No NFN-PG-MG cada gene superior representa um modelo individual do NFN. Cada gene está associado a uma árvore de aridade m, sendo m o número de regras, ou seja, cada sub-árvore ligada ao nó central representa uma função de pertinência. O nó raiz é composto por uma função de agregação que retorna o resultado das m regras associadas àquela variável, conforme discutido na Seção 2.1.3.1 e ilustrado na Figura 14. Desta forma, a profundidade das árvores não se alteram, a alteração ao incluir ou excluir funções de pertinência ocorre na aridade. É realizado cruzamento de baixo nível, sorteando-se um ponto de corte para cada árvore ligada a cada gene superior de

cada um dos pais, e os filhos são gerados recombinando as informações a partir deste ponto. Desta forma, cada modelo individual do pai 1 é cruzado com cada respectivo modelo do pai 2, ou seja, é realizado o cruzamento em n pontos, sendo n o número de variáveis de entrada da rede, permitindo assim a alteração de regras referentes a todas as variáveis em cada cruzamento realizado.

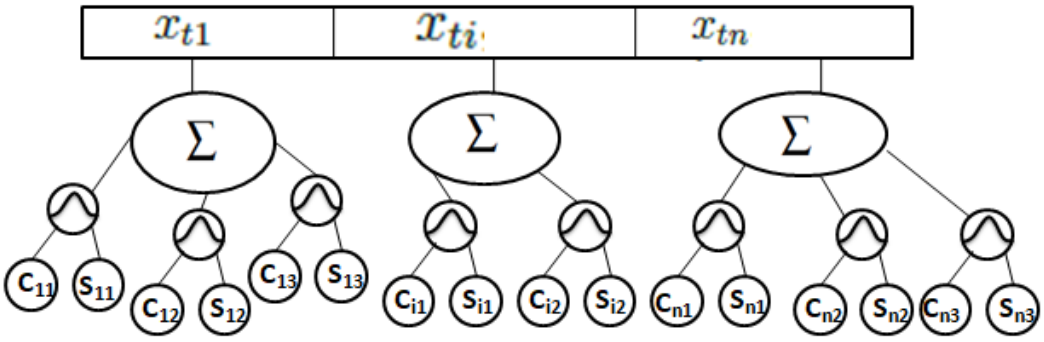

Figura 14 – Exemplo de indivíduo no NFN-PG-MG.

A reprodução é realizada gerando um filho exatamente igual ao pai (clone) e é aplicada quando a probabilidade de cruzamento não é atendida. Já na operação de mutação, é escolhida apenas uma função referente a um gene superior para ser alterada. Ou seja, é realizada mutação de 1 ponto, sendo alterada apenas uma regra referente a uma variável de entrada. O método de seleção de pais é o método do torneio, apresentado na Seção 2.2.1.2. Para geração da nova população foi utilizado elitismo para copiar o melhor indivíduo para a próxima população e os demais indivíduos são gerados pelas operações de cruzamento, reprodução e mutação.

Os indivíduos utilizam o mesmo conjunto de parâmetros dos consequentes e estes são ajustados por um método baseado no Gradiente, similar ao descrito na Seção 2.1.3.3. Os indivíduos são avaliados pelo Erro Quadrático Médio (EQM) obtido por:

$$EQM = \frac{\sum_{t=1}^{k}(e_t^2)}{k}, \tag{12}$$

em que e_t é o erro (diferença entre a saída esperada y_t e a saída do modelo $\hat{y}_t$) no instante t e k é o número de amostras.

Foram propostos três algoritmos, chamados respectivamente de NFN-PG-MG-I (Seção 3.2), NFN-PG-MG-II (Seção 3.3) e NFN-PG-MG-III (Seção 3.4). Nos três algoritmos tanto as funções quanto os parâmetros dos consequentes iniciais são gerados aleatoriamente. Para cada um dos três algoritmos foram propostas duas versões, na primeira versão o número de funções de pertinência é fixo e definido previamente pelo usuário.

Ou seja, todos os modelos criados pela PG contém o mesmo número de funções, podendo ser alterado ao longo do processo evolucionário. Já na segunda versão, o usuário apenas define um limite máximo para as funções de pertinência e a PG criará modelos com número de funções podendo variar de 1 até o limite definido (para cada indivíduo é sorteado um número de funções aleatório entre 1 e o limite máximo), ou seja, os modelos são de tamanhos variados desde o início do processo, sendo a PG responsável por encontrar o número de funções que melhor atende àquele modelo.

O operador de cruzamento é o responsável por modificar o número de funções de pertinência dos indivíduos pai para os indivíduos filho. Na primeira versão, basta permitir que o ponto de cruzamento selecionado no pai 1 e no pai 2 sejam diferentes para que os filhos gerados possam ter número de funções diferentes. Se sempre for selecionado o mesmo ponto de cruzamento para os dois pais, os filhos gerados terão o mesmo número de funções que os pais, visto que o algoritmo inicia com todos os indivíduos com o mesmo tamanho. Isso é facilmente ajustável no algoritmo e, neste caso, o NFN iniciará e terminará com o mesmo número de funções escolhido previamente pelo usuário e PG terá o papel apenas de encontrar as melhores funções para compor o modelo, mas não terá controle quanto ao número de funções, sendo este determinado exclusivamente pelo usuário. Já na segunda versão, o usuário não terá o controle do número de funções e os filhos sempre poderão ter quantidade de funções diferente dos pais, visto que os dois pais selecionados podem ter quantidade de funções diferentes e o ponto de cruzamento sorteado também pode não ser o mesmo para os dois. Nesta versão, mesmo que o ponto de cruzamento seja o mesmo, os filhos podem ter quantidade de funções diferentes caso a quantidade de funções dos pais sejam diferentes. O restante do algoritmo permanece igual para as duas versões.

Embora muitos passos do algoritmo sejam similares, optou-se nas próximas seções por explicar passo a passo cada algoritmo para facilitar o entendimento. A principal diferença se encontra na interação entre a PG para geração e ajuste das funções e o método baseado no gradiente para ajuste dos parâmetros dos consequentes. No caso do NFN-PG-MG-III altera-se também a geração do modelo inicial do NFN e, consequentemente, da população inicial da PG. Os algoritmos devem receber como parâmetros:

- g: número de gerações da PG.
- l: número de épocas do Gradiente.
- α: taxa de aprendizado para o método do Gradiente.
- tam_pop: tamanho da população.
- m_ini: número de funções de pertinência inicial, no caso da primeira versão.
- m_max: número máximo de funções de pertinência, no caso da segunda versão.

- tx_c: taxa de cruzamento - probabilidade de dois pais selecionados sofrerem cruzamento.
- tx_m: taxa de mutação - probabilidade de dois filhos gerados sofrerem mutação.
- Trn: tamanho do torneio (seleção da PG).

3.2 NFN-PG-MG-I

No NFN-PG-MG-I para cada geração da PG são executados l épocas de treinamento para ajuste dos parâmetros do consequente. O número de épocas de treinamento para cada geração é um parâmetro de entrada do algoritmo e experimentos iniciais mostram que esse parâmetro pode ser definido entre 1 e 10. Foram observados resultados melhores com $l = 5$, onde a Programação Genética apresenta convergência mais lenta, mas obtém uma menor taxa de erro. Utilizando $l = 10$ observou-se uma convergência prematura da Programação Genética em um número considerável de testes. O algoritmo pode ser resumido nas seguintes etapas:

1. A primeira etapa é a geração da população inicial. A população inicial da PG-MG é criada com indivíduos gerados aleatoriamente. As funções Gaussianas aleatórias são definidas sorteando-se, para cada função, um valor de c (centro da função) entre o limite inferior e o limite superior do domínio da variável de entrada, e um valor aleatório para o parâmetro s (espalhamento).

2. Os parâmetros q_{ij} são iniciados com valores randômicos entre 0 e 1.

3. O terceiro passo é avaliar a população gerada calculando o erro de modelagem de cada indivíduo (conjunto de regras). Os indivíduos são avaliados pelo Erro Quadrático Médio (EQM) obtido pela Equação 12.

4. Após a avaliação da população o melhor indivíduo (indivíduo com menor erro) é selecionado e terá seus parâmetros ajustados.

5. A próxima etapa consiste no ajuste de parâmetros do consequente do melhor indivíduo (conjunto de regras/funções de pertinência) definidos pela Etapa 4. Esse ajuste é realizado por um método baseado no Gradiente com l épocas de treinamento.

6. Verifica se o critério de parada foi atingido. Caso sim, encerra-se o algoritmo, caso não, continua a execução do algoritmo indo para o passo 7. O critério de parada pode ser um número de gerações estabelecido ou uma medida de erro como EQM (Equação 12) ou $RMSE$ (Equação 13), por exemplo.

7. Se o critério de parada não foi atingido na Etapa 6, o melhor indivíduo, selecionado na Etapa 4, passará pela próxima geração por elitismo e, os demais indivíduos da nova população são obtidos pelas operações de cruzamento, mutação e reprodução, explicados na Seção 3.1.

8. Nesta etapa a população atual é substituída pela nova população gerada na Etapa 7 e repete-se as Etapas 3 a 8 até que seja atingido o critério de parada na Etapa 6.

O fluxo de execução deste algoritmo pode ser visto na Figura 15 e é sumarizado pelo Algoritmo 2.

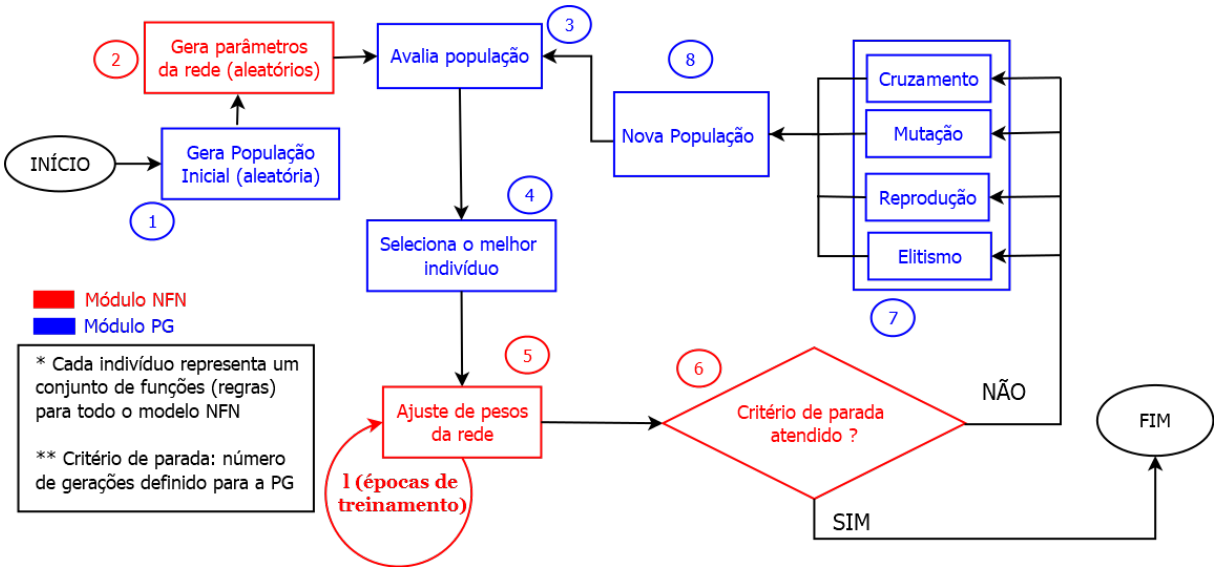

Figura 15 – Fluxograma do algoritmo NFN-PG-MG-I.

3.3 NFN-PG-MG-II

No segundo algoritmo proposto (NFN-PG-MG-II), os parâmetros do consequente são fixados até que se concluam todas as gerações da PG e então são executados l épocas de treinamento para ajuste dos parâmetros. O algoritmo segue as seguintes etapas:

1. A primeira etapa é a geração da população inicial. A população inicial da PG-MG é criada com indivíduos gerados aleatoriamente. As funções Gaussianas aleatórias são definidas sorteando-se, para cada função, um valor de c (centro da função) entre o limite inferior e o limite superior do domínio da variável de entrada, e um valor aleatório para o parâmetro s (espalhamento).

2. Os parâmetros q_{ij} são iniciados com valores randômicos entre 0 e 1.

Algoritmo 2: Algoritmo do NFN-PG-MG-I

```
Entrada x_t, y_t, α, g, l, tam_pop, m_ini, m_max, tx_c, tx_m, ;
Saida ŷ_t ;
Gerar População Inicial;
Inicializar q;
for geracoes = 1 : g do
    for individuo = 1 : tam_pop do
        Calcular ŷ_t ;
        Calcular EQM;
    end
    Selecionar melhor indivíduo;
    Atribuir funções do melhor indivíduo ao NFN;
    //Ajuste de Parâmetros
    for epoca = 1 : l do
        for t = 1,2,... do
            Calcular e_t ;
            for j = 1 : m do
                for i = 1 : n do
                    Calcular d_ij ;
                    Atualizar q_ij ;
                end
            end
            Calcular ŷ_t ;
            Calcular EQM;
        end
    end
    if EQM <= EQM_min then
        Finaliza Algoritmo;
    else
        //Gera Nova População
        Aplica Elitismo;
        for individuo = 2 : tam_pop do
            Pai1 ← SelecaoTorneio(PopulacaoAtual,Trn);
            Pai2 ← SelecaoTorneio(PopulacaoAtual,Trn);
            //Verifica probabilidade de cruzamento
            if rand < tx_c then
                Filho1 ← Cruzamento(Pai1,Pai2);
                Filho2 ← Cruzamento(Pai2,Pai1);
            else
                //Reprodução
                Filho1 ← Pai1;
                Filho2 ← Pai2;
            end
            //Verifica probabilidade de mutação
            if rand < tx_m then
                Filho1 ← Mutacao(Filho1);
            end
            if rand < tx_m then
                Filho2 ← Mutacao(Filho2);
            end
        end
        Substituir Populacao Anterior pela Nova;
    end
end
```

3. O terceiro passo é avaliar a população gerada calculando o erro de modelagem de cada indivíduo (conjunto de regras). Os indivíduos são avaliados pelo Erro Quadrático Médio (EQM) obtido pela Equação 12

4. Considerando o resultado da avaliação realizada na Etapa 3 o melhor indivíduo passará pela próxima geração por elitismo e, os demais indivíduos da nova população são obtidos pelas operações de cruzamento, mutação e reprodução, explicados na Seção 3.1.

5. Nesta etapa a população atual é substituída pela nova população gerada na Etapa 6.

6. Repete-se as Etapas 3, 6 e 5 até atingir o número de gerações estabelecido como critério de parada. Opcionalmente, pode-se utilizar como critério de parada uma medida de erro como EQM (Equação 12) ou $RMSE$ (Equação 13), por exemplo.

7. Após a execução de todas as gerações da PG-MG o melhor indivíduo (indivíduo com menor erro) é selecionado e terá seus parâmetros ajustados na etapa final.

8. Por fim, a última etapa consiste no ajuste de parâmetros do consequente do melhor indivíduo (conjunto de regras/funções de pertinência) definidos pela Programação Genética Multi-Gene. Esse ajuste é realizado por um método baseado no Gradiente com l épocas de treinamento.

Este algoritmo é ilustrado na Figura 16 e sumarizado no Algoritmo 3.

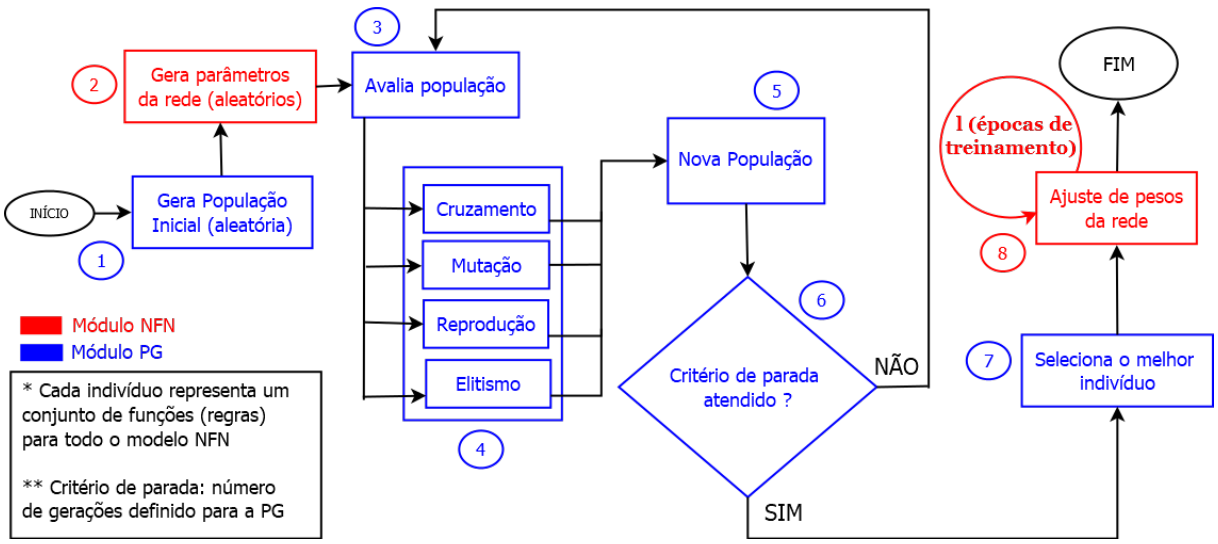

Figura 16 – Fluxograma do algoritmo NFN-PG-MG-II.

Algoritmo 3: Algoritmo do NFN-PG-MG-II

```
Entrada x_t, y_t, α, g, l, tam_pop, m_ini, m_max, tx_c, tx_m, ;
Saida ŷ_t ;
Gerar População Inicial;
Inicializar q;
for geracoes = 1 : g do
    for individuo = 1 : tam_pop do
        Calcular ŷ_t ;
        Calcular EQM;
    end
    //Gera Nova População
    Aplica Elitismo;
    for individuo = 2 : tam_pop do
        Pai1 ← SelecaoTorneio(PopulacaoAtual,Trn);
        Pai2 ← SelecaoTorneio(PopulacaoAtual,Trn);
        //Verifica probabilidade de cruzamento
        if rand < tx_c then
            Filho1 ← Cruzamento(Pai1,Pai2);
            Filho2 ← Cruzamento(Pai2,Pai1);
        else
            //Reprodução
            Filho1 ← Pai1;
            Filho2 ← Pai2;
        end
        //Verifica probabilidade de mutação
        if rand < tx_m then
            Filho1 ← Mutacao(Filho1);
        end
        if rand < tx_m then
            Filho2 ← Mutacao(Filho2);
        end
    end
    Substituir Populacao Anterior pela Nova;
end
Selecionar melhor indivíduo;
Atribuir funções do melhor indivíduo ao NFN;
//Ajuste de Parâmetros
for epoca = 1 : l do
    for t = 1,2,... do
        Calcular e_t ;
        for j = 1 : m do
            for i = 1 : n do
                Calcular d_ij ;
                Atualizar q_ij ;
            end
        end
        Calcular ŷ_t ;
        Calcular EQM;
    end
end
```

3.4 NFN-PG-MG-III

O NFN-PG-MG-III é composto de três fases: inicialização da rede; ajuste das funções de pertinência; ajuste dos parâmetros do consequente. Na primeira etapa as regras iniciais para a rede NFN são criadas com m funções uniformemente espaçadas, onde m é um parâmetro do algoritmo e são executadas p épocas de treinamento do Gradiente para ajuste inicial dos parâmetros, sendo que p dever ser definido como um valor baixo como será explicado posteriormente.

O ajuste das funções de pertinência por meio da Programação Genética é a segunda fase do algoritmo. Neste caso, ao invés da população inicial ser totalmente aleatória, ela é inicializada com as funções criadas na etapa anterior (uniformes) e o restante da população é criada de forma aleatória. Após a execução de todas as gerações da PG, a última etapa do algoritmo consiste no ajuste final dos parâmetros dos consequentes para a nova base de regras retornada. Esse ajuste é realizado por um método baseado no Gradiente, sendo executadas l épocas de treinamento. O algoritmo pode ser resumido nos seguintes passos:

1. A primeira etapa do algoritmo é o particionamento do universo de domínio de cada variável de entrada em m_i funções de pertinência Gaussianas uniformemente espaçadas. Em outras palavras, cria-se o primeiro indivíduo da população. A variável m_i é um parâmetro de entrada do algoritmo.

2. Os parâmetros q_{ij} são iniciados com valores randômicos entre 0 e 1.

3. Após a inicialização dos parâmetros da rede são executadas p épocas de treinamento para ajuste dos parâmetros utilizando o algoritmo baseado no Gradiente descrito na Seção 2.1.3.3. Experimentos computacionais realizados mostram que os melhores resultados foram obtidos com o número de épocas de treinamento com valores de p entre 5 e 10. Este ajuste inicial tem a função de, simplesmente, definir as regras que inicialmente têm mais peso e as regras que inicialmente têm menos peso, considerando funções uniformes, e por isso é utilizado um valor baixo para este parâmetro. Note que o valor de p pode ser definido como 0 caso opte por não realizar a etapa de ajuste inicial dos parâmetros e utilizar as funções apenas para inicializar a PG-MG para a próxima etapa. Destaca-se que o objetivo desta etapa é melhorar a inicialização dos parâmetros dos consequentes e não obter parâmetros ótimos.

4. A quarta etapa é a geração da população inicial. A população inicial da PG-MG é criada com o primeiro indivíduo descrito no Passo 1 e com os demais indivíduos gerados aleatoriamente. As funções Gaussianas aleatórias são definidas

sorteando-se, para cada função, um valor de c (centro da função) entre o limite inferior e o limite superior do domínio da variável de entrada, e um valor aleatório para o parâmetro s (espalhamento).

5. Na quinta etapa avalia-se a população gerada calculando o erro de modelagem de cada indivíduo (conjunto de regras). Os indivíduos são avaliados pelo Erro Quadrático Médio (EQM) obtido pela Equação (12).

6. Considerando o resultado da avaliação realizada na Etapa 5 o melhor indivíduo passará pela próxima geração por elitismo e, os demais indivíduos da nova população são obtidos pelas operações de cruzamento, mutação e reprodução, explicados na Seção 3.1.

7. Nesta etapa a população atual é substituída pela nova população gerada na Etapa 6.

8. Repete-se as Etapas 5, 6 e 7 até atingir o número de gerações estabelecido como critério de parada. Opcionalmente, pode-se utilizar como critério de parada uma medida de erro como EQM (Equação 12) ou $RMSE$ (Equação 13), por exemplo.

9. Após a execução de todas as gerações da PG-MG o melhor indivíduo (indivíduo com menor erro) é selecionado e terá seus parâmetros dos consequentes ajustados na etapa final.

10. Por fim, a última etapa consiste no ajuste de parâmetros do melhor indivíduo (conjunto de regras/funções de pertinência) definidos pela Programação Genética Multi-Gene. Esse ajuste é realizado por um método baseado no Gradiente com l épocas de treinamento.

Este algoritmo é ilustrado na Figura 17 e sumarizado pelo Algoritmo 4.

Algoritmo 4: Algoritmo do NFN-PG-MG-III

```
Entrada x_t, y_t, α, g, l, p, tam_pop, m_ini, m_max, tx_c, tx_m, ;
Saida ŷ_t ;
Gerar funções uniformemente espaçadas;
Inicializar parâmetros q;
//Ajuste de Parâmetros
for epoca = 1 : p do
    for t = 1,2,... do
        Calcular e_t ;
        for j = 1 : m do
            for i = 1 : n do
                Calcular d_ij ;
                Atualizar q_ij ;
            end
        end
        Calcular ŷ_t ;
        Calcular EQM;
    end
end
Gerar População Inicial;
Inicializar q;
for geracoes = 1 : g do
    for individuo = 1 : tam_pop do
        Calcular ŷ_t ;
        Calcular EQM;
    end
    //Gera Nova População
    Aplica Elitismo;
    for individuo = 2 : tam_pop do
        Pai1 ← SelecaoTorneio(PopulacaoAtual,Trn);
        Pai2 ← SelecaoTorneio(PopulacaoAtual,Trn);
        //Verifica probabilidade de cruzamento
        if rand < tx_c then
            Filho1 ← Cruzamento(Pai1,Pai2);
            Filho2 ← Cruzamento(Pai2,Pai1);
        else
            //Reprodução
            Filho1 ← Pai1;
            Filho2 ← Pai2;
        end
        //Verifica probabilidade de mutação
        if rand < tx_m then
            Filho1 ← Mutacao(Filho1);
        end
        if rand < tx_m then
            Filho2 ← Mutacao(Filho2);
        end
    end
    Substituir Populacao Anterior pela Nova;
end
Selecionar melhor indivíduo;
Atribuir funções do melhor indivíduo ao NFN;
//Ajuste de Parâmetros
for epoca = 1 : l do
    for t = 1,2,... do
        Calcular e_t ;
        for j = 1 : m do
            for i = 1 : n do
                Calcular d_ij ;
                Atualizar q_ij ;
            end
        end
        Calcular ŷ_t ;
        Calcular EQM;
    end
end
```

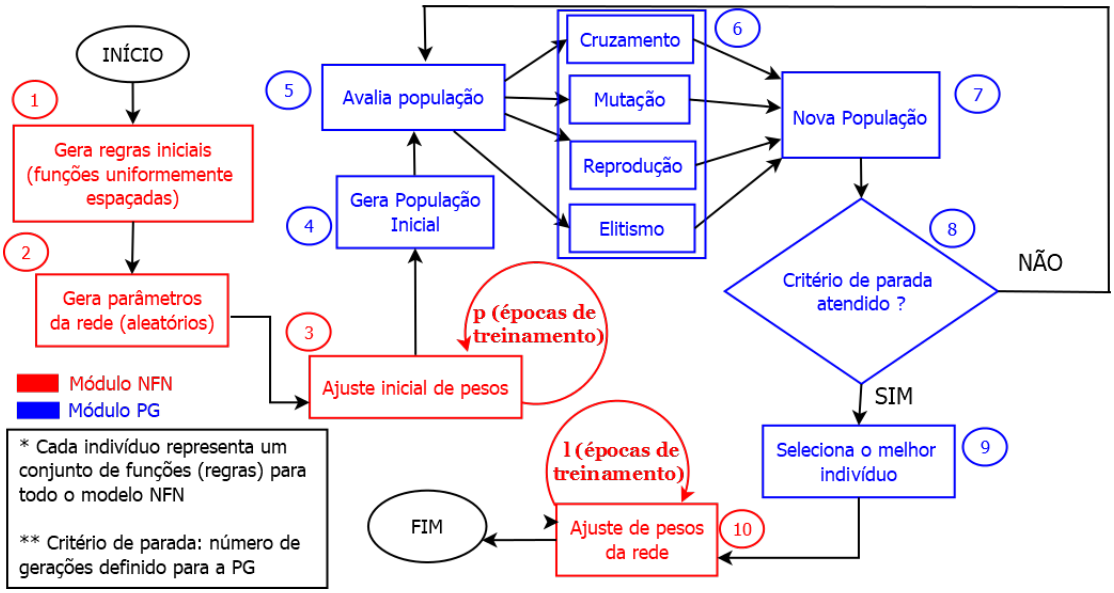

Figura 17 – Fluxograma do algoritmo NFN-PG-MG-III.

Capítulo 4

Experimentos Computacionais

Este capítulo apresenta os experimentos computacionais realizados para avaliar as abordagens propostas e compará-las com abordagens alternativas. A Seção 4.1 apresenta a metodologia dos experimentos e as medidas de desempenho utilizadas. Na Seção 4.2 avaliam-se os modelos na previsão de vazão semanal. A Seção 4.3 analisa o comportamento dos algoritmos na identificação de um sistema não-linear. Os resultados dos modelos para previsão de temperatura são apresentados na Seção 4.4. Já na Seção 4.5 são discutidos os resultados dos modelos para previsão de série temporal.

4.1 Metodologia

Os modelos propostos foram avaliados e comparados com modelos alternativos em problemas de previsão e identificação de sistemas não-lineares. Os modelos alternativos utilizados nos experimentos foram o ANFIS (*Adaptive Neuro-Fuzzy Inference System*) com aprendizado por Mínimos Quadrados e Gradiente Descendente com *Backpropagation*, uma Rede Neural MLP (*Multilayer Perceptron*) com aprendizado pelo método do Gradiente Descendente com *Backpropagation* com duas camadas ocultas, uma Rede Neural MLP (*Multilayer Perceptron*) com aprendizado pelo método do Gradiente Descendente com *Backpropagation* com três camadas ocultas e uma rede Neo-Fuzzy-Neuron com aprendizado pelo método do Gradiente.

Para cada experimento os conjuntos de dados foram divididos em três subconjuntos, um para treinamento com 60% das amostras, um para validação com 20% e outro para teste com 20% das amostras. Cada experimento foi executado 10 vezes (MONTGOMERY, 2017) e foram utilizados as seguintes métricas para comparar e avaliar o desempenho dos modelos: erro mínimo, erro médio e desvio padrão. A medida de erro utilizada foi o

RMSE (Root Mean Square Error) (13):

$$RMSE = \frac{1}{k}\left(\sum_{t=1}^{k}(y_t - \hat{y}_t)\right)^{\frac{1}{2}}, \tag{13}$$

em que k é o número de amostra do conjunto de teste, $\hat{y}_t$ é a saída estimada, y_t é a saída desejada. Os parâmetros dos modelos foram definidos como descrito a seguir para os experimentos realizados:

- ANFIS: épocas de treinamento = 500; funções de pertinência inicial = 2; tipo das funções pertinência = Gaussiana; α = 0.01.
- NFN-PG-MG-I: tamanho da população = 50; gerações = 300; épocas de treinamento = 5; taxa de cruzamento = 0.9; taxa de mutação = 0.08; α = 0.01; funções de pertinência iniciais para cada indivíduo = 5; tipo das funções de pertinência = Gaussiana.
- NFN-PG-MG-II: tamanho da população = 50; gerações = 300; épocas de treinamento = 500; taxa de cruzamento = 0.9; taxa de mutação = 0.08; α = 0.01; funções de pertinência iniciais para cada indivíduo = 5; tipo das funções de pertinência = Gaussiana.
- NFN-PG-MG-III: tamanho da população = 50; gerações = 300; p = 5; épocas de treinamento (l) = 200; taxa de cruzamento = 0.9; taxa de mutação = 0.08; α = 0.01; funções de pertinência iniciais = 5; máximo de funções de pertinência = 10; tipo das funções de pertinência = gaussiana.
- NFN: épocas de treinamento = 500; α = 0.01; funções de pertinência = 5; tipo das funções de pertinência = Gaussiana.
- RNA MLP- 2 camadas: camadas ocultas = 2; neurônios por camada = 10; épocas de treinamento = 500; α = 0.01; função de ativação tangente hiperbólica.
- RNA MLP- 3 camadas: camadas ocultas = 3; neurônios por camada = 10; épocas de treinamento = 500; α = 0.01; função de ativação tangente hiperbólica.

As medidas de erro podem não ser suficientes para comprovar se um modelo é estatisticamente superior a outro. Por isso, além das medidas de erro, o desempenho dos modelos será avaliado pelo teste estatístico ANOVA (RAMSAY, 2005). O teste ANOVA é um teste de análise de variância que avalia as médias dos grupos de dados e identifica se há diferença estatística entre eles. Isso é feito pela comparação da variabilidade entre grupos e da variabilidade dentro de cada grupo. A variância é dada pela Equação (14):

$$S^2 = \frac{\sum_i (z_i - \hat{z})^2}{t-1} \tag{14}$$

em que, z_i é a média amostral do grupo i, $\hat{z}$ é a média amostral global e t é o número de valores observados.

A análise estatística dos modelos foi realizada da seguinte forma: primeiro analisou-se os seis modelos propostos (duas versões para cada um dos três algoritmos) para verificar se há diferença estatística entre eles. Após isso o modelo que se mostrou estatisticamente superior pela análise do RMSE médio foi comparado com cada um dos três modelos alternativos. A primeira versão dos algoritmos demonstrou melhor desempenho em quase todos os casos, por este motivo optou-se por apresentar apenas o resultado desta versão nas próximas seções.

Os conjuntos de amostras referentes a todas as bases utilizadas foram testados para verificar os pressupostos de aleatoriedade, normalidade e homocedasticidade que validam a execução do teste ANOVA. Todos os modelos atenderam aos pressupostos exceto o modelo ANFIS que, por uma característica do algoritmo, retorna sempre o mesmo valor de RMSE para uma determinada configuração inicial. Neste caso, foi apresentado o p-valor nas tabelas referentes ao ANOVA para o modelo ANFIS porém, como não há variância nos dados, é necessário avaliar também o retorno deste modelo com relação à média dos demais. O grau de significância utilizado para o teste foi de $0,05$.

4.2 Previsão de Vazão Semanal

Nesta seção, avaliam-se os modelos na previsão da vazão média semanal de uma usina hidrelétrica de grande porte (Usina de Sobradinho). O objetivo do modelo é prever a vazão da semana seguinte baseado nas semanas anteriores, conforme Equação (15) (LEMOS; CAMINHAS; GOMIDE, 2011; SILVA et al., 2014).

$$y_t = f(y_{t-1}, y_{t-2}, y_{t-3}). \tag{15}$$

O conjunto de dados se refere ao período de 1931 a 2000 e é composto por 3.707 amostras com 3 variáveis de entrada e 1 de saída. Destas 2.224 foram utilizadas para treinar os modelos, 741 para validar e 742 para avaliar o desempenho dos modelos. Assim como em Lemos, Caminhas e Gomide (2011) e Silva et al. (2014) os dados disponibilizados estão normalizados entre 0 e 1 para manter a privacidade.

Como pode ser visto na Tabela 1 o NFN-PG-MG-I obteve o melhor desempenho, seguido pelo NFN-PG-MG-III. Os resultados obtidos pelo NFN tradicional e NFN-PG-MG-II são comparáveis e superam os dos demais modelos. A Figura 18 ilustra os valores desejados e os estimados pelo NFN-PG-MG-I para os dados de teste e a Figura

19 mostra as funções de pertinência geradas pela PG-MG. A convergência do algoritmo pode ser observada na Figura 20.

Tabela 1 – Desempenho na previsão de vazão semanal.

Modelo	RMSE (min)	RMSE (med)	Desv. Pad
NFN-PG-MG-I	0,0233278	0,0242848	0,0007752
NFN-PG-MG-III	0,0241328	0,0249433	0,0005129
NFN	0,0223711	0,0250892	0,0010261
NFN-PG-MG-II	0,0246414	0,0255287	0,0003498
ANFIS	0,0319734	0,0319734	0,0000000
MLP-3 camadas	0,0672769	0,0833734	0,0116284
MLP-2 camadas	0,0729739	0,0962813	0,0186128

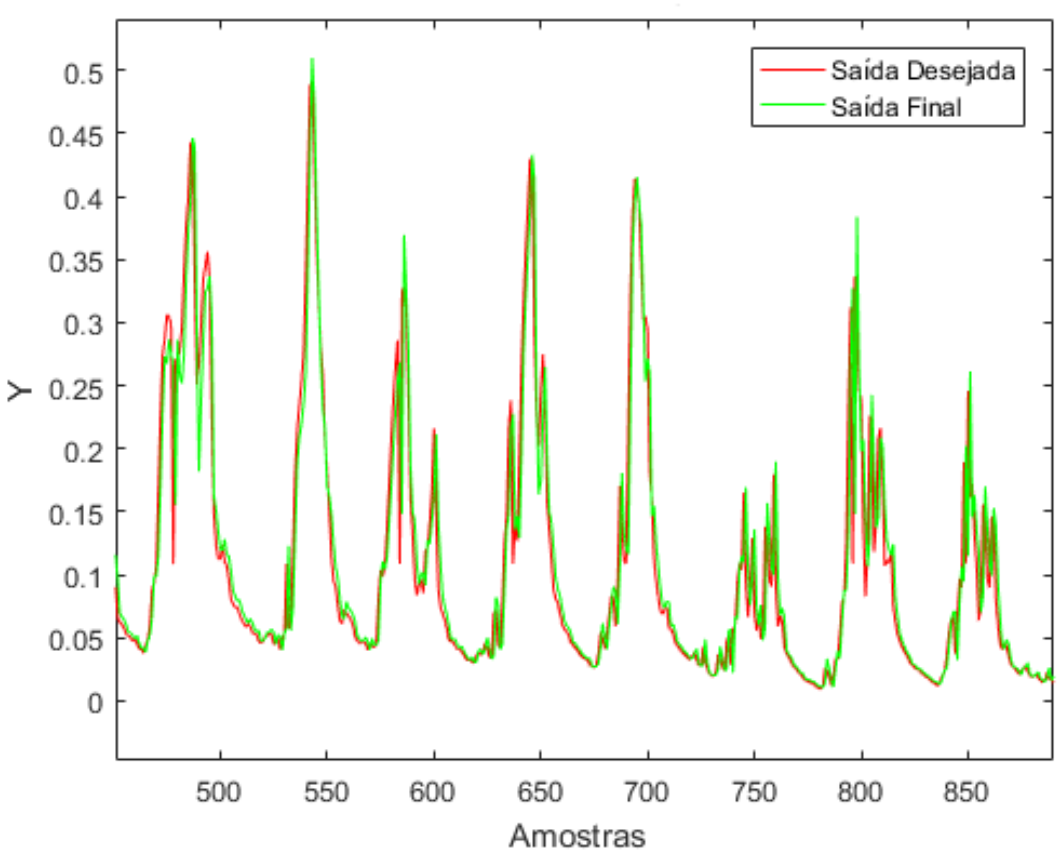

Figura 18 – Previsão de vazão semanal pelo NFN-PG-MG-I-v1.

Avaliando os modelos propostos pelo teste ANOVA, obteve-se um p-valor de $0,00516$, o que indica que há diferença estatística entre os eles, permitindo afirmar que o modelo com menor RMSE médio (NFN-PG-MG-I) é superior aos demais. Desta forma, o NFN-PG-MG-I foi comparado também utilizando-se o teste ANOVA com os modelos alternativos e os resultados podem ser vistos na Tabela 2. Como pode ser observado, os

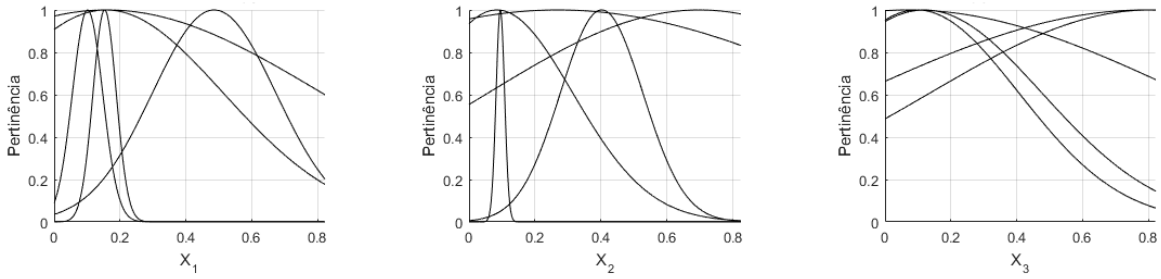

Figura 19 – Funções de pertinência finais para previsão de vazão semanal pelo NFN-PG-MG-I.

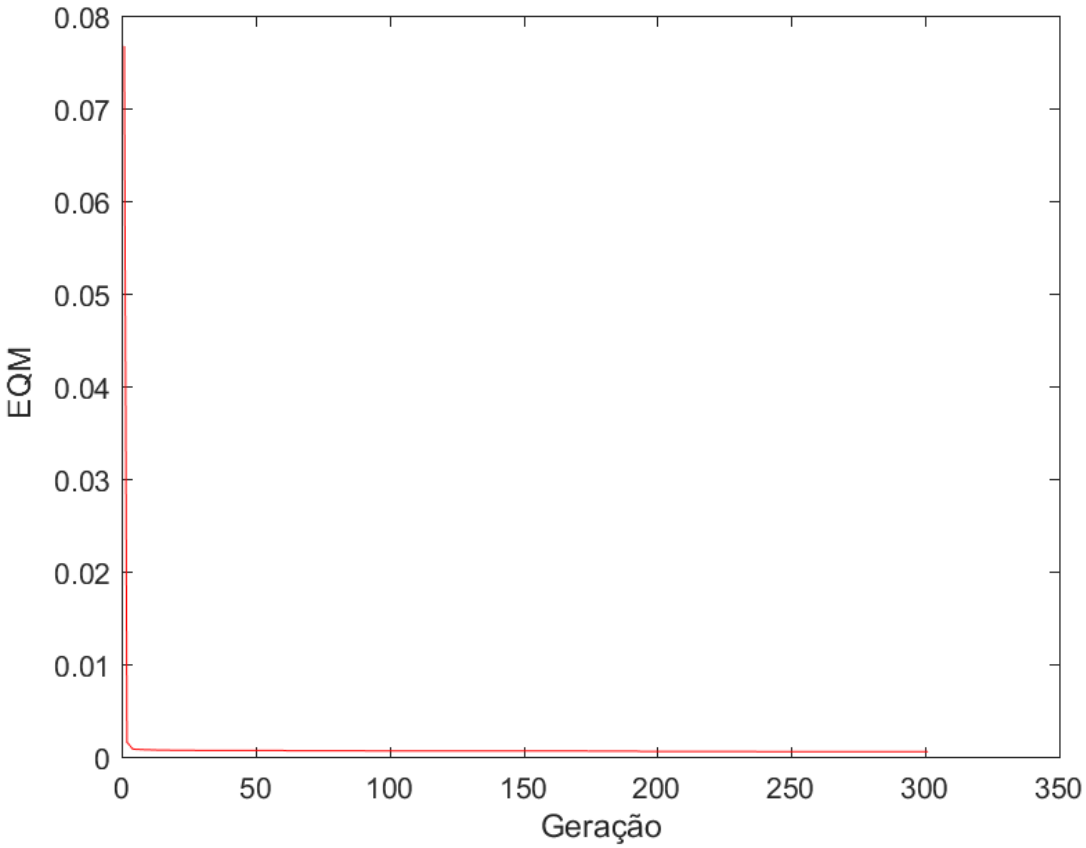

Figura 20 – Convergência do algoritmo NFN-PG-MG-I para previsão de vazão semanal.

resultados obtidos pelo NFN e pelo NFN-PG-MG-I não apresentam diferença estatística significativa, visto que o p-valor ficou acima do grau de significância definido. Já com relação aos demais modelos o NFN-PG-MG-I se demonstrou estatisticamente superior para o experimento apresentado.

Tabela 2 – Teste ANOVA do modelo NFN-PG-MG-I na previsão de vazão semanal.

Modelo	p-valor
NFN-PG-MG-I vs NFN	0,0634
NFN-PG-MG-I vs MLP-2 camadas	3,75 $\times 10^{-10}$
NFN-PG-MG-I vs MLP-3 camadas	4,20 $\times 10^{-12}$
NFN-PG-MG-I vs ANFIS	3,65 $\times 10^{-17}$

4.3 Identificação de Sistema Não-Linear

Este experimento visa analisar o comportamento dos algoritmos propostos na identificação de processo não linear descrito por (16):

$$y_t = \frac{y_{t-1}y_{t-2}y_{t-3}(y_{t-3}-y_{t-1})\beta_{t-1}+\beta_t}{1+y_{t-3}^2+y_{t-2}^2}, \tag{16}$$

em que $yt = 0$ e $\beta_t = 0$ para $t \leq 3$, β_t é definido pela Equação (17) para $3 < t \leq 500$ e pela Equação (18) para $t > 500$.

$$\beta_t = \sin\left(\frac{2\pi t}{250}\right), \tag{17}$$

$$\beta_t = \sin\left(\frac{2\pi t}{250}\right) + 0.4\sin\left(\frac{2\pi t}{25}\right). \tag{18}$$

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:

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}y_{t-5}), \tag{19}$$

em que $\hat{y}_t$ é a saída. Para o experimento foram criadas 10.000 amostras, sendo 4.000 utilizadas para treinar os modelos, 2.000 para validar e 2.000 para avaliar o seu desempenho pelo RMSE.

Os resultados obtidos pelos modelos na identificação do sistema não-linear são apresentados na Tabela 3. Observa-se que os melhores resultados foram obtidos pela rede ANFIS. Dentre os algoritmos baseados no NFN, o NFN-PG-MG-I obteve o melhor resultado seguido pelo NFN-PG-MG-II. O modelo NFN-PG-MG-III se mostrou inferior aos demais baseados no NFN, incluindo o NFN tradicional. A Figura 21 ilustra os valores desejados e os estimados para os dados de avaliação e a Figura 22 ilustra as

funções de pertinência geradas pelo NFN-PG-MG-I. A convergência do algoritmo pode ser observada na Figura 23.

Tabela 3 – Desempenho em identificação de sistema não-linear.

Modelo	RMSE (min)	RMSE (med)	Desv. Pad
ANFIS	0,0000084	0,0000084	0,0000000
NFN-PG-MG-I	0,0001160	0,0003289	0,0002664
NFN-PG-MG-II	0,0005638	0,0012207	0,0004897
NFN	0,0008628	0,0013669	0,0002830
NFN-PG-MG-III	0,0016606	0,0022040	0,0003148
MLP-3 camadas	0,0423198	0,0729564	0,0187451
MLP-2 camadas	0,0385384	0,0766687	0,0230878

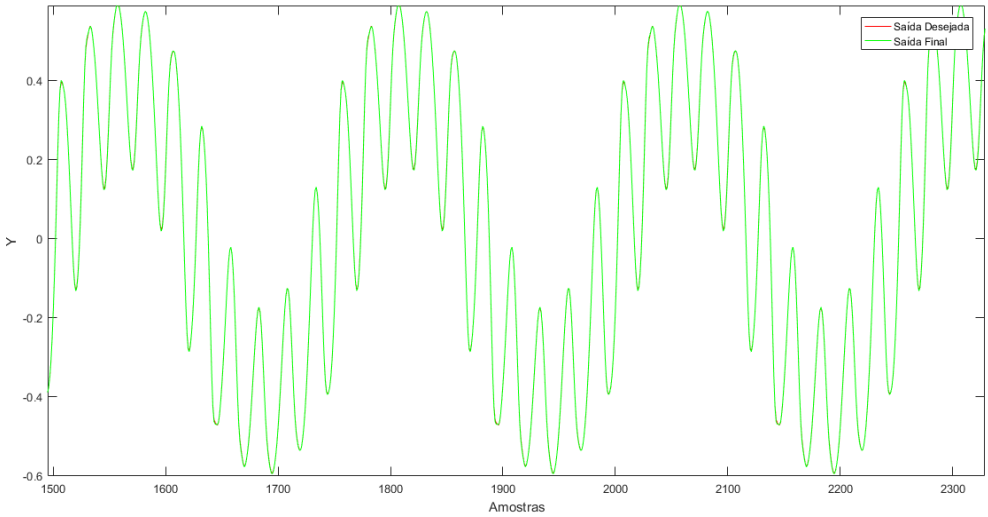

Figura 21 – Identificação de sistema não-linear pelo NFN-PG-MG-I.

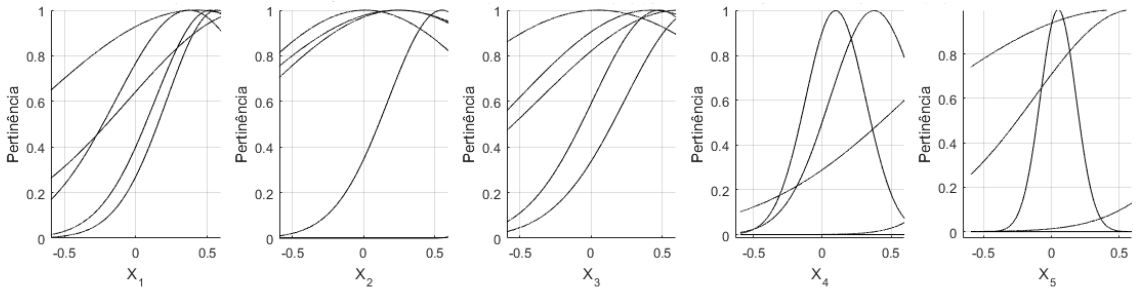

Figura 22 – Funções de pertinência finais para identificação de sistema não-linear pelo NFN-PG-MG-I.

Após aplicar o teste ANOVA aos modelos propostos, obteve-se um p-valor de $1,79 \times 10^{-11}$, concluindo que há diferença estatística entre os modelos. Sendo assim, o

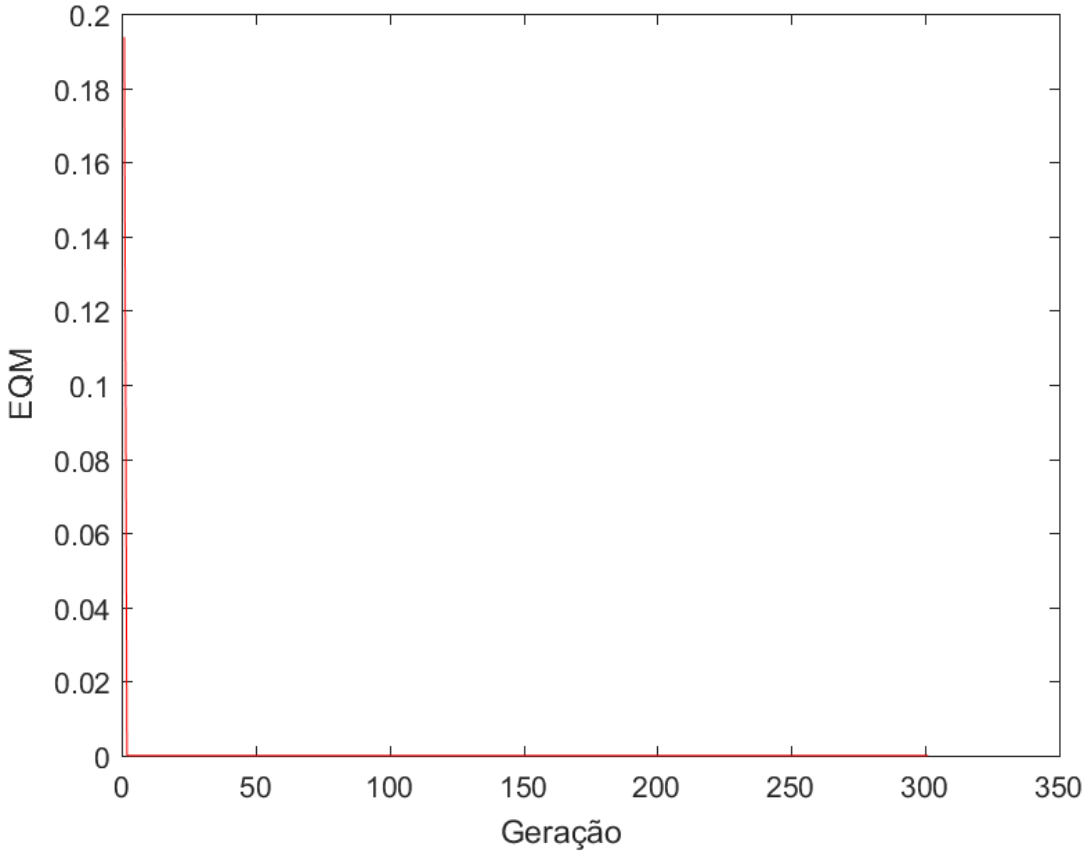

Figura 23 – Convergência do algoritmo NFN-PG-MG-I para identificação de sistema não-linear.

modelo com menor RMSE médio (NFN-PG-MG-I) foi comparado, também pelo teste ANOVA, com os modelos alternativos. Analisando os resultados apresentados na Tabela 4, conclui-se que há diferença estatística entre todos os modelos, sendo o ANFIS estatisticamente superior ao NFN-PG-MG-I que é estatisticamente superior aos demais modelos.

Tabela 4 – Teste ANOVA do modelo NFN-PG-MG-I na identificação de sistema não-linear.

Modelo	p-valor
NFN-PG-MG-I vs NFN	1,12 $\times 10^{-7}$
NFN-PG-MG-I vs MLP-2 camadas	4,48 $\times 10^{-9}$
NFN-PG-MG-I vs MLP-3 camadas	3,61 $\times 10^{-10}$
NFN-PG-MG-I vs ANFIS	0,0012

4.4 Previsão de Temperatura

Esta seção visa avaliar os modelos para previsão de temperatura em três regiões geográficas com padrões climáticos distintos. Considera-se dados referentes a temperaturas médias mensais em Ottawa: região de clima frio; Vale da Morte: região com clima extremamente seco e altas temperaturas; e Lisboa região com clima bastante variado ao longo do ano, oscilando de um inverno bastante frio com neve a um verão quente. O objetivo é prever a temperatura média mensal um passo à frente. Trabalhos anteriores sugerem utilizar os cinco primeiros valores defasados da série como entrada (LEITE et al., 2012; SILVA et al., 2012). Desta forma, o modelo pode ser descrito por:

$$y_t = f(y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-5}). \tag{20}$$

Serão apresentados 4 experimentos, 1 para cada região e 1 aplicando-se o modelo à uma base contento os dados das 3 regiões em conjunto.

4.4.1 Ottawa

Neste experimento consideram-se os dados de temperatura média mensal da região de Ottawa no período de janeiro de 1895 a dezembro de 2009. O conjunto de dados é composto por 1.374 amostras, sendo 824 para treinar os modelos, 275 para validar e 275 para avaliar seu desempenho. Os resultados obtidos pelos modelos para previsão de temperatura em Ottawa são exibidos na Tabela 5. Os resultados do NFN-PG-MG-I e do NFN tradicional se demonstraram competitivos, tendo o NFN tradicional apresentado um desempenho um pouco superior. Quando aos modelos propostos, o melhor desempenho pode ser observado pelo NFN-PG-MG-I, seguido pelo NFN-PG-MG-III e NFN-PG-MG-II respectivamente. Os valores desejados e os estimados pelo NFN-PG-MG-I para os dados de avaliação podem ser vistos na Figura 24, já as funções de pertinência são exibidas na Figura 25. A convergência do algoritmo pode ser vista na Figura 26.

O teste ANOVA para os modelos propostos obteve o p-valor de $5,65 \times 10^{-08}$, o que indica que há diferença estatística entre os eles, permitindo afirmar que o modelo com menor RMSE médio (NFN-PG-MG-I) é superior aos demais. Ao analisar o NFN-PG-MG-I com o modelo NFN, que obteve melhores resultados, concluiu-se que não há diferença estatística entre os dois, não podendo afirmar qual dos dois é de fato superior. Já com relação ao MLP com 2 e 3 camadas e ao ANFIS, conclui-se que o NFN-PG-MG-I é superior estatisticamente como mostrado na Tabela 6.

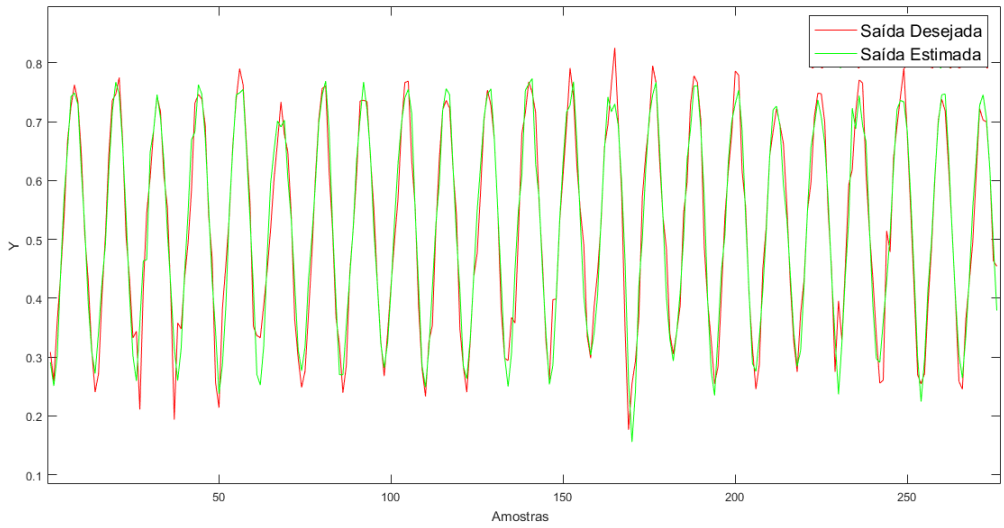

Figura 24 – Previsão de temperatura em Ottawa pelo NFN-PG-MG-I.

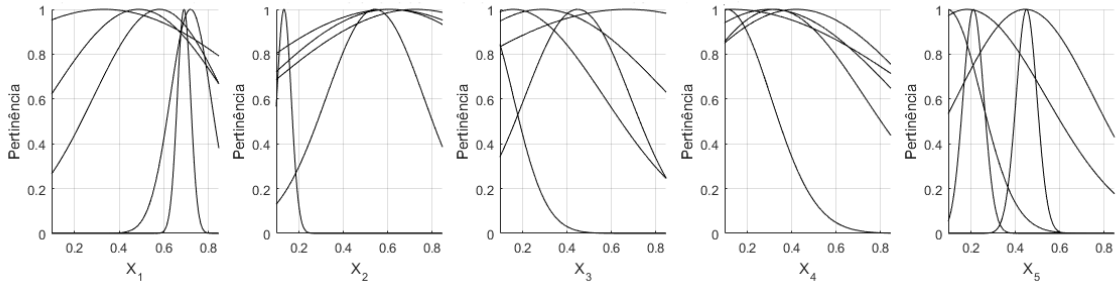

Figura 25 – Funções de pertinência finais para previsão de temperatura em Ottawa pelo NFN-PG-MG-I.

Tabela 5 – Desempenho na previsão de temperatura em Ottawa.

Modelo	RMSE (min)	RMSE (med)	Desv. Pad
NFN	0,0331154	0,0337854	0,0003997
NFN-PG-MG-I	0,0325634	0,0338468	0,0010981
NFN-PG-MG-III	0,0337191	0,0343145	0,0003519
NFN-PG-MG-II	0,0344032	0,0363913	0,0011542
ANFIS	0,0376826	0,0376826	0,0000000
MLP-3 camadas	0,0815700	0,1009696	0,0164085
MLP-2 camadas	0,0703481	0,1088822	0,0295322

4.4.2 Vale da Morte

Neste experimento consideram-se as informações de temperatura média mensal da região do Vale da Morte no período de janeiro de 1901 a dezembro de 2009. O conjunto

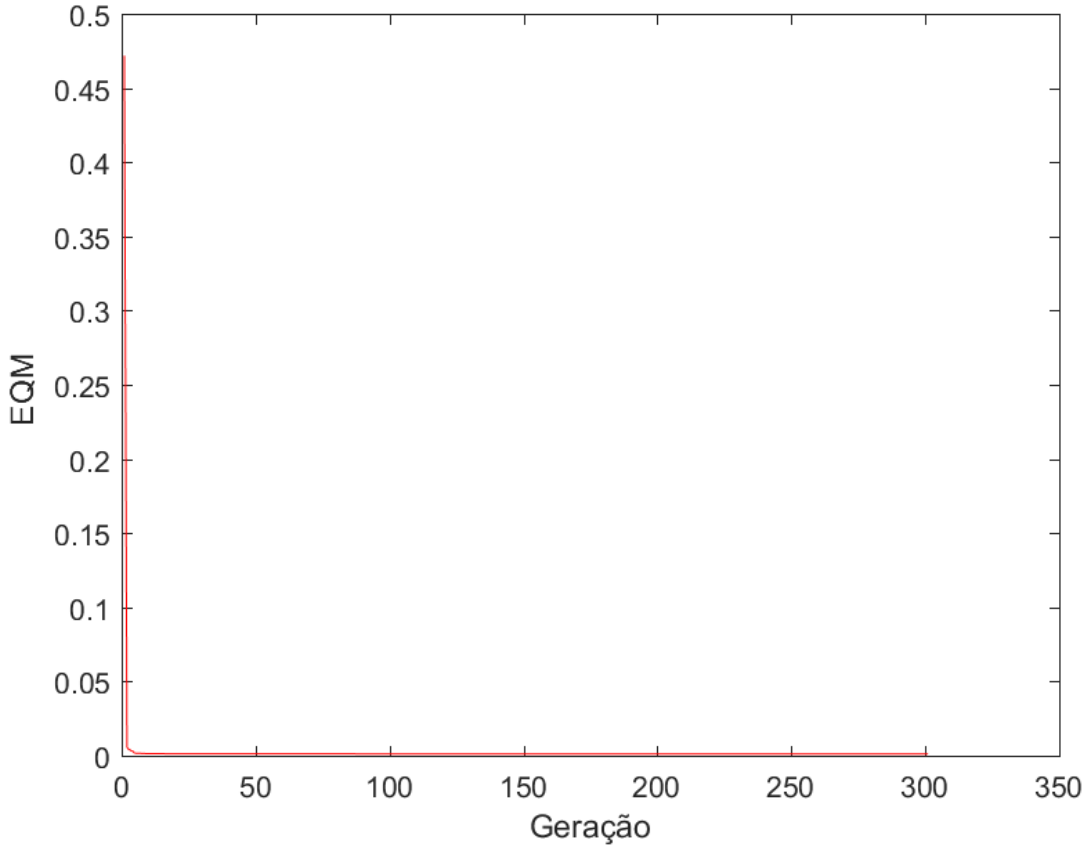

Figura 26 – Convergência do algoritmo NFN-PG-MG-I para previsão de temperatura em Ottawa.

Tabela 6 – Teste ANOVA do modelo NFN-PG-MG-I na previsão de temperatura em Ottawa.

Modelo	p-valor
NFN-PG-MG-I vs NFN	0,8698
NFN-PG-MG-I vs MLP-2 camadas	2,32 $\times 10^{-7}$
NFN-PG-MG-I vs MLP-3 camadas	1,54 $\times 10^{-10}$
NFN-PG-MG-I vs ANFIS	1,89 $\times 10^{-9}$

de dados contém 1.302 amostras, sendo 781 utilizadas para treinar os modelos, 260 para validar e 261 para avaliar seu desempenho. Os resultados obtidos pelos modelos para previsão de temperatura no Vale da Morte são exibidos na Tabela 7. Para esta base de dados, o melhor desempenho foi do NFN-PG-MG-I seguido pelo NFN-PG-MG-III. O NFN tradicional apresentou melhor desempenho que os demais, sendo o NFN-PG-MG-II o de pior desempenho. Os valores desejados e os estimados pelo NFN-PG-MG-I para os dados de avaliação podem ser vistos na Figura 27, já as funções de pertinência obtidas pelo algoritmo é ilustrada na Figura 28. A convergência do algoritmo pode ser observada na Figura 29.

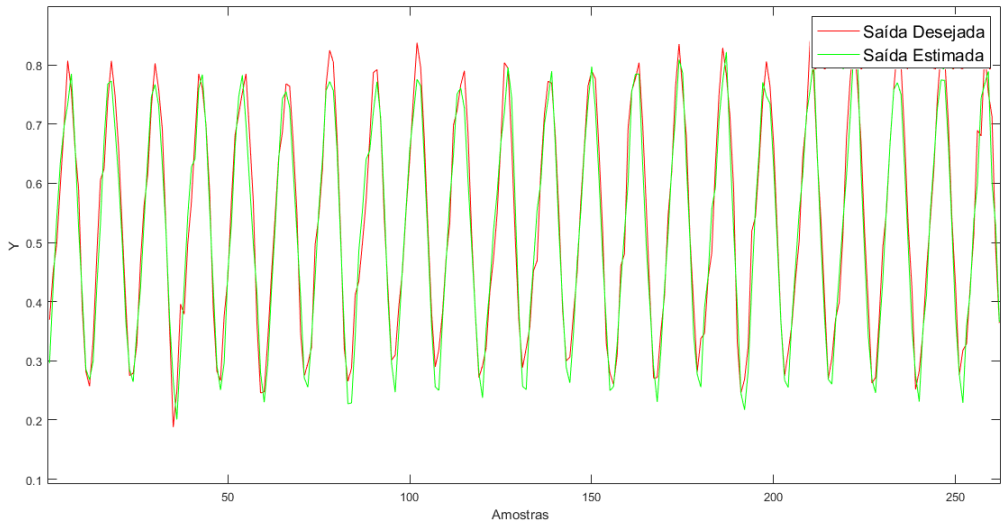

Figura 27 – Previsão de temperatura no Vale da Morte pelo NFN-PG-MG-I

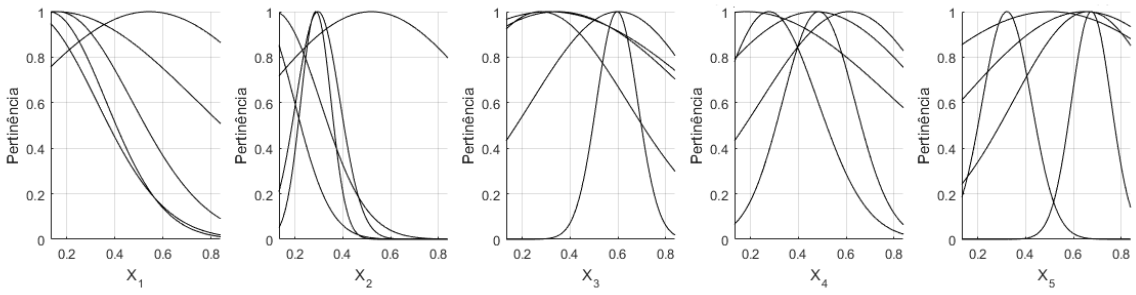

Figura 28 – Funções de pertinência finais para previsão de temperatura no Vale da Morte pelo NFN-PG-MG-I.

Tabela 7 – Desempenho na previsão de temperatura no Vale da Morte.

Modelo	RMSE (min)	RMSE (med)	Desv. Pad
NFN-PG-MG-I	0,0244763	0,0250551	0,0005692
NFN-PG-MG-III	0,0251621	0,0260473	0,0007683
NFN	0,0247127	0,0266811	0,0027430
NFN-PG-MG-II	0,0269283	0,0300724	0,0021483
ANFIS	0,0300854	0,0300854	0,0000000
MLP-2 camadas	0,0506189	0,0894537	0,0224695
MLP-3 camadas	0,0629002	0,0919026	0,0170895

O teste ANOVA para os modelos propostos, obteve o p-valor de $2,54 \times 10^{-10}$, o que indica que há diferença estatística entre os eles, permitindo afirmar que o NFN-PG-MG-I é superior aos demais, pois obteve menor RMSE médio. Os resultados do teste ANOVA para o NFN-PG-MG-I com os modelos alternativos podem ser vistos na Tabela 8.

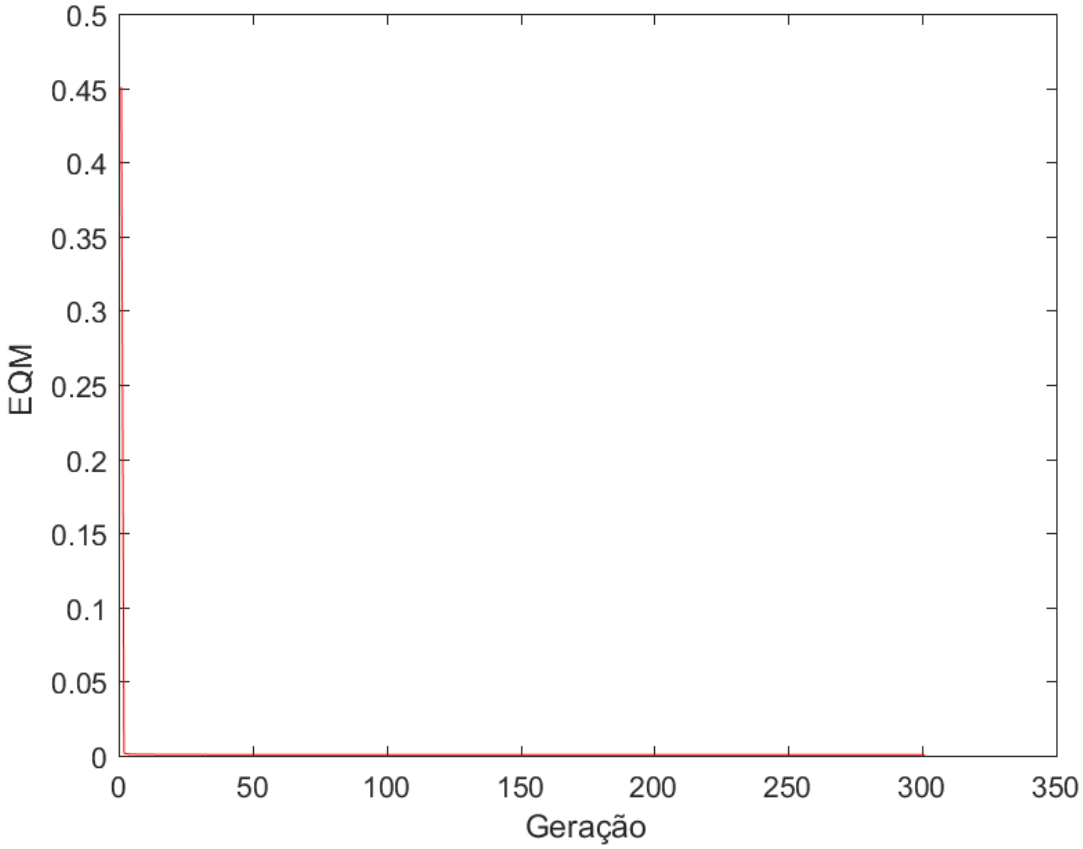

Figura 29 – Convergência do algoritmo NFN-PG-MG-I para previsão de temperatura em Ottawa.

Percebe-se que não há diferença estatística entre o NFN-PG-MG-I e o NFN tradicional, já os demais modelos se demonstraram estatisticamente inferiores ao modelo proposto.

Tabela 8 – Teste ANOVA do modelo NFN-PG-MG-I na previsão de temperatura no Vale da Morte.

Modelo	p-valor
NFN-PG-MG-I vs NFN	0,0830
NFN-PG-MG-I vs MLP-2 camadas	3,98 $\times 10^{-8}$
NFN-PG-MG-I vs MLP-3 camadas	3,11 $\times 10^{-10}$
NFN-PG-MG-I vs ANFIS	2,79 $\times 10^{-16}$

4.4.3 Lisboa

Para a região de Lisboa os dados se referem à temperatura média mensal no período de janeiro de 1910 a dezembro de 2009. O conjunto de dados é composto por 1.194 amostras, sendo 716 para treinar os modelos, 239 para validação e 239 para avaliar seu desempenho. Os resultados obtidos pelos modelos para previsão de temperatura

em Lisboa são exibidos na Tabela 9. O melhor desempenho observado foi do NFN-PG-MG-I seguido do NFN-PG-MG-III. O NFN-PG-MG-II se demonstrou inferior aos demais modelos baseados no NFN, porém ainda superior aos modelos MLP. Os valores desejados e os estimados pelo NFN-PG-MG-I para os dados de avaliação podem ser vistos na Figura 30, e a Figura 31 mostra as funções de pertinência geradas pelo algoritmo. Já a convergência do algoritmo é exibida na Figura 32.

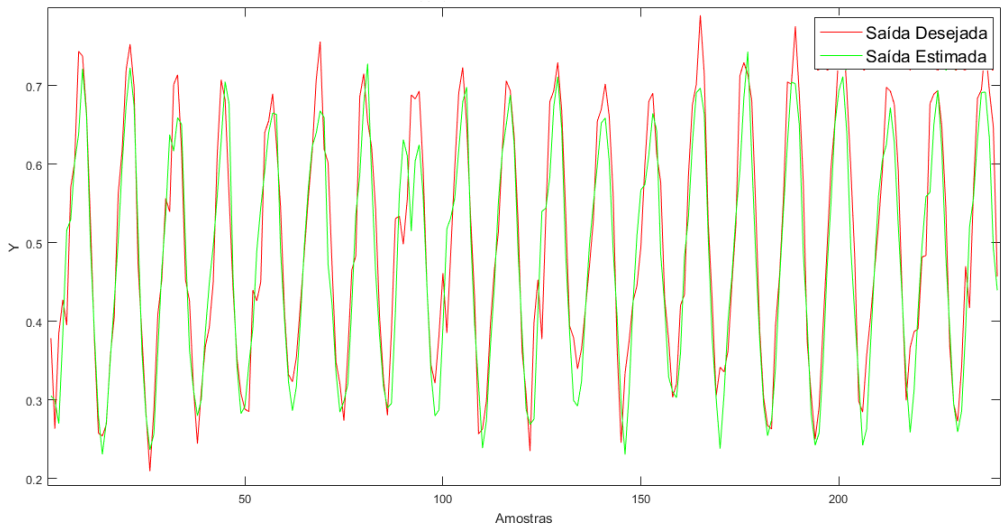

Figura 30 – Previsão de temperatura em Lisboa pelo NFN-PG-MG-I.

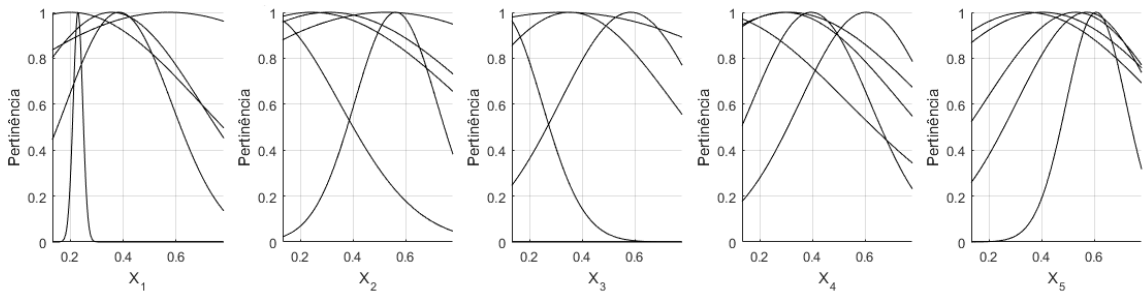

Figura 31 – Funções de pertinência finais para previsão de temperatura em Lisboa pelo NFN-PG-MG-I.

Os modelos propostos apresentaram diferença estatística pelo teste ANOVA que obteve o p-valor de $5,88 \times 10^{-8}$, concluindo que o NFN-PG-MG-I é superior aos demais, pois obteve menor RMSE médio. Os resultados do teste ANOVA para o NFN-PG-MG-I com os modelos alternativos podem ser vistos na Tabela 10. O NFN-PG-MG-I e o NFN tradicional apresentaram igualdade estatística, já com relação aos modelos MLP e ANFIS, o NFN-PG-I se demostrou superior.

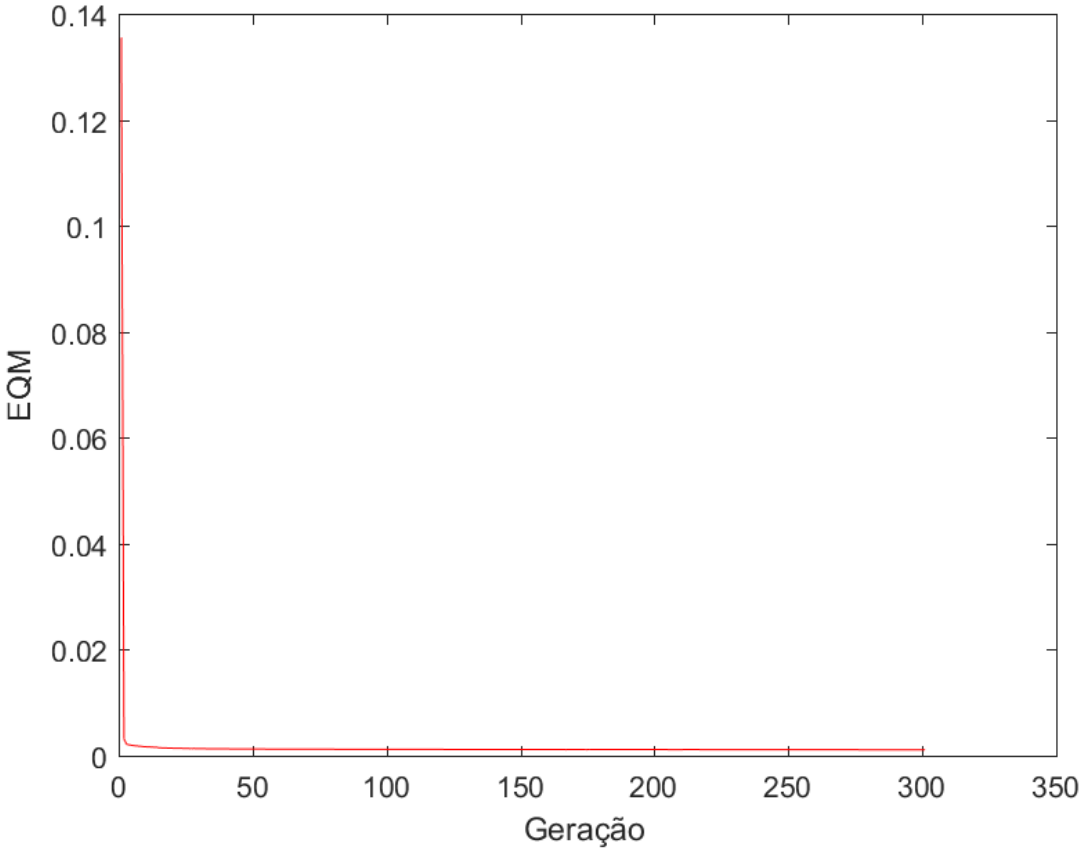

Figura 32 – Convergência do algoritmo NFN-PG-MG-I para previsão de temperatura em Lisboa.

Tabela 9 – Desempenho na previsão de temperatura em Lisboa.

Modelo	RMSE (min)	RMSE (med)	Desv. Pad
NFN-PG-MG-I	0,0330777	0,0350318	0,0016136
NFN-PG-MG-III	0,0345385	0,0351708	0,0004496
NFN	0,0322673	0,0361059	0,0029955
NFN-PG-MG-II	0,0355998	0,0390821	0,0012388
ANFIS	0,0426674	0,0426674	0,0000000
MLP-3 camadas	0,0817671	0,1048698	0,0179410
MLP-2 camadas	0,0699536	0,1088691	0,0280405

4.4.4 Ottawa, Vale da Morte e Lisboa

Este experimento foi realizado com um conjunto de dados formado pelos dados das temperaturas das três diferentes regiões (Ottawa, Vale da Morte e Lisboa). O conjunto é composto por 3.882 amostras, sendo que 2.329 foram utilizadas para treinar os modelos, 776 para validar e 777 para avaliar o desempenho dos modelos. Os resultados obtidos pelos modelos para previsão de temperatura considerando as três regiões são exibidos

Tabela 10 – Teste ANOVA do modelo NFN-PG-MG-I na previsão de temperatura em Lisboa.

Modelo	p-valor
NFN-PG-MG-I vs NFN	0,3313
NFN-PG-MG-I vs MLP-2 camadas	1,41 $\times 10^{-7}$
NFN-PG-MG-I vs MLP-3 camadas	3,57 $\times 10^{-10}$
NFN-PG-MG-I vs ANFIS	1,34 $\times 10^{-11}$

na Tabela 11. Para este experimento o modelo ANFIS demonstrou melhor resultado seguido pelo NFN-PG-MG-III. Os valores desejados e os estimados pelo NFN-PG-MG-III para os dados de avaliação podem ser vistos na Figura 33 e as funções de pertinência na Figura 34. A convergência do algoritmo pode ser observada na Figura 35, sendo o eixo horizontal do gráfico referente ao total de gerações (300) e épocas de treinamento (5 iniciais + 500 finais), totalizando 805 iterações.

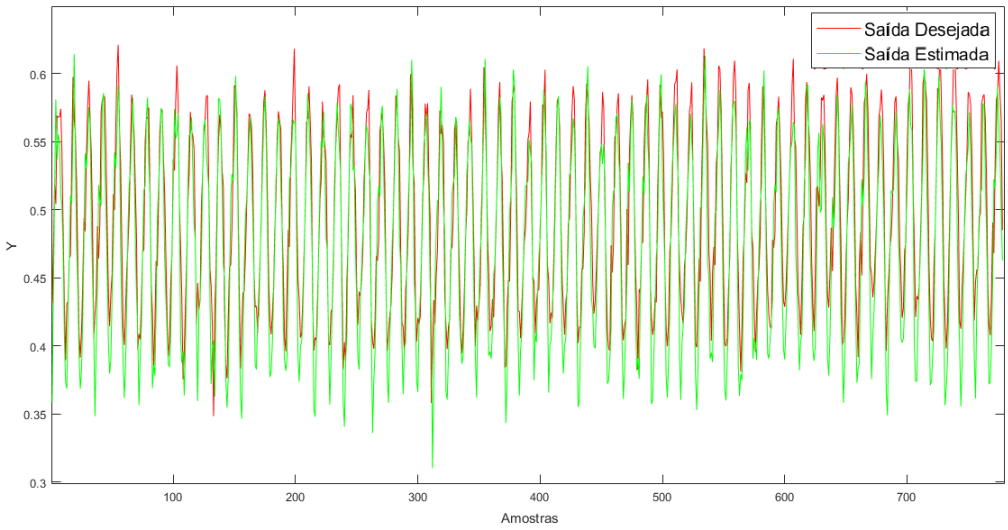

Figura 33 – Previsão de temperatura em Ottawa, Vale da Morte e Lisboa pelo NFN-PG-MG-III.

Ao realizar o teste ANOVA constatou-se a diferença estatística entre os modelos propostos com p-valor de $4,24 \times 10^{-11}$, concluindo que o NFN-PG-MG-III, que obteve menor RMSE médio, é superior aos demais. Ao comparar com os modelos alternativos, observou-se diferença estatística entre todos, sendo o ANFIS superior ao NFN-PG-MG-III que é estatisticamente superior ao NFN e ao MLP com 2 e 3 camadas como pode ser visto na Tabela 12.

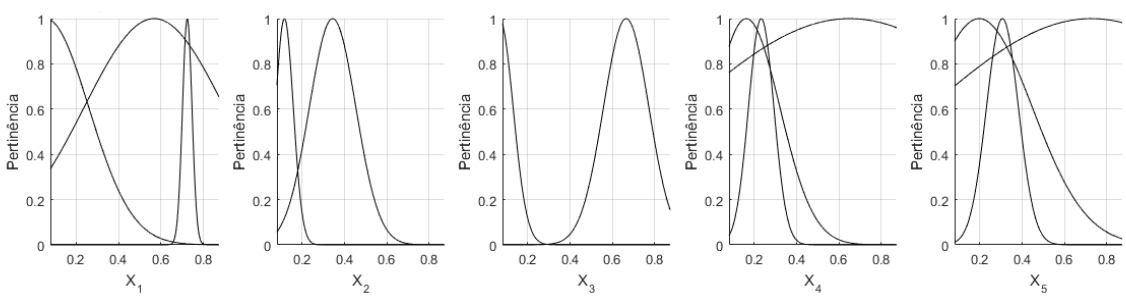

Figura 34 – Funções de pertinência finais para previsão de temperatura em Ottawa, Vale da Morte e Lisboa pelo NFN-PG-MG-III.

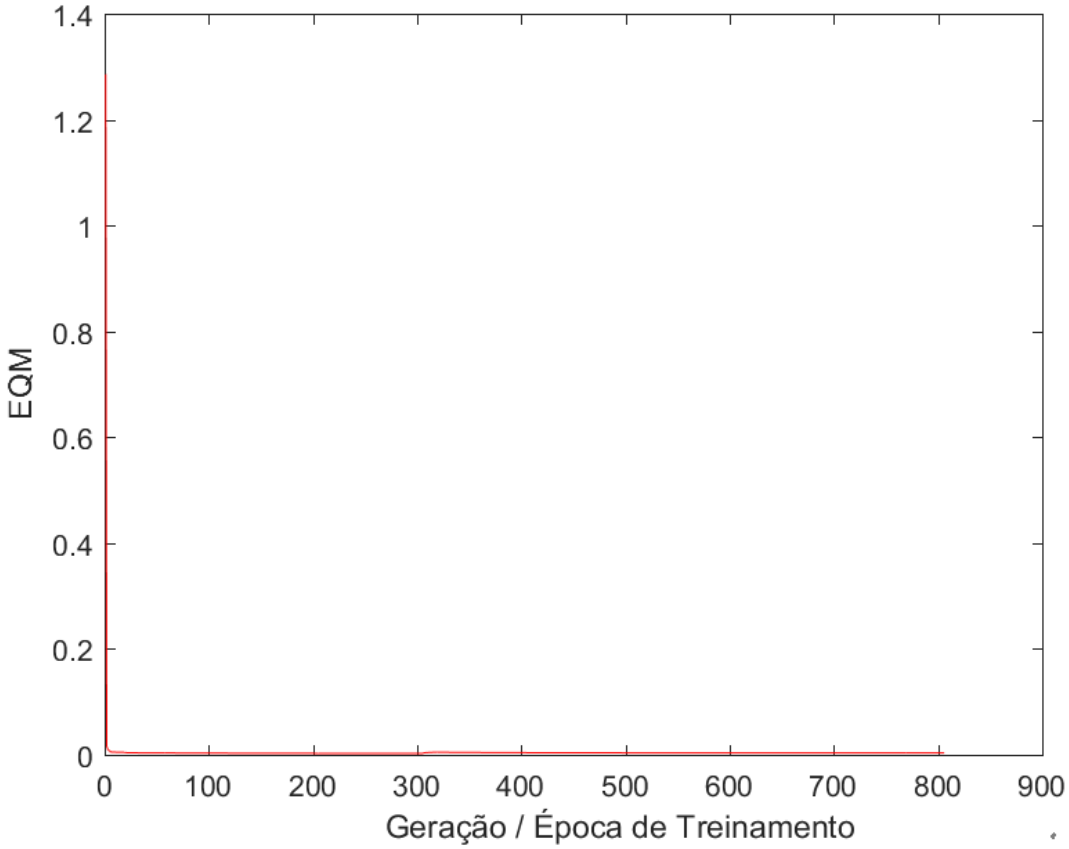

Figura 35 – Convergência do algoritmo NFN-PG-MG-III para previsão de temperatura em Ottawa, Vale da Morte e Lisboa.

4.5 Série Temporal de Mackey-Glass

Nesta seção os modelos são avaliados para previsão da série temporal clássica de Mackey-Glass (MACKEY; GLASS, 1977). A série é criada pela seguinte equação diferencial com atraso:

Tabela 11 – Desempenho na previsão de temperatura em Ottawa, Vale da Morte e Lisboa.

Modelo	RMSE (min)	RMSE (med)	Desv. Pad
ANFIS	0,0349498	0,0349498	0,0000000
NFN-PG-MG-III	0,0453267	0,0536730	0,0053396
NFN-PG-MG-I	0,0461923	0,0542186	0,0039806
NFN-PG-MG-II	0,0584540	0,0712369	0,0054274
NFN	0,0666964	0,0729835	0,0037895
MLP-3 camadas	0,0822092	0,1066284	0,0165257
MLP-2 camadas	0,0827563	0,1239246	0,0349244

Tabela 12 – Teste ANOVA do modelo NFN-PG-MG-III na previsão de temperatura em Ottawa, Vale da Morte e Lisboa.

Modelo	p-valor
NFN-PG-MG-III vs NFN	2,58 $\times 10^{-8}$
NFN-PG-MG-III vs MLP-2 camadas	6,28 $\times 10^{-6}$
NFN-PG-MG-III vs MLP-3 camadas	1,55 $\times 10^{-8}$
NFN-PG-MG-III vs ANFIS	1,78 $\times 10^{-9}$

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x(t-\tau)^{10}} - 0.1x(t), \tag{21}$$

em que $x(0) = 1.2$ e $\tau = 17$. O objetivo é prever o valor x_{t+6} para qualquer valor de t baseado nas entradas $[x_{t-18}x_{t-12}x_{t-6}x_t]$. O conjunto de dados é composto de 3500 amostras, com 4 variáveis de entrada e 1 de saída, das quais 3000 foram coletadas com $t\varepsilon[201,3.200]$ e 500 com $t\varepsilon[5.001,5.500]$ (KASABOV; SONG, 2002). Foram utilizadas 2.100 para treinar os modelos, 700 para validar e 700 para avaliar o desempenho dos modelos pelo RMSE.

A Tabela 13 mostra o desempenho dos modelos para previsão da série temporal de Mackey-Glass. Observa-se que o melhor desempenho foi obtido pelo modelo ANFIS seguido do NFN-PG-MG-III e do NFN-PG-MG-I. A Figura 36 ilustra os valores desejados e os estimados pelo NFN-PG-MG-III para os dados de avaliação. As funções de pertinência obtidas pelo algoritmo podem ser vistas na Figura 37. Já a convergência do algoritmo pode ser observada na Figura 38, sendo o eixo horizontal do gráfico referente ao total de gerações (300) e épocas de treinamento (5 iniciais + 500 finais), totalizando 805 iterações.

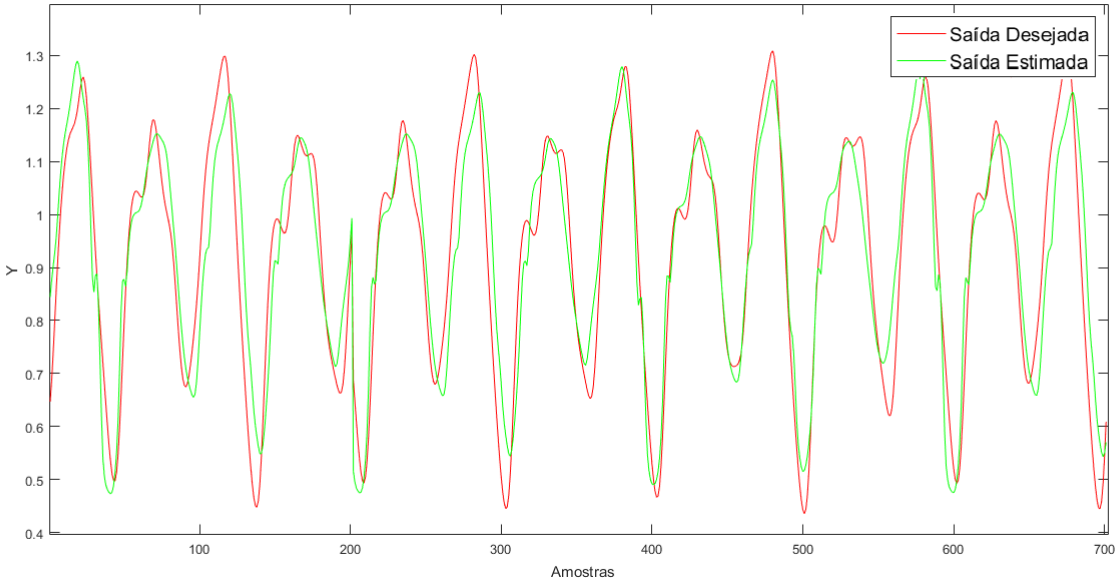

Figura 36 – Previsão da série temporal de Mackey-Glass pelo NFN-PG-MG-III.

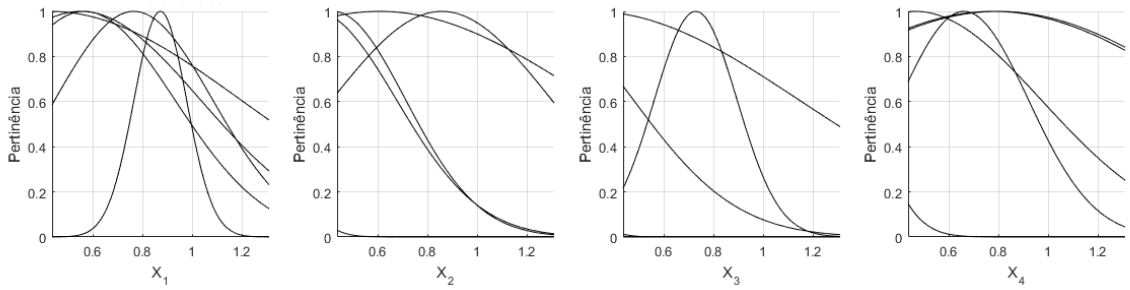

Figura 37 – Funções de pertinência finais para previsão da série temporal de Mackey-Glass pelo NFN-PG-MG-III.

Tabela 13 – Desempenho na previsão da série temporal de Mackey-Glass.

Modelo	RMSE (min)	RMSE (med)	Desv. Pad
ANFIS	0,0185148	0,0185148	0,0000000
NFN-PG-MG-III	0,0508581	0,0553154	0,0024686
NFN-PG-MG-I	0,0528963	0,0561310	0,0017597
NFN	0,0534159	0,0579309	0,0025032
NFN-PG-MG-II	0,0595553	0,0600545	0,0002393
MLP-3 camadas	0,1130268	0,1357693	0,0203024
MLP-2 camadas	0,1064175	0,1382744	0,0259236

Após aplicar o teste ANOVA aos modelos propostos, obteve-se um p-valor de $1,07 \times 10^{-9}$, concluindo que há diferença estatística entre os modelos. O modelo NFN-PG-MG-III, que obteve menor RMSE médio foi comparado com os modelos alternativos

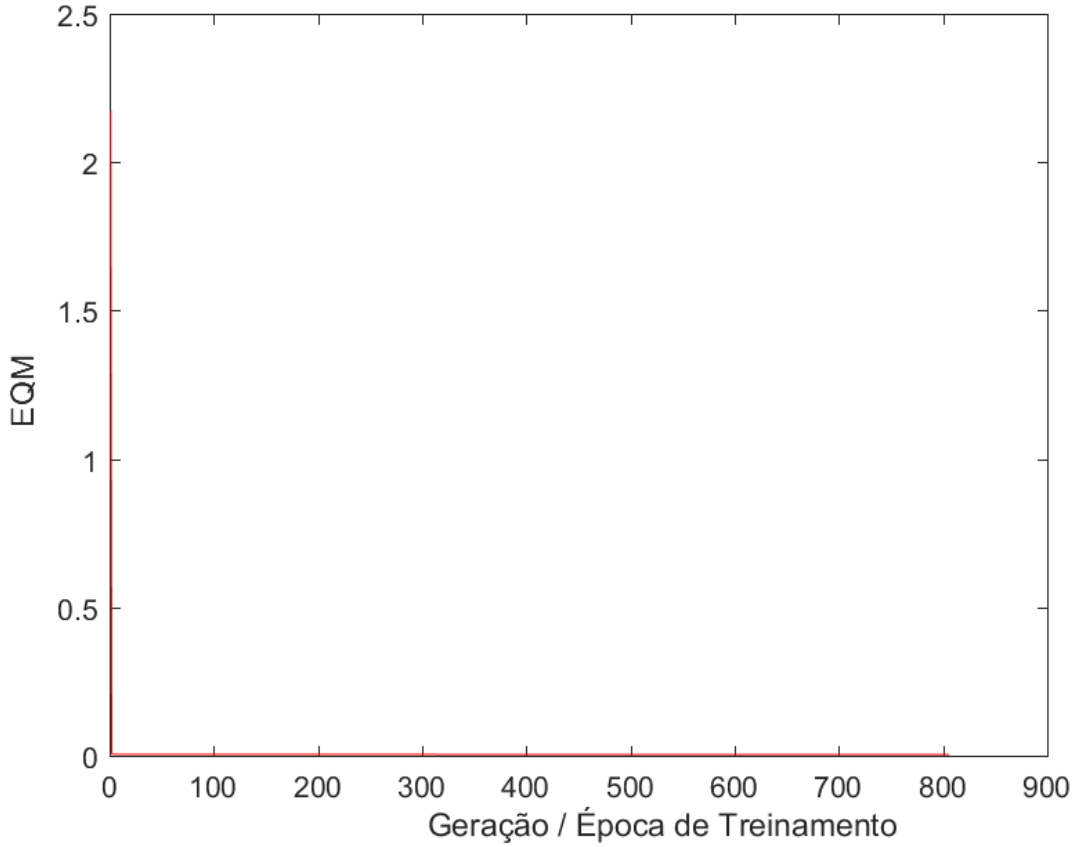

Figura 38 – Convergência do algoritmo NFN-PG-MG-I para previsão da série temporal de Mackey-Glass.

e os resultados podem ser vistos na Tabela 14, que mostra que o NFN-PG-MG-III demonstrou-se também superior ao modelo NFN e aos modelos MLP com 2 e 3 camadas e modelo ANFIS superior a todos para previsão da série temporal de Mackey-Glass.

Tabela 14 – Teste ANOVA do modelo NFN-PG-MG-III na série temporal de Mackey-Glass.

Modelo	p-valor
NFN-PG-MG-III vs NFN	0,01204
NFN-PG-MG-III vs MLP-2 camadas	6,84 $\times 10^{-9}$
NFN-PG-MG-III vs MLP-3 camadas	2,50 $\times 10^{-10}$
NFN-PG-MG-III vs ANFIS	3,33 $\times 10^{-17}$

Capítulo 5

Considerações Finais

Este capítulo apresenta as considerações finais do trabalho. A Seção 5.1 discute as conclusões a partir dos resultados alcançados e a Seção 5.2 apresentar as perspectivas de continuidade da pesquisa.

5.1 Conclusão

Este trabalho introduziu três novos algoritmos baseados em Programação Genética Multi-Gene para construção da base de regras em redes Neo-*Fuzzy*-Neuron, sendo que cada um deles possui duas versões que se diferenciam pelo número de funções de pertinência inicial ser fixo (versão 1) ou variável (versão 2), totalizando seis novos modelos. As abordagens utilizam a PG para criação e ajuste das funções de pertinência e um método baseado no Gradiente para ajuste dos parâmetros dos consequentes.

Experimentos computacionais em problemas de previsão e identificação de sistemas sugerem que a abordagem seja promissora. Os modelos propostos se demonstram eficientes quando comparados com modelos alternativos, em especial o NFN-PG-MG-I que se demonstrou estatisticamente superior na maioria dos casos. Isso demonstra que fazer o ajuste de parâmetros e funções simultaneamente pode trazer ganhos ao algoritmo, além de, no caso da abordagem apresentada, permitir que a PG e o método do Gradiente cooperem entre si. O NFN-PG-MG-III também se demonstrou eficiente, superando o primeiro em um dos experimentos, porém o algoritmo possui mais passos e é mais complexo devido a exigir um particionamento do universo de discurso no início do processo, o que exige alguns cálculos para definir os parâmetros da funções (centro e espalhamento) que são dispensáveis no NFN-GP-MG-I que possui modelagem mais simples. O NFN-PG-MG-II se demonstrou inferior na maioria dos casos, concluindo que, caso o ajuste de pesos e de funções não seja feito simultaneamente como no NFN-PG-MG-I, é essencial que haja um ajuste de pesos inicial como no NFN-PGM-MG-III.

Ainda é possível observar que, na maioria dos casos, a versão 1 dos algoritmos se demonstraram superiores à versão dois, o que significa que os modelos apresentados possuem melhor desempenho quando o número de funções de pertinência inicial é um parâmetro de entrada definido pelo usuário. Isso sugere que ainda é necessário analisar possíveis alterações nos algoritmos para que sejam autônomos não só para criar as regras mas também para definir a quantidade de regras que devem compor a base.

Por fim pode-se concluir que a Programação Genética se demonstrou uma técnica eficiente para criação da base de regras em redes NFN, especialmente quando aplicadas em problemas de previsão e identificação de sistemas e os modelos apresentados se demonstraram uma boa alternativa aos modelos convencionais.

5.2 Perspectivas de continuidade

Com base nos experimentos realizados e nas conclusões extraídas, sugere-se como propostas de trabalhos futuros:

- estudo de técnicas para exclusão de funções de pertinência redundantes;
- utilização da Programação Genética também para ajuste dos pesos, modelando as funções com seus pesos como indivíduos da PG;
- utilização de funções de pertinência triangulares e complementares na rede NFN com taxa de aprendizado ótima e Gaussianas também com taxa de aprendizado ótima;
- utilização de outros modelos *Neuro-Fuzzy*, incluindo modelos de Takagi-Sugeno de ordem 1;
- alteração dos métodos de cruzamento, responsáveis pela inclusão e exclusão de regras, de forma a tornar os algoritmos com número de regras variável mais eficientes;
- implementação de outros operadores de cruzamento na PG-MG, considerando utilizar cruzamento em alto nível e cruzamento em apenas um ponto;
- implementação de outros operadores de mutação na PG-MG, considerando utilizar mutação de vários pontos e mutação em alto nível.

Referências

ABADEH, M. S.; MOHAMADI, H.; HABIBI, J. Design and analysis of genetic fuzzy systems for intrusion detection in computer networks. **Expert Systems with Applications**, Elsevier, v. 38, n. 6, p. 7067–7075, jun. 2011. ISSN 0957-4174. Citado na página 30.

ABRAHAM, A. Neuro fuzzy systems: State-of-the-art modeling techniques. In: MIRA, J.; PRIETO, A. (Ed.). **Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 269–276. ISBN 978-3-540-45720-6. Citado 3 vezes nas páginas 2, 10 e 11.

ANDRADE, L. C. M. de; FLAUZINO, R. A.; SILVA, I. N. da. Sistemas fuzzy para previsão de demanda de energia elétrica no curtíssimo prazo. 2010. Citado na página 1.

ASHISH, K. et al. Genetic-neuro-fuzzy system for grading depression. **Applied computing and informatics**, Elsevier, v. 14, n. 1, p. 98–105, 2018. Citado 2 vezes nas páginas 1 e 32.

AZAD, R. M. A.; RYAN, C. A simple approach to lifetime learning in genetic programming-based symbolic regression. **Evolutionary computation**, MIT Press, v. 22, n. 2, p. 287–317, 2014. Citado na página 21.

AZNARTE, J. L. et al. Financial time series forecasting with a bio-inspired fuzzy model. **Expert Syst. Appl.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 39, n. 16, p. 12302–12309, nov. 2012. ISSN 0957-4174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2012.02.135>. Citado 2 vezes nas páginas 1 e 30.

BANZHAF, W. et al. **Genetic programming: an introduction**. [S.l.]: Morgan Kaufmann San Francisco, 1998. v. 1. Citado 2 vezes nas páginas 21 e 24.

BEYER, H.-G.; SCHWEFEL, H.-P. Evolution strategies–a comprehensive introduction. **Natural computing**, Springer, v. 1, n. 1, p. 3–52, 2002. Citado na página 3.

BODYANSKIY, Y.; KOKSHENEV, I.; KOLODYAZHNIY, V. An adaptive learning algorithm for a neo fuzzy neuron. In: CITESEER. **EUSFLAT Conf.** [S.l.], 2003. p. 375–379. Citado na página 13.

BODYANSKIY, Y.; KULISHOVA, N.; MALYSHEVA, D. The multidimensional extended neo-fuzzy system and its fast learning for emotions online recognition. In: IEEE. **2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)**. [S.l.], 2018. p. 473–477. Citado na página 18.

BODYANSKIY, Y. et al. The autoencoder based on generalized neo-fuzzy neuron and its fast learning for deep neural networks. In: **2018 IEEE Second International Conference on Data Stream Mining Processing (DSMP)**. [S.l.: s.n.], 2018. p. 113–118. Citado na página 18.

BODYANSKIY, Y.; PLISS, I.; VYNOKUROVA, O. Flexible neo-fuzzy neuron and neuro-fuzzy network for monitoring time series properties. **Information Technology and Management Science**, Versita, v. 16, n. 1, p. 47–52, 2013. Citado na página 17.

BODYANSKIY, Y. et al. Hybrid neuro-neo-fuzzy system and its adaptive learning algorithm. In: **2015 Xth International Scientific and Technical Conference "Computer Sciences and Information Technologies"(CSIT)**. [S.l.: s.n.], 2015. p. 111–114. Citado na página 17.

BODYANSKIY, Y. V.; TYSHCHENKO, O. K.; KOPALIANI, D. S. An extended neo-fuzzy neuron and its adaptive learning algorithm. **CoRR**, abs/1610.06483, 2016. Disponível em: <http://arxiv.org/abs/1610.06483>. Citado na página 18.

CARMONA, C. et al. A fuzzy genetic programming-based algorithm for subgroup discovery and the application to one problem of pathogenesis of acute sore throat conditions in humans. **Inf. Sci.**, Elsevier Science Inc., New York, NY, USA, v. 298, n. C, p. 180–197, mar. 2015. ISSN 0020-0255. Disponível em: <https://doi.org/10.1016/j.ins.2014.11.030>. Citado na página 1.

CARMONA, C. J. et al. Fuzzy rules for describing subgroups from influenza a virus using a multi-objective evolutionary algorithm. **Applied Soft Computing**, Elsevier, v. 13, n. 8, p. 3439–3448, 2013. Citado na página 30.

CARMONA, C. J. et al. Mefes: an evolutionary proposal for the detection of exceptions in subgroup discovery. an application to concentrating photovoltaic technology. **Knowledge-Based Systems**, Elsevier, v. 54, p. 73–85, 2013. Citado 2 vezes nas páginas 1 e 31.

CARMONA, C. J. et al. Evolutionary fuzzy rule extraction for subgroup discovery in a psychiatric emergency department. **Soft Computing**, Springer, v. 15, n. 12, p. 2435–2448, 2011. Citado na página 29.

CARMONA, C. J. et al. Nmeef-sd: non-dominated multiobjective evolutionary algorithm for extracting fuzzy rules in subgroup discovery. **IEEE Transactions on Fuzzy Systems**, IEEE, v. 18, n. 5, p. 958–970, 2010. Citado na página 29.

CHEN, C.-H.; LIN, C.-J.; LIN, C.-T. Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, IEEE, v. 39, n. 4, p. 459–473, 2009. Citado na página 1.

CORDÓN, O. et al. Ten years of genetic fuzzy systems: current framework and new trends. **Fuzzy sets and systems**, Elsevier, v. 141, n. 1, p. 5–31, 2004. Citado 2 vezes nas páginas 3 e 28.

CORDÓN, O. et al. **Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases**. World Scientific, 2001. (Advances in fuzzy systems - applications and theory). ISBN 9789810240165. Disponível em: <https://books.google.com.br/books?id=bwa9QgAACAAJ>. Citado 2 vezes nas páginas 27 e 28.

DASH, R.; DASH, P. Efficient stock price prediction using a self evolving recurrent neuro-fuzzy inference system optimized through a modified technique. **Expert Syst. Appl.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 52, n. C, p. 75–90, jun. 2016. ISSN 0957-4174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2016.01.016>. Citado na página 1.

EIBEN, A. E.; SMITH, J. E. et al. **Introduction to evolutionary computing**. [S.l.]: Springer, 2003. v. 53. Citado na página 23.

ELHAG, S. et al. On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. **Expert Syst. Appl.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 42, n. 1, p. 193–202, jan. 2015. ISSN 0957-4174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2014.08.002>. Citado 2 vezes nas páginas 1 e 31.

FAHIM, M. et al. Efm: evolutionary fuzzy model for dynamic activities recognition using a smartphone accelerometer. **Applied intelligence**, Springer, v. 39, n. 3, p. 475–488, 2013. Citado na página 31.

FERNANDEZ, A. et al. Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges. **Knowledge-Based Systems**, v. 80, p. 109 – 121, 2015. ISSN 0950-7051. 25th anniversary of Knowledge-Based Systems. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950705115000209>. Citado 2 vezes nas páginas 26 e 27.

FERREIRA, C. Gene expression programming: a new adaptive algorithm for solving problems. **Complex Systems**, v. 13, n. 2, p. 87–129, 2001. Cite arxiv:cs/0102027Comment: 22 pages, 17 figures. Disponível em: <http://arxiv.org/abs/cs/0102027>. Citado 2 vezes nas páginas 25 e 33.

FULCHER, J. Computational intelligence: an introduction. In: **Computational intelligence: a compendium**. [S.l.]: Springer, 2008. p. 3–78. Citado 2 vezes nas páginas 6 e 9.

FULLÉR, R. **Neural fuzzy systems**. [S.l.]: Citeseer, 1995. Citado na página 10.

GALLOVA, S. Neuro-fuzzy learning and genetic algorithm approach with chaos theory principles applying for diagnostic problem solving. In: **Proceedings of the World Congress on Engineering**. [S.l.: s.n.], 2009. v. 1, p. 9. Citado 2 vezes nas páginas 1 e 29.

GANGA, G. M. D.; CARPINETTI, L. C. R.; POLITANO, P. R. Gestão do desempenho em cadeias de suprimentos usando lógica fuzzy. **Gestão & Produção, São Carlos**, Citeseer, v. 18, n. 4, p. 755–774, 2011. Citado na página 6.

GOLDBERG, D. E.; HOLLAND, J. H. Genetic algorithms and machine learning. **Machine learning**, Springer, v. 3, n. 2, p. 95–99, 1988. Citado na página 3.

GUÉLY, F.; SIARRY, P. Gradient descent method for optimizing various fuzzy rule bases. In: IEEE. **[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems**. [S.l.], 1993. p. 1241–1246. Citado na página 2.

HAN, L. et al. Neuro-fuzzy control based on on-line least square support vector machines. In: IEEE. **2015 34th Chinese Control Conference (CCC)**. [S.l.], 2015. p. 3411–3416. Citado na página 1.

HANG, J.; ZHANG, J.; CHENG, M. Application of multi-class fuzzy support vector machine classifier for fault diagnosis of wind turbine. **Fuzzy Sets Syst.**, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 297, n. C, p. 128–140,

ago. 2016. ISSN 0165-0114. Disponível em: <https://doi.org/10.1016/j.fss.2015.07.005>. Citado na página 1.

HERRERA, F. Genetic fuzzy systems: taxonomy, current research trends and prospects. **Evolutionary Intelligence**, v. 1, n. 1, p. 27–46, 2008. Disponível em: <http://dblp.uni-trier.de/db/journals/evi/evi1.html#Herrera08>. Citado 2 vezes nas páginas 2 e 26.

HOLLAND, J. H.; REITMAN, J. S. Cognitive systems based on adaptive algorithms. In: . New York, NY, USA: ACM, 1977. p. 49–49. Disponível em: <http://doi.acm.org/10.1145/1045343.1045373>. Citado na página 28.

HU, Z.; BODYANSKIY, Y. V.; TYSHCHENKO, O. K. A deep cascade neural network based on extended neo-fuzzy neurons and its adaptive learning algorithm. In: **2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)**. [S.l.: s.n.], 2017. p. 801–805. Citado na página 18.

IBÁNEZ, O. et al. Modeling the skull–face overlay uncertainty using fuzzy sets. **IEEE Transactions on Fuzzy Systems**, IEEE, v. 19, n. 5, p. 946–959, 2011. Citado na página 30.

JANG, J.-S. R.; SUN, C.-T.; MIZUTANI, E. Neuro-fuzzy and soft computing; a computational approach to learning and machine intelligence. CUMINCAD, 1997. Citado na página 6.

JOHANSEN, T. A.; SHORTEN, R.; MURRAY-SMITH, R. On the interpretation and identification of dynamic takagi-sugeno fuzzy models. **IEEE Transactions on Fuzzy systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 8, n. 3, p. 297–313, 2000. Citado na página 8.

JUANG, C.-F. A tsk-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. **Trans. Fuz Sys.**, IEEE Press, Piscataway, NJ, USA, v. 10, n. 2, p. 155–170, abr. 2002. ISSN 1063-6706. Disponível em: <http://dx.doi.org/10.1109/91.995118>. Citado na página 1.

KACPRZYK, J.; PEDRYCZ, W. **Springer Handbook of Computational Intelligence**. [S.l.]: Springer Publishing Company, Incorporated, 2015. ISBN 3662435047, 9783662435045. Citado na página 5.

KARYOTIS, C. et al. A fuzzy modelling approach of emotion for affective computing systems. p. 453–460, 2016. The full text is currently unavailable on the repository.; International Conference on Internet of Things and Big Data, IoTBD 2016 ; Conference date: 23-04-2016 Through 25-04-2016. Citado na página 1.

KASABOV, N.; FILEV, D. Evolving intelligent systems: Methods, learning, applications. In: **2006 International Symposium on Evolving Fuzzy Systems**. [S.l.: s.n.], 2006. p. 8–18. Citado na página 2.

KASABOV, N. K.; SONG, Q. Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. **IEEE Transactions on Fuzzy Systems**, v. 10, n. 2, p. 144–154, April 2002. ISSN 1063-6706. Citado na página 62.

KAUR, A.; BHARDWAJ, A.; BEEN, U. A. H. Genetic neuro fuzzy system for hypertension diagnosis. **International Journal of Computer Science and Information Technologies**, Citeseer, v. 5, n. 4, p. 4986–4989, jul. 2014. ISSN 0975-9646. Citado na página 31.

KELLER DERONG LIU, D. B. F. J. M. **Fundamentals of Computational Intelligence. Neural Networks, Fuzzy Systems and Evolutionary Computation**. Wiley, 2016. ISBN 978-1-110-21434-2. Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=9705b13aad38704600306e3d4dc24372>. Citado 2 vezes nas páginas 2 e 19.

KLIR, G. J.; YUAN, B. Fuzzy sets and fuzzy logic: theory and applications. **Possibility Theory versus Probab. Theory**, v. 32, n. 2, p. 207–208, 1996. Citado na página 15.

KOSHIYAMA, A. S.; VELLASCO, M. M.; TANSCHEIT, R. Gpfis-class: A genetic fuzzy system based on genetic programming for classification problems. **Appl. Soft Comput.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 37, n. C, p. 561–571, dez. 2015. ISSN 1568-4946. Disponível em: <https://doi.org/10.1016/j.asoc.2015.08.055>. Citado 2 vezes nas páginas 26 e 32.

KOZA, J. R. **Genetic programming: on the programming of computers by means of natural selection**. [S.l.]: MIT press, 1992. v. 1. Citado 4 vezes nas páginas 3, 20, 22 e 26.

KOZA, J. R. Genetic programming as a means for programming computers by natural selection. **Statistics and computing**, Springer, v. 4, n. 2, p. 87–112, 1994. Citado na página 21.

KREINOVICH, V.; QUINTANA, C.; REZNIK, L. Gaussian membership functions are most adequate in representing uncertainty in measurements. In: **Proceedings of NAFIPS**. [S.l.: s.n.], 1992. p. 15–17. Citado na página 13.

LANGDON, W. B. Size fair and homologous tree crossovers for tree genetic programming. **Genetic programming and evolvable machines**, Springer, v. 1, n. 1-2, p. 95–119, 2000. Citado na página 23.

LEE, C.-C. Fuzzy logic in control systems: fuzzy logic controller. i. **IEEE Transactions on systems, man, and cybernetics**, IEEE, v. 20, n. 2, p. 404–418, 1990. Citado na página 2.

LEITE, D. et al. Evolving fuzzy granular modeling from nonstationary fuzzy data streams. **Evolving Systems**, v. 3, n. 2, p. 65–79, Jun 2012. ISSN 1868-6486. Disponível em: <https://doi.org/10.1007/s12530-012-9050-9>. Citado na página 53.

LEMOS, A.; CAMINHAS, W.; GOMIDE, F. Fuzzy evolving linear regression trees. **Evolving Systems**, v. 2, n. 1, p. 1–14, Mar 2011. ISSN 1868-6486. Disponível em: <https://doi.org/10.1007/s12530-011-9028-z>. Citado na página 47.

LI, H.; WANG, Y.; ZHANG, G. Probabilistic fuzzy classification for stochastic data. **IEEE Transactions on Fuzzy Systems**, v. 25, n. 6, p. 1391–1402, Dec 2017. ISSN 1063-6706. Citado na página 1.

LIU, Y.-H.; CHEN, Y.-T. Face recognition using total margin-based adaptive fuzzy support vector machines. **Trans. Neur. Netw.**, IEEE Press, Piscataway, NJ, USA, v. 18, n. 1, p. 178–192, jan. 2007. ISSN 1045-9227. Disponível em: <http://dx.doi.org/10.1109/TNN.2006.883013>. Citado na página 1.

MACKEY, M.; GLASS, L. Oscillation and chaos in physiological control systems. **Science**, American Association for the Advancement of Science, v. 197, n. 4300, p. 287–289, 1977. ISSN 0036-8075. Disponível em: <https://science.sciencemag.org/content/197/4300/287>. Citado na página 61.

MAIA, R. D. **Projeto de Estruturas Neuro-Fuzzy do Tipo Takagi-Sugeno Utilizando Programação Genética**. Tese (Doutorado) — Universidade Federal de Minas Gerais, 2005. Citado na página 28.

MALLINSON, H.; BENTLEY, P. Evolving fuzzy rules for pattem classification. **Computational Intelligence for Modelling, Control & Automation: Evolutionary Computation & Fuzzy Logic for Intelligent Control, Knowledge Acquisition & Information Retrieval**, IOS Press, v. 55, p. 184, 1999. Citado na página 2.

MAMDANI, E.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. **Int. J. Hum.-Comput. Stud.**, Academic Press, Inc., Duluth, MN, USA, v. 51, n. 2, p. 135–147, ago. 1999. ISSN 1071-5819. Disponível em: <http://dx.doi.org/10.1006/ijhc.1973.0303>. Citado na página 1.

MAMDANI, E. H.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. **International Journal of Man-Machine Studies**, v. 7, n. 1, p. 1–13, 1975. Disponível em: <http://dblp.uni-trier.de/db/journals/ijmms/ijmms7.html#MamdaniA75>. Citado na página 8.

MATÍA, F.; MARICHAL, G. N.; JIMÉNEZ, E. **Fuzzy Modeling and Control: Theory and Applications**. [S.l.]: Springer, 2014. ISBN 978-94-6239-081-2. Citado 2 vezes nas páginas 2 e 8.

MELIN, P. et al. Optimal design of fuzzy classification systems using pso with dynamic parameter adaptation through fuzzy logic. **Expert Syst. Appl.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 40, n. 8, p. 3196–3206, jun. 2013. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2012.12.033>. Citado na página 1.

MENDES, J. et al. Neo-fuzzy neuron learning using backfitting algorithm. **Neural Computing and Applications**, p. 1–10, 2017. Disponível em: <https://app.dimensions.ai/details/publication/pub.1100113729>. Citado na página 18.

MIKI, T.; YAMAKAWA, T. Analog implementation of neo-fuzzy neuron and its on-board learning. **Computational Intelligence and Applications**, WSES Press, p. 144–149, 1999. Citado na página 12.

MIRANIAN, A.; ABDOLLAHZADE, M. Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 24, n. 2, p. 207–218, 2013. Citado na página 1.

MITRA, S.; HAYASHI, Y. Neuro-fuzzy rule generation: survey in soft computing framework. **IEEE transactions on neural networks**, IEEE, v. 11, n. 3, p. 748–768, 2000. Citado na página 2.

MODELLING the service life of rendered facades using fuzzy systems. **Automation in Construction**, v. 51, n. Complete, p. 1–7, 2015. Citado na página 1.

MONTGOMERY, D. C. **Design and analysis of experiments**. [S.l.]: John wiley & sons, 2017. Citado na página 45.

MYLONAS, S. K.; STAVRAKOUDIS, D. G.; THEOCHARIS, J. B. Genesis: A ga-based fuzzy segmentation algorithm for remote sensing images. **Knowledge-Based Systems**, Elsevier, v. 54, p. 86–102, 2013. Citado 2 vezes nas páginas 1 e 31.

OLIVEIRA, L. O. V. et al. Sequential symbolic regression with genetic programming. In: **Genetic Programming Theory and Practice XII**. [S.l.]: Springer, 2015. p. 73–90. Citado na página 21.

OMISORE, M. O.; SAMUEL, O. W.; ATAJEROMAVWO, E. J. A genetic-neuro-fuzzy inferential model for diagnosis of tuberculosis. **Applied Computing and Informatics**, Elsevier, v. 13, n. 1, p. 27–37, 2017. Citado 2 vezes nas páginas 1 e 32.

POLI, R.; LANGDON, W. B.; MCPHEE, N. F. **A Field Guide to Genetic Programming**. [S.l.]: Lulu Enterprises, UK Ltd, 2008. ISBN 1409200736, 9781409200734. Citado 3 vezes nas páginas 20, 22 e 25.

RADHAKRISHNA, V. et al. A novel fuzzy gaussian-based dissimilarity measure for discovering similarity temporal association patterns. **Soft Comput.**, Springer-Verlag, Berlin, Heidelberg, v. 22, n. 6, p. 1903–1919, mar. 2018. ISSN 1432-7643. Disponível em: <https://doi.org/10.1007/s00500-016-2445-y>. Citado na página 1.

RAMSAY, J. Functional data analysis. **Encyclopedia of Statistics in Behavioral Science**, Wiley Online Library, 2005. Citado na página 46.

ROSS, T. J. **Fuzzy logic with engineering applications**. [S.l.]: John Wiley & Sons, 2005. Citado na página 15.

SETLAK, G. et al. Deep evolving stacking convex cascade neo-fuzzy network and its rapid learning. In: **2018 Federated Conference on Computer Science and Information Systems (FedCSIS)**. [S.l.: s.n.], 2018. p. 29–33. Citado na página 18.

SHAMSHIRBAND, S. et al. Co-fais: cooperative fuzzy artificial immune system for detecting intrusion in wireless sensor networks. **Journal of Network and Computer Applications**, Elsevier, v. 42, p. 102–117, 2014. Citado na página 1.

SHEN, K.-Y.; TZENG, G.-H. Drsa-based neuro-fuzzy inference systems for the financial performance prediction of commercial banks. **International Journal of Fuzzy Systems**, v. 16, n. 2, 2014. Citado na página 1.

SHIHABUDHEEN, K.; PILLAI, G. Recent advances in neuro-fuzzy system. **Know.-Based Syst.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 152, n. C, p. 136–162, jul. 2018. ISSN 0950-7051. Disponível em: <https://doi.org/10.1016/j.knosys.2018.04.014>. Citado 2 vezes nas páginas 10 e 11.

SILVA, A. M. et al. A fast learning algorithm for evolving neo-fuzzy neuron. **Appl. Soft Comput.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 14, p. 194–209, jan. 2014. ISSN 1568-4946. Disponível em: <https://doi.org/10.1016/j.asoc.2013.03.022>. Citado 3 vezes nas páginas vii, 12 e 47.

SILVA, A. M. et al. Evolving neural fuzzy network with adaptive feature selection. In: **2012 11th International Conference on Machine Learning and Applications**. [S.l.: s.n.], 2012. v. 2, p. 440–445. Citado na página 53.

SMITH, S. F. **A Learning System Based on Genetic Adaptive Algorithms**. Tese (Doutorado), Pittsburgh, PA, USA, 1980. AAI8112638. Citado 2 vezes nas páginas 28 e 33.

SOLIMAN, M. A. et al. Hybrid anfis-ga-based control scheme for performance enhancement of a grid-connected wind generator. **IET Renewable Power Generation**, IET, v. 12, n. 7, p. 832–843, maio 2018. ISSN 1752-1424. Citado na página 32.

SOUALHI, A. et al. Long-term prediction of bearing condition by the neo-fuzzy neuron. In: **2013 9th IEEE International Symposium on Diagnostics for Electric Machines, Power Electronics and Drives (SDEMPED)**. [S.l.: s.n.], 2013. p. 586–591. Citado na página 17.

STORN, R.; PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. **Journal of global optimization**, Springer, v. 11, n. 4, p. 341–359, 1997. Citado na página 3.

SUGENO, M.; KANG, G. T. Structure identification of fuzzy model. **Fuzzy Sets Syst.**, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 28, n. 1, p. 15–33, out. 1988. ISSN 0165-0114. Disponível em: <http://dx.doi.org/10.1016/0165-0114(88)90113-3>. Citado na página 17.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. **IEEE Transactions on Systems, Man, and Cybernetics**, SMC-15, n. 1, p. 116–132, Jan 1985. ISSN 0018-9472. Citado 3 vezes nas páginas 8, 10 e 12.

TANAKA, K.; WANG, H. O. **Fuzzy control systems design and analysis: a linear matrix inequality approach**. [S.l.]: John Wiley & Sons, 2004. Citado na página 9.

THOMAS, S. et al. Prediction of ground motion parameters using randomized anfis (ranfis). **Appl. Soft Comput.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 40, n. C, p. 624–634, mar. 2016. ISSN 1568-4946. Disponível em: <https://doi.org/10.1016/j.asoc.2015.12.013>. Citado na página 1.

TIBIRIÇÁ, C. A. G. et al. Uma abordagem híbrida fuzzy-bayesiana para modelagem de incertezas. Florianópolis, SC, 2005. Citado na página 1.

TODOROV, Y.; TERZIYSKA, M. Simple heuristic approach for training of type-2 neo-fuzzy neural network. In: **2015 20th International Conference on Process Control (PC)**. [S.l.: s.n.], 2015. p. 278–283. Citado na página 17.

VENTURINI, G. Sia: A supervised inductive algorithm with genetic search for learning attributes based concepts. In: BRAZDIL, P. B. (Ed.). **Machine Learning: ECML-93**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993. p. 280–296. ISBN 978-3-540-47597-2. Citado na página 28.

VICTOIRE, T. A.; SAKTHIVEL, M. A refined differential evolution algorithm based fuzzy classifier for intrusion detection. **European Journal of Scientific Research**, v. 65, n. 2, p. 246–259, 2011. Citado 2 vezes nas páginas 1 e 30.

WANG, C. et al. Fuzzy b-spline membership function (bmf) and its applications in fuzzy-neural control. **Proceedings of the IEEE International Conference on Systems, Man and Cybernetics**, Institute of Electrical and Electronics Engineers Inc., v. 2, p. 2008–2014, 12 1994. ISSN 0884-3627. Citado na página 18.

WANG, J.-S.; LEE, C. S. Self-adaptive neuro-fuzzy inference systems for classification applications. **Trans. Fuz Sys.**, IEEE Press, Piscataway, NJ, USA, v. 10, n. 6, p. 790–802, dez. 2002. ISSN 1063-6706. Disponível em: <http://dx.doi.org/10.1109/TFUZZ.2002.805880>. Citado na página 2.

WILLIAMS, S. O.; OLATUNJI, O. M. Genetic neuro-fuzzy system for the diagnosis of typhoid fever. In: **2013 Conference on Medical Innovation and Computing Service**. [S.l.: s.n.], 2013. Citado na página 30.

WU, Q. Hybrid fuzzy support vector classifier machine and modified genetic algorithm for automatic car assembly fault diagnosis. **Expert Syst. Appl.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 38, n. 3, p. 1457–1463, mar. 2011. ISSN 0957-4174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2010.07.052>. Citado na página 1.

YAMAKAWA, T. et al. A new effective learning algorithm for a neo fuzzy neuron model. In: KOREAN INSTITUTE OF INTELLIGENT SYSTEMS. **Proceedings of the Korean Institute of Intelligent Systems Conference**. [S.l.], 1993. p. 1017–1020. Citado na página 12.

YAMAKAWA, T. et al. 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**. [S.l.]: IEEE, 1992. p. 477–484. ISBN 0780311043. Citado 4 vezes nas páginas 3, 11, 13 e 16.

YANG, Y.-H.; LIU, C.-C.; CHEN, H. H. Music emotion classification: A fuzzy approach. In: ACM. **Proceedings of the 14th ACM international conference on Multimedia**. [S.l.], 2006. p. 81–84. Citado na página 1.

ZADEH, L. Fuzzy sets. **Information and Control**, v. 8, n. 3, p. 338 – 353, 1965. ISSN 0019-9958. Disponível em: <http://www.sciencedirect.com/science/article/pii/S001999586590241X>. Citado 2 vezes nas páginas 1 e 5.

ZAYCHENKO, Y.; GASANOV, A. Investigations of cascade neo-fuzzy neural networks in the problem of forecasting at the stock exchange. In: **2012 IV International Conference "Problems of Cybernetics and Informatics"(PCI)**. [S.l.: s.n.], 2012. p. 1–3. Citado na página 13.

ZURITA, D. et al. Industrial time series modelling by means of the neo-fuzzy neuron. **IEEE Access**, v. 4, p. 6151–6160, 2016. ISSN 2169-3536. Citado na página 17.