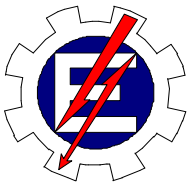


UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PROPOSTA DE UM MODELO DE REDE NEURO-FUZZY-POLINOMIAL OTIMIZADO POR ALGORITMOS DE ENXAME APLICADO À PREVISÃO

Yvo Marcelo Chiaradia Masselli

Itajubá, junho de 2009

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Yvo Marcelo Chiaradia Masselli

PROPOSTA DE UM MODELO DE REDE NEURO-FUZZY-POLINOMIAL OTIMIZADO POR ALGORITMOS DE ENXAME APLICADO À PREVISÃO

Tese submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Doutor Ciências em Engenharia Elétrica.
Área de concentração: Sistemas Elétricos de Potência

Orientador: Prof. Germano Lambert Torres
Co-Orientador: Prof. Luiz Eduardo Borges da Silva

Junho de 2009
Itajubá - MG

À minha família sempre unida e presente, física e espiritualmente.

À Deus, que ampara, conforta e fortalece o sentimento de luta e persistência

Agradecimentos

Meus sinceros agradecimentos:

- Ao Professor Germano Lambert Torres pela orientação, amizade, compreensão e paciência, que foram fundamentais ao desenvolvimento deste trabalho;
- Aos colegas da UNIFEI pelo apoio e colaboração;
- À Universidade Federal de Itajubá – UNIFEI, através do Programa de Pós Graduação em Engenharia Elétrica, pela oportunidade;
- À CAPES pelo auxílio.

RESUMO

O processo de previsão é útil em diversas atividades humanas. É nele que se baseiam os desenvolvimentos futuros, as etapas de um planejamento e as verificações de disponibilidade dos sistemas. É sabido que quanto maior for o horizonte de previsão mais difícil se torna acertar os valores previstos face aos que realmente ocorrem. E mais, que a qualidade do processo de previsão está intimamente ligada à qualidade da base histórica dos dados disponível e da repetibilidade desse conjunto de dados.

Esta tese apresenta um estudo sobre as principais técnicas de inteligência artificial, operando isoladamente ou em cooperação, de forma a compor poderosos sistemas híbridos. Estes são aplicados em problemas de modelagem e identificação de sistemas complexos das mais diversas naturezas. Em seguida é proposto um novo modelo, baseado em redes neurais polinomiais e lógica difusa, otimizados pela técnica de otimização por enxame de partículas para a previsão de sistemas.

Para a comprovação da viabilidade, são realizados diversos testes envolvendo todas as técnicas apresentadas, e os resultados são comparados com os obtidos pelo método proposto.

Palavras – Chave: Otimização por Enxame, Redes Neurais, Previsão, Sistemas Inteligentes, Sistemas Híbridos

ABSTRACT

The forecasting process is very useful in many human activities. Future developments, planning steps and system availability are based on forecasting process. It is very-well known that how much bigger will be the forecasting time horizon more dificult makes right the foreseen values to really occurs. In addition, the quality of the forecast process is closely on with the quality of the historical base of the data available and the repeatability of this data set.

This thesis presents a study on the main techniques of artificial intelligence, operating separately or in cooperation, of form to compose powerful hybrid systems. These are applied in problems of modeling and identification of complex systems of the most diverse natures.

After is proposed a new model, based on polynomial nets is considered neural and fuzzy logic, optimized for the technique of particle swarm optimization.

For the evidence of the viability, all are carried through diverse tests involving the presented techniques, and the results are compared with the gotten ones for the considered method.

Keywords: Particle swarm optimization, Neural Networks, Forecasting, Intelligent Systems, Hybrid Systems

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1
INTRODUÇÃO

A cada dia surgem novas metodologias envolvendo técnicas de IA para a solução de complexos problemas do mundo real.

Dentre as áreas de maior interesse está aquela voltada à criação de modelos representativos de sistemas.

Com o avanço da tecnologia, estes se tornam cada vez mais complexos e difíceis de serem compreendidos. Quando colocados em operação, podem assumir comportamentos indesejados ou até então inesperados. Por este motivo, quando se dispõe de um modelo que represente de forma fiel o comportamento de suas variáveis, é possível prever situações futuras com base em um histórico.

A lógica difusa, proposta por Zadeh [1] é a técnica de IA capaz de representar, de forma simples, sistemas onde suas variáveis possuam inconsistências e não-linearidades. Isto faz com que apresente resultados satisfatórios quando comparados aos métodos matemáticos tradicionais.

No entanto, o projeto de um sistema difuso exige um tipo conhecimento prévio, normalmente fornecido por um especialista, o que muitas vezes a inviabiliza.

Para solucionar este problema, Jang [2] propôs um método de geração automática do conhecimento necessário ao sistema difuso, através de uma estrutura similar a uma rede neural artificial do tipo multicamadas [3], denominado ANFIS.

A partir da proposta de Jang surgem, constantemente, diversos outros modelos de criação automática de sistemas difusos. Exemplos mais recentes e bem sucedidos podem ser facilmente encontrados na literatura [4, 5, 6].

No intuito de aprimorar tais sistemas, são a eles associadas técnicas de otimização como

algoritmos genéticos. Estes tem como vantagem a simplicidade de implementação e o fato de não utilizarem qualquer informação de gradiente, evitando assim, problemas com mínimos locais.

Este trabalho apresenta um estudo sobre as principais técnicas de inteligência artificial utilizadas na extração de conhecimento em uma série de dados, o que é de fundamental importância para a criação de modelos baseados em lógica difusa.

A aplicação de previsão de valores futuros é utilizada para verificar a viabilidade das técnicas apresentadas.

Em seguida é proposto um novo modelo, baseado em redes neurais polinomiais [7] e otimizados pela técnica de otimização por enxame de partículas [8, 9], com o propósito de gerar automaticamente um sistema difuso, capaz não só representar o sistema em questão, mas também estimar comportamentos futuros, baseados em fatos passados.

1.1 OBJETIVOS

Este trabalho possui dois objetivos: primeiramente o de apresentar um estudo capaz de auxiliar no projeto de sistemas de previsão através de um comparativo, baseado em testes práticos, entre as principais metodologias utilizadas atualmente para este fim.

O segundo, e principal, é o de propor um sistema híbrido, entre técnicas de inteligência artificial, de fácil implementação, aplicável na identificação e previsão, capaz de gerar resultados tão bons quanto os atuais sistemas e em menor tempo de processamento.

1.2 ORGANIZAÇÃO

O trabalho está organizado de forma a facilitar a compreensão das técnicas apresentadas, assim como das hibridizações decorrentes destas. No capítulo 2 são abordados os principais conceitos sobre as técnicas de inteligência artificial utilizadas na tarefa de identificação e

previsão. Estes são de fundamental importância para o entendimento dos sistemas formados pela fusão das técnicas apresentadas. Em seguida, no capítulo 3, são discutidas as principais metodologias de composição entre as técnicas anteriores. Desta forma podem-se formar poderosos sistemas híbridos, utilizados na solução de problemas complexos, com características de não-linearidade e inconsistências.

O capítulo 4 apresenta os principais conceitos sobre previsão em séries temporais, que serão a base para a compreensão do estudo de casos. O capítulo 5 se refere à primeira parte prática do trabalho. Nele são mostrados os resultados obtidos na avaliação de todas as técnicas e sistemas apresentados, quando aplicados na previsão de séries temporais.

O capítulo 6 aborda os principais fundamentos da técnica de otimização por enxame de partículas e apresenta uma nova metodologia, proposta como alternativa a todas até então apresentados. Neste também são mostrados os resultados de testes realizados para avaliação de desempenho da nova proposta e um comparativo a fim de comprovar a viabilidade desta.

CAPÍTULO 2
PRINCIPAIS TÉCNICAS DE IA

Este capítulo apresenta um estudo sobre as principais técnicas de inteligência artificial aplicadas na identificação de sistemas. São apresentados conceitos de redes neurais artificiais com topologia fixa, como as do tipo MLP, assim como redes neurais polinomiais, do tipo auto organizáveis. Além disto trás conceitos sobre sistemas baseados em lógica fuzzy, além dos algoritmos genéticos, muito utilizados em situações envolvendo otimização. Em suma, este capítulo constitui a base para a compreensão de sistemas baseados na hibridização entre as técnicas apresentadas.

2.1 Redes Neurais Artificiais

São modelos computacionais que trabalham de forma similar ao cérebro humano. Assim como a estrutura do sistema nervoso humano, as redes neurais artificiais (RNA) são formadas por elementos de processamento operando paralelamente, chamados de neurônios, interligados por conexões nervosas chamadas sinapses [10].

A este conjunto de neurônios fortemente conectados, pode-se associar algum tipo de conhecimento que será armazenado nas conexões existentes entre os neurônios adjacentes.

Ao processo de associação de conhecimento à RNA dá-se o nome de aprendizado, que é atribuído à RNA pela execução de um outro processo chamado treinamento.

Após “treinada”, estará apta a operar de maneira desejada.

Algumas de suas principais características são:

- Capacidade de Aprendizagem
- Adaptabilidade
- Capacidade de Generalização

A seguir serão dados maiores detalhes sobre projeto, desenvolvimento e implementação de uma RNA.

2.1.2 Neurônio Artificial

O neurônio artificial é tido como a principal unidade de uma rede neural artificial. Ele foi totalmente inspirado no neurônio biológico, sendo formado por um conjunto de entradas, com um "peso" associado, e uma saída, conforme mostrado na figura abaixo:

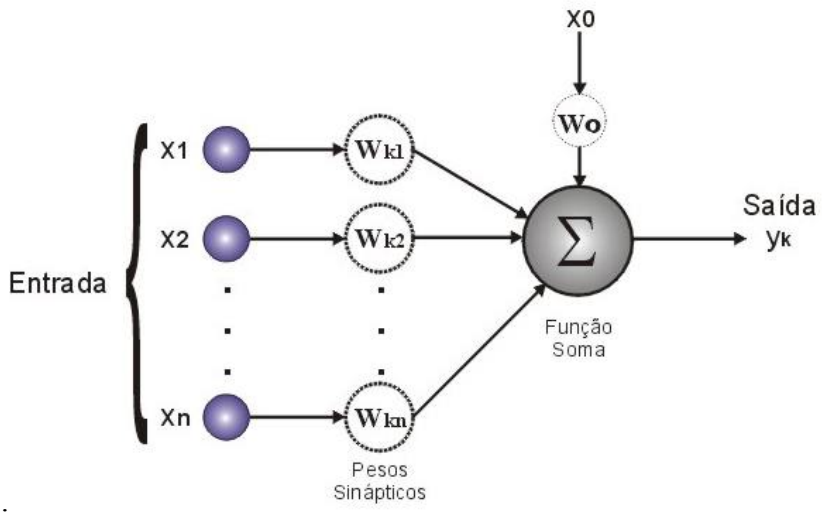


Figura 2.1.2.1: Modelo de um neurônio artificial proposto por McCulloch e Pitts [2]

Os valores apresentados às unidades de entrada, $x_1,x_2,x_3,...,x_n$, são multiplicados pelos seus respectivos pesos $W=[w_{k1},w_{k2},w_{k3},....w_{kn}]$. Desta forma, é simples notar que este expressa a "força de ligação" entre dois neurônios quaisquer [11]. O valor da entrada x_0 é sempre igual a 1 e o peso associado a esta entrada é denominado "bias" . O resultado da soma ponderada corresponde ao valor de saída y_k, expresso da seguinte forma:

$$y_k = w_0 + x_1 * w_{k1} + x_2 * w_{k2} + x_3 * w_{k3} + \ldots + x_n * w_{kn} = \sum_{i=1} (W_i * X_i) + x_0 * w_0$$

2.1.3 Perceptron

No intuito de permitir que pequenas variações na entrada do neurônio possam ser percebidas na saída, Rosenblatt [12] propôs a inserção de uma "função de ativação" capaz de reproduzir na saída estas pequenas alterações na entrada, amplificando ou atenuando os sinais que irão gerar a saída y_k do neurônio artificial proposto por McCulloch e Pitts.

Este modelo foi chamado de *Perceptron* e a função de ativação a ele incorporada pode assumir diversos tipos, como logarítmica, tangente hiperbólica e outras.

A figura a seguir ilustra o modelo proposto por Rosenblatt de acordo com [13].

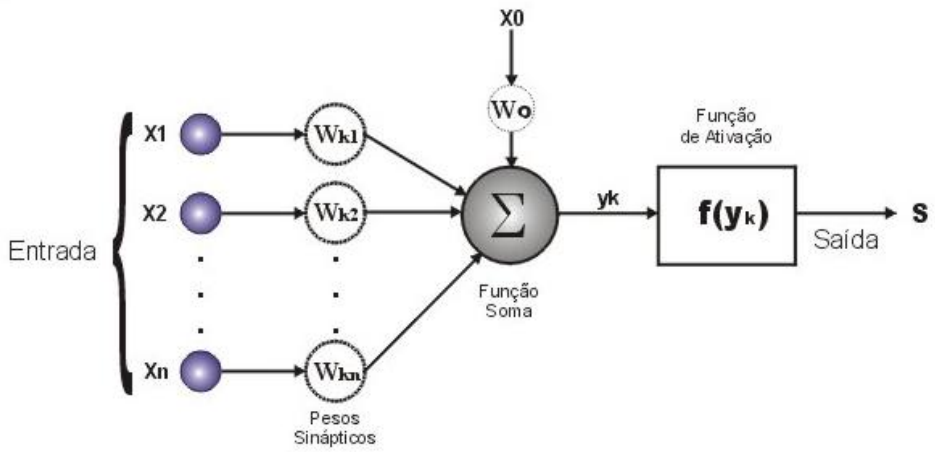


Figura 2.1.3.1: Modelo do Perceptron proposto por Rosenblatt [13]

A função de ativação é responsável por limitar a amplitude do sinal na saída do neurônio, de acordo com a função de ativação escolhida.

2.1.4 Redes com Múltiplas Camadas

Um neurônio operando isoladamente é capaz de lidar apenas com problemas onde o conjunto de entrada é linearmente separável. Caso contrário é necessário o uso de redes com múltiplas camadas de neurônios (MLP). Estas foram concebidas por Rumelhart [3], para resolver problemas onde há a necessidade de separação em classes, de acordo com os diversos tipos de

características presentes no conjunto de entradas. A MLP é um tipo de rede muito utilizado devido à sua baixa complexidade e fácil desenvolvimento. O fluxo de informação, neste caso, é sempre no sentido da entrada para a saída da RNA, o que a classifica como uma rede do tipo "*feedforward"*. No processo de treinamento é utilizado o algoritmo "*backpropagation*" [3].

Este realiza duas etapas: uma chamada *"forward"*, e outra *"backward"*. Na primeira, um padrão qualquer é apresentado às unidades de entrada, processado nas camadas seguintes, e um valor de saída gerado.

Na segunda fase a saída produzida é comparada ao valor desejado e o erro médio quadrático (MSE), utilizado como medida de performance, é computado para este padrão. Em seguida os pesos nas unidades de saída são ajustados de acordo com o gradiente descendente do erro, em função do conjunto dos pesos da saída. O ajuste nas camadas intermediárias e de entrada são calculados da mesma maneira, sempre tomando como referência o erro na saída da rede que é retropropagado até a entrada, daí o motivo desta fase ser chamada de "backward".

Este processo é realizado iterativamente para cada novo padrão apresentado na entrada. Quando todos os padrões tiverem sido apresentados diz-se que uma "época" foi completada. Para se completar o treinamento de uma RNA pode ser necessário um grande número de épocas, isso dependerá do número e da complexidade dos padrões de entrada.

Apesar das facilidades de implementação do algoritmo pode-se dizer que nem sempre apresentará bons resultados. Um dos pontos mais relevantes se refere ao cálculo do gradiente local das funções de ativação para a minimização do erro. Em regiões de gradientes próximos de zero (platôs) ocorre a chamada "paralisia da rede", ficando o treinamento estacionado numa região de mínimo local.

Isto constitui uma convergência prematura e faz com a rede não responda de maneira correta. Muitos fatores influenciam no sucesso do treinamento de uma RNA, dentre eles os mais

importantes são: a topologia, a diversidade de padrões no conjunto de treinamento e o algoritmo de treinamento.

2.2 Lógica Fuzzy

A todo instante lidamos com informações imprecisas e inconsistentes. Afirmações como "o sistema está pouco instável", "as bolsa de valores teve uma pequena queda", ou "a violência está muito grande", são comuns no dia-a-dia, e quando surge a necessidade de quantificá-las é fácil perceber quão imprecisas elas são.

Isto se deve ao fato de estarmos habituados ao raciocínio lógico e exato, de forma que um dado valor ou característica pertença ou não a um determinado conjunto.

Da mesma forma, a representação lógica tradicional proposta por Boole [14] não possibilita a representação destas imprecisões ou inconsistências.

A lógica difusa (ou *Fuzzy Logic*), criada por L.A. Zadeh [1], propõe uma representação de forma que um determinado valor possa fazer parte de um único conjunto ou de vários, através de ponderações. Estes são os chamados conjuntos difusos e sugerem a representação de uma grandeza qualquer da seguinte forma:

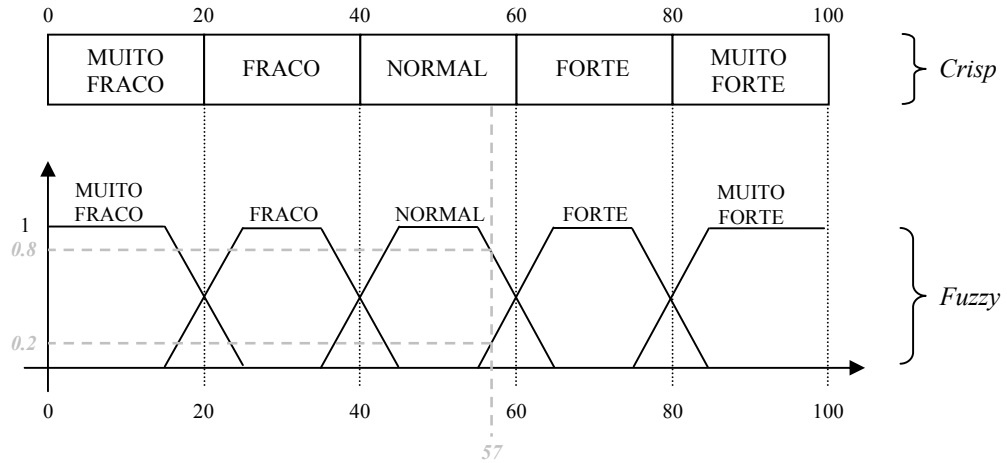


Figura 2.2.1: Representação CRISP x Representação Difusa

Nota-se, na representação anterior, que uma mesma grandeza pode ser representada de duas formas: a primeira através de conjuntos denominados "*crisp*", e a segunda através de funções de pertinência.

No primeiro caso, um determinado valor pertence ou não ao seu conjunto correspondente, conforme propõe a teoria de conjuntos booleanos. Nos conjuntos difusos correspondentes, pode-se perceber que um determinado valor pode pertencer a dois conjuntos distintos com "graus" diferentes.

Na representação *crisp*, o valor 57 é classificado como "normal", enquanto que para seu correspondente difuso, este será classificado como "normal" com grau de pertinência 0,8 e também "forte" com grau de pertinência 0,2. Esta forma de representação permite lidar com imprecisões comuns ao cotidiano.

No exemplo apresentado foram utilizadas funções do tipo trapezoidal, porém outros tipos podem ser utilizados, como triangulares, gaussianas, e outras.

O papel das funções de pertinência é o de criar um modelo classificação de variáveis numéricas em correspondentes lingüísticos. Esta é a primeira fase do projeto de um sistema baseado em lógica difusa. De posse dos termos lingüísticos, que classificam as variáveis de entrada de um sistema difuso, é possível a execução de um segundo processo denominado inferência.

Para o entendimento do processo de inferência é necessária a definição da base de regras. Esta é sem dúvida a parte mais importante de um sistema difuso e é nela que está armazenado todo o "conhecimento" incorporado ao sistema.

Para que se possa incorporar um tipo qualquer de conhecimento a um sistema, é necessária uma forma adequada de representação. A lógica difusa propõe um tipo de representação do conhecimento através de regras de produção.

Estas seguem a seguinte estrutura:

Se <PREMISSA> então <conclusão>

A primeira parte da regra, denominada premissa, contém fatos declarados através de operadores relacionais (=, >, <, ≥, ≤, ≠) e interligados através de operadores lógicos (“E”, “OU” e “NÃO”). Já a segunda parte, chamada de conseqüente, refere-se à conclusão proposta pela regra. Esta também pode apresentar expressões interligadas por operadores lógicos.

Quando se tem a base de regras formada, é necessário um mecanismo capaz de estabelecer a relação entre as regras de maneira a reunir o “conhecimento” necessário a uma determinada conclusão. Este é o papel do mecanismo de inferência: de acordo com o estado e correspondente classificação das variáveis de entrada, o mesmo determina quais regras serão necessárias à geração da conclusão e como estas estarão relacionadas.

À saída do mecanismo de inferência correspondem termos lingüísticos com seus respectivos graus de pertinência. Estes são necessários ao processo de defuzzificação que realiza as operações necessárias ao cálculo do valor numérico de saída do sistema.

A seguir é apresentada a representação de um sistema difuso com seus principais componentes:

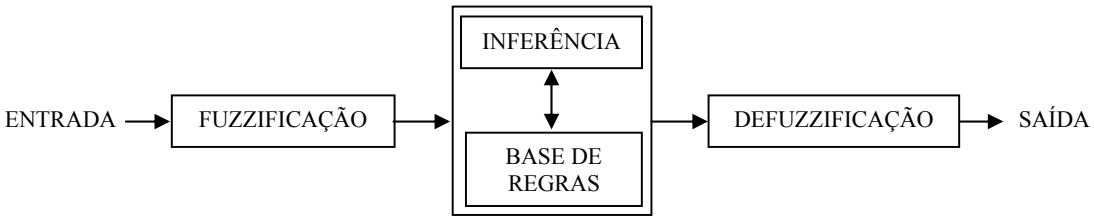


Figura 2.2.2: Componentes de um sistema baseado em lógica fuzzy

A seguir é apresentado um exemplo do processamento realizado por um sistema difuso:

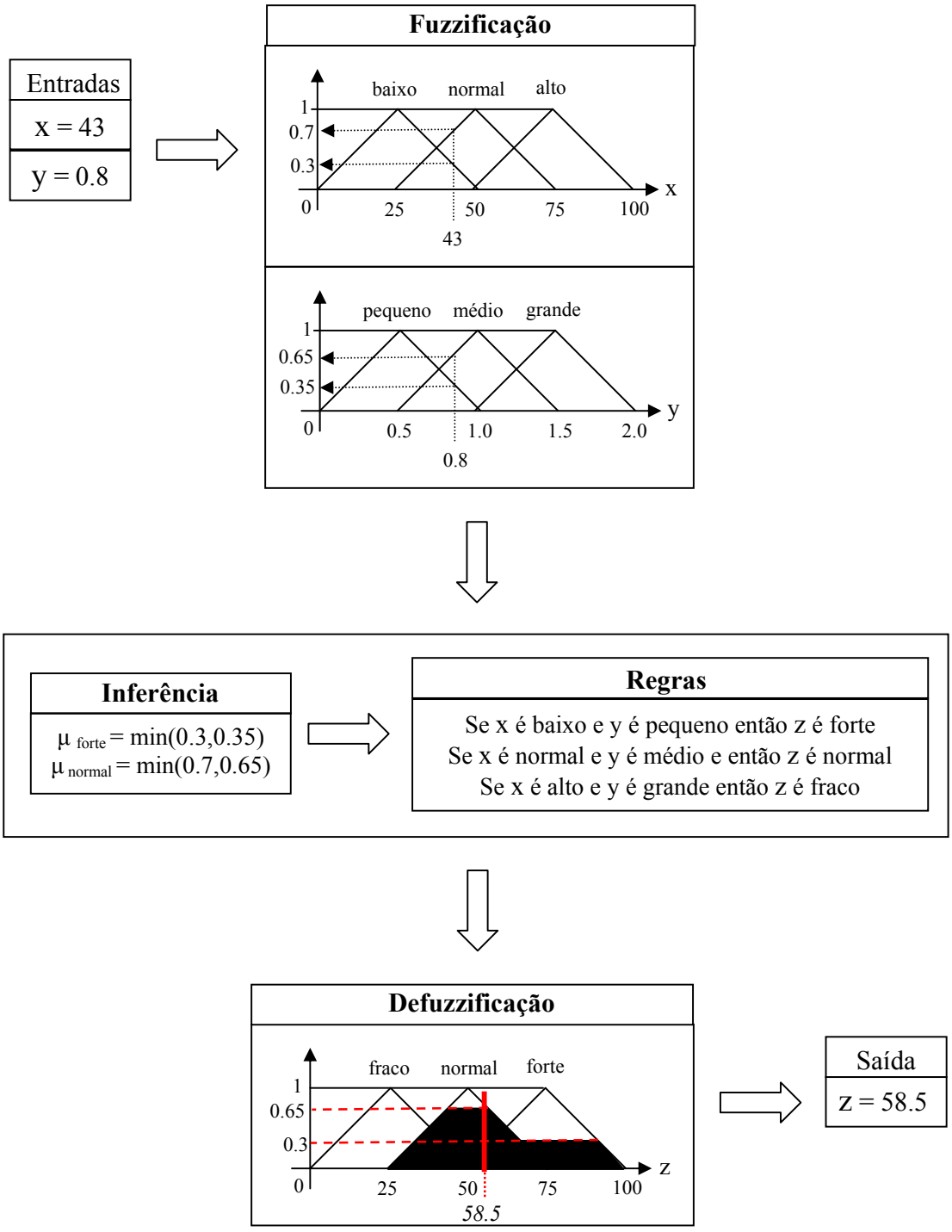


Figura 2.2..3: Exemplo de operação de um sistema fuzzy

Neste exemplo, o método utilizado para o processo de defuzzificação foi “centróide”, ou seja, valor de saída corresponde ao centro de gravidade (ou de massa) da região ativada no conseqüente.

Embora as representações anteriores apresentem apenas uma entrada e uma saída - SISO (*Single Input Single Output*), pode-se também ter sistemas difusos do tipo MISO (*Muliple Input Single Output*) ou MIMO (*Multiple Input Multiple Output*).

Em geral, pode-se dizer que um sistema difuso é responsável por criar um tipo de mapeamento das relações existente entre variáveis de entrada e saída do sistema.

A figura a seguir mostra a superfície correspondente ao mapeamento dos pares entradas-saída em função das regras estabelecidas.

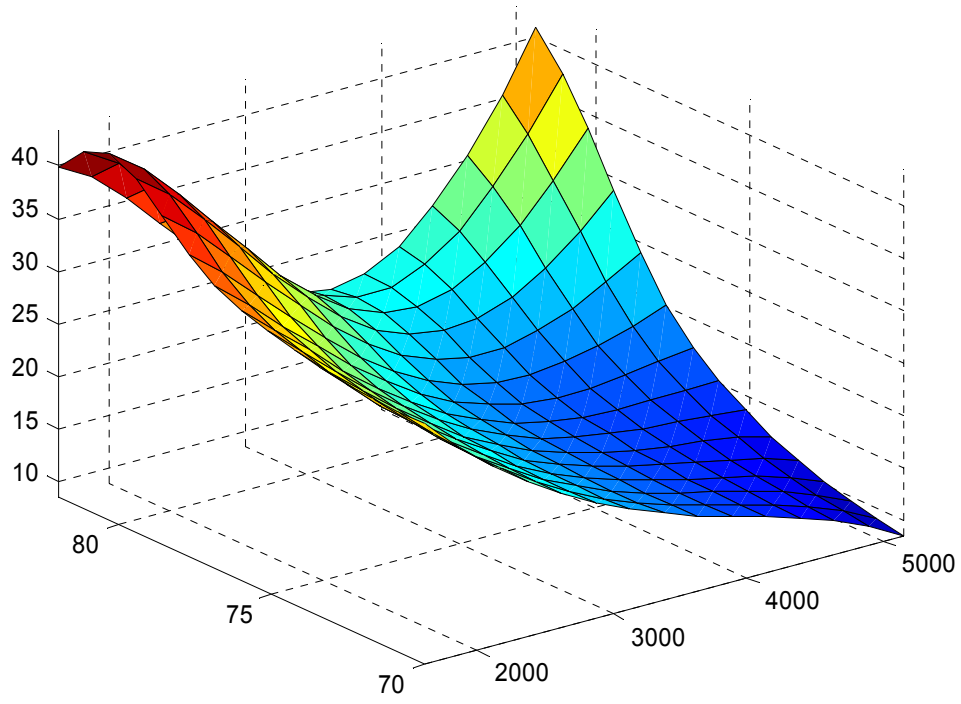


Figura 2.2.4: Superfície representativa das relações entre entradas e saídas do sistema fuzzy

A grande vantagem na aplicação dos sistemas difusos esta relacionada à facilidade no projeto, ao passo que os mesmos quando projetados pelos métodos convencionais se tornam extremamente complexos e de difícil análise. Outra vantagem é com relação à facilidade na manutenção da base de regras.

2.3 Redes Neurais Polinomiais

Há muito tempo, pesquisadores de diversas áreas têm realizado estudos no intuito de desenvolver modelos matemáticos capazes de representar o comportamento de complexos sistemas dinâmicos. A necessidade de conhecer e criar modelos, que representem de forma fiel as variáveis que regem os fenômenos do "mundo real" se faz presente constantemente, pois através destes são desenvolvidos sistemas de predição e controle. Porém, em muitos casos, isto se torna tarefa extremamente complexa devido ao comportamento não-linear e aleatório destas variáveis.

Diversas metodologias de modelagem de sistemas dinâmicos vêm sendo investigadas, gerando aproximações cada vez melhores.

Dentre aquelas em destaque, as redes neurais artificiais (RNA) têm apresentado resultados satisfatórios. Normalmente, estas são utilizadas em aplicações de reconhecimento (identificação) de determinados padrões presentes em uma variável, assim como em situações onde se deseja a predição de uma grandeza variante ao longo do tempo.

Assim, pesquisas nesta área fazem com que novos modelos surjam, cada um com suas aplicações e com características próprias.

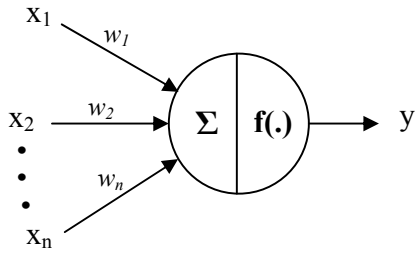
Um dos tipos de RNA mais utilizados em aplicações práticas é o *Multilayer Perceptron Network (MLP)*, ou rede de *perceptrons* multicamadas, que possui topologia fixa, é amplamente conectada e seu treinamento depende do ajuste de um grande número de parâmetros.

Sua grande desvantagem está na dificuldade em explicar o funcionamento de sua estrutura interna, o que a classifica como um modelo do tipo “caixa preta”. Outro inconveniente é o de não se poder definir, com precisão, o número de elementos de processamento ideal em sua camada interna, fazendo com que o processo de definição de sua arquitetura seja conduzido por tentativa e erro.

Dentre os diversos modelos de RNA, temos as redes neurais polinomiais (RNP), ou *Polynomial Neural Network*, que possuem um tipo de processamento, com determinadas particularidades, que possibilitam identificar características tanto lineares quanto não-lineares em um conjunto de dados.

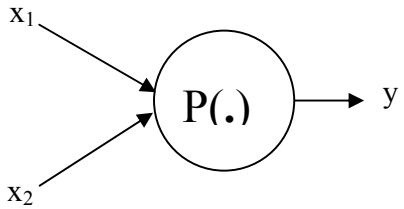
As RNP são modelos matemáticos baseados em um conjunto de funções, que quando têm seus parâmetros ajustados de acordo com um conjunto de dados, podem ser usadas para gerar uma representação generalizada destes, de forma a permitir uma predição dos valores nas suas vizinhanças, sem a necessidade de se incorporar nenhum novo tipo de informação à estrutura [7].

Este tipo de rede é capaz de fornecer aproximações mais precisas quando se comparadas a outros modelos como funções não-lineares, MLP e regressão multivariável não-linear, por exemplo. A seguir são apresentados os modelos de neurônios que compõe as redes citadas:



(a)

Figura 2.3.1a: Modelo de neurônio do tipo perceptron



(b)

Figura 2.3.1b: Modelo de neurônio polinomial

A figura 2.3.1a mostra um neurônio característico de uma rede do tipo MLP, onde a função de ativação não depende das entradas, ou seja, sua escolha é determinada pelo projetista e a mesma não tem a capacidade de "absorver", ou representar, características presentes no conjunto de dados. Toda informação referente ao "aprendizado" está contida no conjunto de pesos, representados pela letra "w".

Sua saída é da forma:

$$y = f(\Sigma(Xn * Wn))$$

Onde $X=[x_1, x_2, x_3 \ldots x_n]$ é o vetor coluna composto pelas *n* variáveis de entrada, $W=[w_1, w_2, w_3 \ldots w_n]$ é o conjunto com os pesos referentes à cada uma das entradas e *f*, normalmente, é uma função do tipo linear ou sigmoidal.

Na figura 2.3.1b está ilustrado um neurônio típico de uma rede neural polinomial.

Inicialmente é possível perceber que o mesmo possui apenas duas entradas e que não há pesos em suas conexões. A principal diferença se deve à função de ativação, que é determinada de acordo com os conjuntos de valores que suas entradas podem assumir. Isto é, a ativação deste tipo de neurônio é realizada por um polinômio, com grau pré-definido, cujos coeficientes são calculados de acordo com o conjunto de informações referentes àquelas entradas.

Como exemplo, podemos ter o seguinte:

$$y = A + B * x_1 + C * x_2 + D * x_1 * x_2$$

Onde *y* representa sua saída. *A*, *B, C* e *D* são seus coeficientes, e x_1, x_2 suas respectivas entradas.

Diferente dos modelos comumente aplicados, como a MLP, a RNP é auto-organizável, não possui ponderação entre suas conexões e seu processo de treinamento consiste em ajustar os

coeficientes dos polinômios e de eliminar conexões redundantes ou sem influência no valor da saída da rede.

A essência de seu projeto é baseada no *Group Method of Data Handling* (GMDH), proposto por Ivakhnenko [15] na década de 60**,** cujo intuito é realizar um tipo de mapeamento polinomial (linear, quadrático ou cúbico) entre um conjunto de variáveis de entrada e saída. Para isto são combinadas duas a duas, e um polinômio pré-determinado é utilizado para representá-las, gerando uma descrição parcial das mesmas. O uso de um polinômio é justificado pelo teorema de Kolmogorov-Gabor que mostra que qualquer função y=f(x) pode ser representada como:

$$y = w_0 + \sum_i w_i x_i + \sum_i \sum_j w_{ij} x_i x_j + \sum_i \sum_j \sum_k w_{ijk} x_i x_j x_k + \ldots ,$$

Onde X é o vetor que contém as variáveis de entrada e W é o vetor dos coeficientes.

O termo parcial é pertinente, uma vez que a mesma variável se combina, em pares, com todas as outras gerando um conjunto de representações candidatas. O uso de polinômios de diversas ordens faz com que seja possível o mapeamento de todos os tipos de características do conjunto.

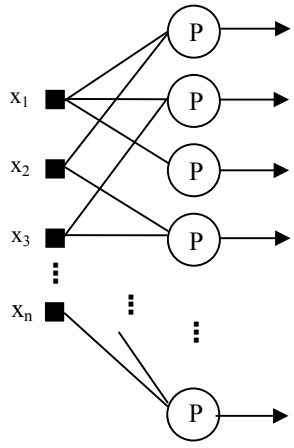


Figura 2.3.2: Topologia inicial de uma rede neural polinomial

A figura anterior mostra a topologia inicial de uma RNP. Os elementos de processamento, ou neurônios, são gerados através de todas as possíveis combinações entre pares de entradas. Desta forma pode-se dizer que a quantidade *N* de neurônios gerados em uma determinada camada *j* é igual a todas às *X* entradas, combinadas duas a duas:

$$\mathrm{Nj} = \binom{\mathrm{X}}{2} = \frac{\mathrm{X}!}{2!(\mathrm{X}-2)!}$$

Como cada elemento de processamento gera uma descrição "parcial" das entradas, através combinação em pares de todas as entradas, conforme citado acima, é importante notar que um pequeno aumento no número de entradas provoca um grande aumento na quantidade de neurônios. Por este motivo é necessário eliminar algumas destas unidades ao longo do processo de treinamento, caso contrário, a rede tomará proporções indesejáveis. Isto corresponde à segunda etapa do processo de treinamento. Outra característica que difere as RNP dos outros tipos de rede é a de que sua topologia é formada ao longo do processo de treino, o que a faz ser chamada de "rede plástica".

2.3.1 Processo de Treinamento

Quando se fala em "treinar" um modelo de rede neural artificial qualquer, deve-se pensar em ajustar um determinado número de parâmetros ou coeficientes para que a estrutura projetada produza, de acordo com os dados apresentados em sua entrada, uma saída de acordo com a desejada. Para que isto ocorra é extremamente importante um conjunto de valores que representem de forma correta o comportamento das variáveis envolvidas no processo. Este é chamado de conjunto de treinamento. Muitas vezes estes dados são provenientes de medições realizadas em campo e podem conter erros e até mesmo estarem ausentes devido às medições não

realizadas. Desta forma deve-se levar em conta que estes fatores influenciarão no processo de treinamento como um todo.

De posse do conjunto de treinamento, devem-se identificar as variáveis de entrada e combiná-las duas a duas. A seguir, determinar a ordem do polinômio que será responsável pela ativação de cada neurônio, e então estimar seus coeficientes. Para isto podem ser utilizados vários métodos, o mais utilizado é o Método dos Mínimos Quadrados [16].

Após a determinação dos coeficientes dos polinômios, pode-se calcular a saída de cada neurônio, sendo estas utilizadas como entradas para a camada seguinte.

Neste ponto é importante salientar que a combinação das entradas em pares irá gerar um grande número de elementos de processamento, e estes quando combinados para formar uma nova camada irão produzir uma quantidade ainda maior de neurônio. Sendo assim, para a formação de uma camada qualquer da RNP, são selecionados apenas aqueles cujas saídas apresentarem valores mais próximas aos desejados, e os outros descartados. Este processo se repete de forma que a estrutura seja definida ao longo do processo de treinamento.

A seguir são dadas informações detalhadas sobre o algoritmo utilizado para o treinamento.

2.3.2 Algoritmo de treinamento

O algoritmo de treinamento tem como função estabelecer a seqüência de ações necessárias à determinação de todos os parâmetros da RNP, assim como a definição de sua topologia.

Primeiramente, é sugerido que o conjunto de dados disponíveis, contendo os valores das observações, seja dividido em duas partes: uma para o processo de treino da rede e outra para a validação. O processo de validar o treinamento consiste em avaliar o comportamento da rede,

quando submetida a entradas não utilizadas anteriormente, ou seja, até então desconhecidos por ela. Sendo assim, tem-se uma disposição dos dados, conforme apresentado a seguir.

	Y	X (variáveis independentes)			
TREINAMENTO	Y_1	x_{11}	x_{12}	...	x_{1m}
	Y_2	x_{21}	x_{22}	...	x_{2m}
	.	.	.	...	.
	.	.	.	...	.
	.	.	.	...	.
	Y_{nt}	$x_{nt,1}$	$x_{nt,2}$	...	$x_{nt,m}$
VALIDAÇÃO	.	.	.	...	.
	.	.	.	...	.
	.	.	.	...	.
	Y_{nv}	x_n	x_{n2}	...	x_{nm}

O próximo passo consiste em combinar os pares de entradas que ativarão o neurônio correspondente. Para uma rede com quatro entradas tem-se a seguinte estrutura:

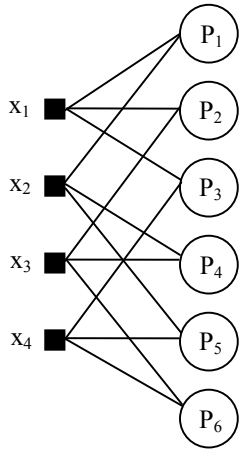


Figura 2.3.2.1: Combinação entre as entradas de uma RNP

O número de neurônios gerados é função do número de entradas, assim deve-se estimar cada um de seus polinômios.

Para o primeiro neurônio, P_1 é determinado através da resolução do sistema formado pela combinação das colunas contendo, respectivamente, os valores de x_1 e x_2:

$$z_1 = A + Bx_{11} + Cx_{12} + Dx_{11}^2 + Ex_{12}^2 + Fx_{11}x_{12}$$
$$z_2 = A + Bx_{21} + Cx_{22} + Dx_{21}^2 + Ex_{22}^2 + Fx_{21}x_{22}$$
$$z_3 = A + Bx_{31} + Cx_{32} + Dx_{31}^2 + Ex_{32}^2 + Fx_{31}x_{32}$$
$$z_4 = A + Bx_{41} + Cx_{42} + Dx_{41}^2 + Ex_{42}^2 + Fx_{41}x_{42}$$

Os valores dos coeficientes A, B, C, D, E, F podem ser calculados através do método dos mínimos quadrados, de forma que o neurônio seja ativado de acordo com um polinômio da seguinte forma:

$$\hat{z} = \hat{A} + \hat{B}x_1 + \hat{C}x_2 + \hat{D}x_1^2 + \hat{E}x_2^2 + \hat{F}x_1x_2$$

De forma idêntica são estimados os coeficientes dos polinômios de todos os neurônios gerados.

O passo seguinte consiste em avaliar suas saídas, de acordo com todo o conjunto de entradas. Para isto é adotado um critério externo qualquer como, por exemplo, o critério de regularidade, de acordo com [17]:

$$r_j^2 = \frac{\sum_{i=nt+1}^{n} (y_i - z_{ij})^2}{\sum_{i=nt+1}^{n} y_i^2} \qquad j = 1,2,...,\frac{m(m-1)}{2}$$

Onde *nt* é a quantidade de valores de *x,* utilizados para o treinamento, e *m* é o número de variáveis independentes (entradas); *y* corresponde ao valor da saída desejada da rede e *z* ao valor estimado.

De acordo com os valores obtidos, classificam-se como melhores, os neurônios que apresentarem menor valor pelo critério de regularidade, uma vez que estes são funções diretas do erro médio quadrático.

Posteriormente são selecionados os *k* melhores que permanecerão na estrutura. É importante frisar que o número *k* é atribuído pelo projetista, assim como o número de camadas que a rede irá possui em sua topologia final.

Escolhidos os neurônios da primeira camada, o processo se repete, porém agora, as entradas para a segunda camada são as saídas da camada anterior. A figura a seguir ilustra as fases deste processo.

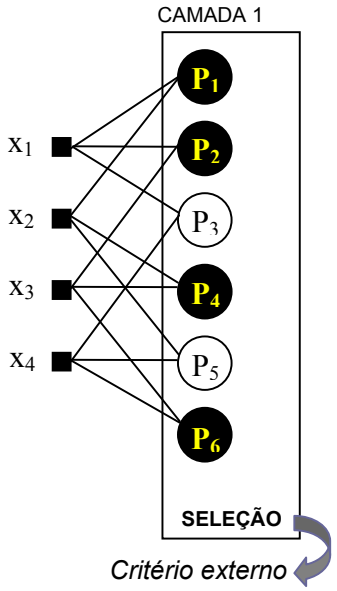


Figura 2.3.2.2a: Formação da primeira camada de uma RNP

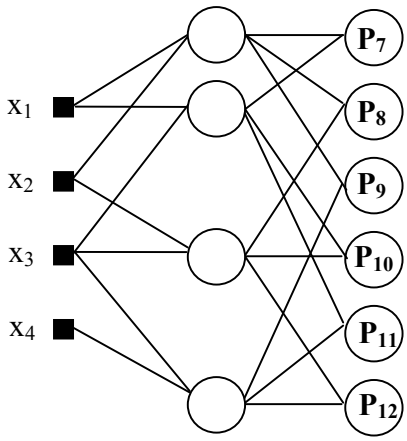


Figura 2.3.2.2b: Formação da segunda camada de uma RNP

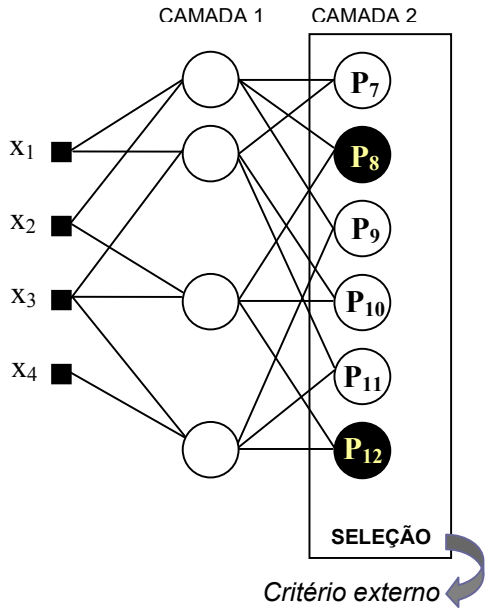


Figura 2.3.2.2c: Formação da terceira camada de uma RNP

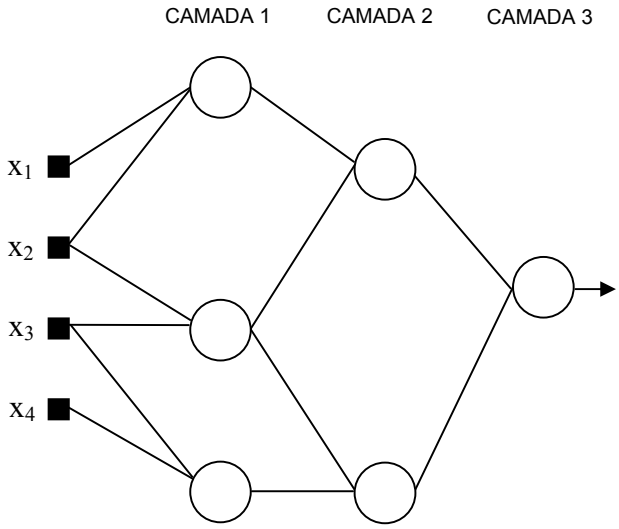


Figura 2.3.2.2d: Topologia final de uma RNP após o treinamento

Ao final tem-se uma estrutura, onde cada camada contém neurônios que são ativados pelos polinômios que melhor representam as características das variáveis das camadas anteriores. Estas podem ser expressas, de maneira recursiva, em termos das variáveis originais.

2.3.3 Considerações Importantes

Quando se opta pela aplicação de uma RNP, devem ser observados aspectos referentes à natureza do processo, a quantidade de variáveis envolvidas e suas características. Ainda assim, algumas outras importantes características têm que ser levadas em consideração. A seguir estão detalhadas as principais:

- *Número de variáveis de entrada*: conforme dito anteriormente, um aumento relativamente pequeno no número de neurônios pode elevar significativamente a quantidade de neurônios na camada posterior, tornando o projeto impraticável.

- *Grau dos polinômios:* a escolha do grau do polinômio é extremamente importante e pode influenciar de forma direta no desempenho da RNP. Normalmente se utiliza graus menores nas camadas mais próximas da entrada e graus maiores na camada de saída.

- *Quantidade de neurônios na camada intermediária:* a atribuição de uma quantidade suficiente de neurônios também é fundamental para o sucesso da implementação. Um erro na escolha irá afetar a capacidade de generalização da rede.

- *Número de camadas:* o número de camadas recomendado é de três ou quatro, sendo que quantidades maiores ou menores fazem com que a rede produza resultados com baixa qualidade.

2.4 ALGORITMOS GENÉTICOS

Os algoritmos genéticos [18] podem ser considerados como uma das técnicas mais aplicadas e importantes da computação evolucionária (CE). Foram propostos por John H. Holland em 1975 no trabalho denominado "Adaptation in Natural and Artificial Systems" [19].

Podem ser entendidos como ferramentas de otimização capazes de minimizar e maximizar funções, baseados na Teoria da Evolução do inglês Charles Darwin que combina genética e seleção natural.

Em outras palavras, os AG são modelos computacionais capazes de processar possíveis soluções para um determinado problema em estruturas semelhantes ao cromossomo biológico, além de aplicar a estas, operadores genéticos capazes de preservar e aproximar valores aleatórios de soluções em potencial.

2.4.1 Aspectos Práticos

O algoritmo genético é um processo de busca heurística aplicado em problemas onde se deseja encontrar soluções ótimas, que corresponde a limites máximos ou mínimos de uma determinada função, em um espaço de busca limitado. Esta é chamada de função objetivo, ou *função aptidão* segundo a terminologia dos AG.

Ao conjunto de possíveis soluções dá-se o nome de "população". Cada possível solução é denominada "indivíduo" ou "cromossomo", segundo a terminologia dos AG. Cada cromossomo é composto por um conjunto de valores, denominados "genes" e o valor que cada gene pode assumir é chamado "alelo".

O processo se inicia com a criação de população inicial, de forma aleatória. Esta é composta por número de indivíduos pré-determinado, que nada mais são do que conjuntos de números gerados aleatoriamente, considerados como soluções em potencial para o problema em questão. O passo seguinte consiste em avaliar a "aptidão" de cada um destes indivíduos em relação à função aptidão. A cada indivíduo avaliado é atribuído um valor de acordo com sua "proximidade" da solução do problema. Em outras palavras, quanto mais próximo da solução, maior a aptidão atribuída a este indivíduo e, conseqüentemente, maior a chance de suas características (genótipos) serem "herdadas" por seus descendentes nas futuras gerações.

Após avaliados, passam por um processo de seleção, onde os melhores tem maiores chances de permanecerem na população e se cruzarem entre si para formar a próxima geração. Os "filhos" resultantes podem ainda sofrer um processo de mutação genética.

Este processo é repetido iterativamente até que a solução seja encontrada ou então o número de gerações especificado seja atingido. O fluxograma a seguir ilustra o processo:

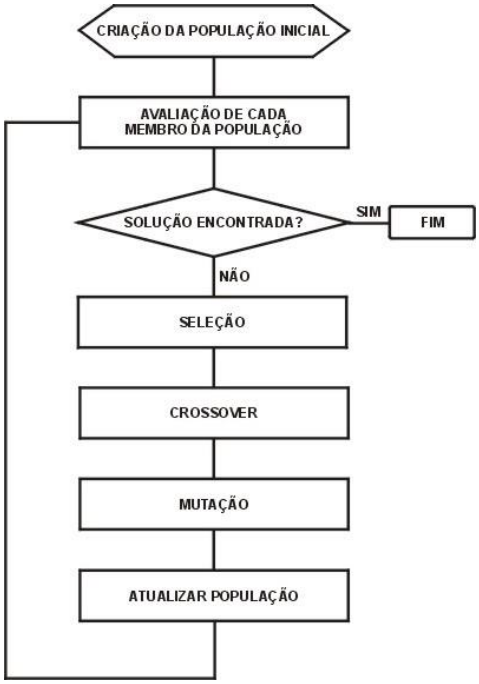


Figura 2.4.1.1: Fluxograma básico dos AGs [19]

2.4.2 População Inicial

Criar uma população inicial significa definir um conjunto de valores (indivíduos), de uma forma pré-definida, que serão candidatos à solução do problema em questão.

A forma de criação da população inicial é de grande importância para o sucesso do processo. Para isto é fundamental o conhecimento do espaço de busca ou região factível, que normalmente corresponde ao domínio do problema em questão.

Conhecendo-se estes limites máximos e mínimos que determinam o espaço de busca é possível então, gerar, segundo uma distribuição normal, uma quantidade de valores aleatórios igual ao número de indivíduos escolhido.

Outro ponto fundamental é definir como serão representados os valores gerados aleatoriamente. Na maioria dos casos cada um destes valores é representado por uma *string*

binária, correspondente ao valor original. Como exemplo, adotou-se um indivíduo com valor igual a 0,637197 e outro com valor igual a 10, conforme a tabela a seguir:

Indivíduo	*Valor*	*Codificação Binária:*	*Codificação Real:*
1	0,637197	'10001011101101010001 11'	0,637197
2	10	'00000000000000000001010'	10

Tabela 2.4.2.1: Exemplos de Codificações

No caso anterior a codificação foi binária com comprimento igual a vinte e dois. O valor do primeiro indivíduo em decimal é igual a 2288967, mapeado nos limites do espaço de busca escolhido [-1;2]. Este mapeamento foi feito de acordo com a seguinte expressão:

$$Indivíduo = [min+(max-min)*b_{10}] / (2^{L}-1)$$

Onde L é o comprimento do número binário codificado e b_{10} seu correspondente em decimal. Os valores "min" e "max" correspondem aos extremos do intervalo que determina o espaço de busca.

A escolha do alfabeto para a codificação da população inicial depende muito da aplicação da técnica, visto que na maioria dos casos é usado o alfabeto binário, porém o "real" também pode ser utilizado, desde que sejam aplicados operadores genéticos compatíveis.

O passo seguinte consiste em avaliar, de acordo com a função aptidão, cada indivíduo da população inicial e atribuir um valor de aptidão compatível com sua proximidade da solução ideal.

2.4.3 Função Aptidão

A função aptidão é de grande importância para o sucesso do algoritmo e permite avaliar indivíduos candidatos além de determinar sua permanência ou não, na população que irá compor a geração seguinte.

Na maioria dos casos, definir a função aptidão é uma tarefa simples, uma vez que ela será a própria função que se deseja otimizar. Após definida, cada indivíduo é avaliado, assim como a probabilidade de cada um deles permanecer nas gerações futuras. Isto é representado por um parâmetro chamado probabilidade de seleção, ou simplesmente *PS*.

Matematicamente, a probabilidade de seleção pode ser definida como:

$$PS_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$$

Depois de calculada, utiliza-se o método de seleção conhecido por "roda da roleta" para selecionar os indivíduos que irão compor a população intermediária. Este processo simula o papel da seleção natural na evolução, e faz com que sobrevivam e reproduzam, os indivíduos mais bem adaptados ao meio, no caso do AG aqueles com maiores valores para a função aptidão.

A população intermediária é criada para alocar os indivíduos selecionados pela roda da roleta, denominados pais e que serão submetidos aos operadores genéticos para gerar a nova população.

No método de seleção os indivíduos são dispostos como em uma roleta, de acordo com seu respectivo valor de probabilidade de seleção. A figura a seguir ilustra esta situação.

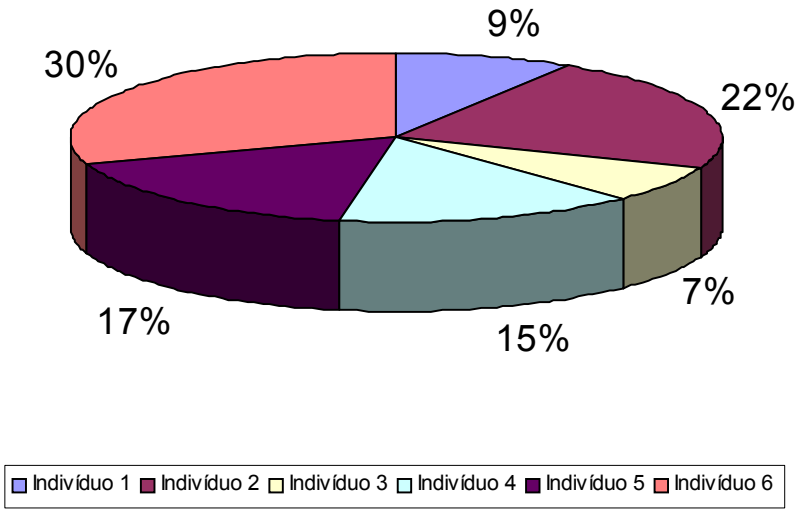


Figura 2.4.3.1 – Seleção pelo Método da Roleta

O processo de seleção simula o movimento real de uma roleta, de forma a selecionar aqueles que irão compor a próxima população.

Este processo é repetido até que o número de indivíduos na população intermediária seja suficiente para a criação da próxima geração, que terá sempre um tamanho fixo pré-determinado.

Em seguida são aplicados os operadores genéticos, descritos a seguir.

2.4.4 Operadores Genéticos

Quando dois indivíduos de mesma espécie se reproduzem existe a troca do material genético entre eles, o que garantirá a presença de características dos pais em seus descendentes. Este processo, dentro do AG, é chamado de "*crossover*" e garante que esta troca seja efetuada ao longo das gerações no processo de evolução.

Outro importante fenômeno presente no processo de reprodução é o aparecimento de características nos filhos que não estão presentes nos pais. Este fenômeno é chamado de mutação

e pode-se dizer que também é o responsável pela diversidade da população a cada processo de reprodução.

2.4.5 Crossover

A troca de material genético entre dois indivíduos se dá pelo processo conhecido por cruzamento (ou *crossover)*. Neste, o indivíduo gerado é formado por parte do material genético de ambos. Esta troca proporciona, na prática, a exploração pontos totalmente novos no espaço de busca. Este tipo de exploração é denominado *exploration*.

A figura a seguir ilustra a forma como acontece o tipo mais comum conhecido por "crossover de um ponto".

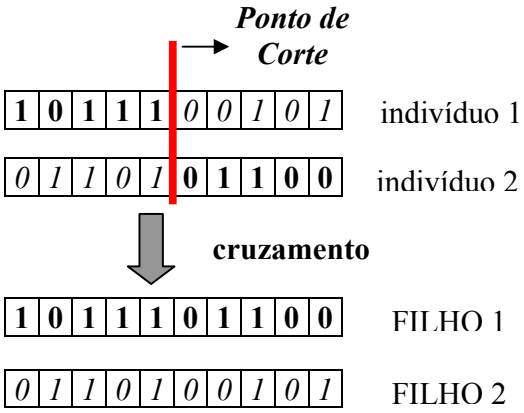


Figura 2.4.5.1: Exemplo de crossover para a codificação binária

Ponto de corte é o local, escolhido aleatoriamente, que limita a porção do material cedido por cada indivíduo.

Para a codificação real deve-se optar por um tipo de crossover próprio, uma vez que os métodos convencionais podem não adicionar nenhum novo tipo de informação.

2.4.6 Mutação

O processo conhecido por mutação consiste em acrescentar aos indivíduos gerados (filhos) características diferentes daquelas presentes em seus "pais". Isto garante um processo chamado *exploitation* que, na prática, é responsável por explorar o espaço de busca utilizando informações de pontos anteriormente visitados, a fim de encontrar melhores pontos.

A mutação acontece através da inserção de alguma nova característica ao indivíduo gerado, o que pode ser feito dentro do AG invertendo-se o valor de um bit, por exemplo.

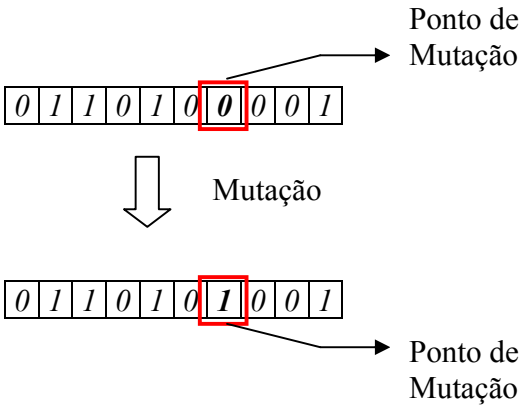


Figura 2.4.6.1: Exemplo de mutação para a codificação binária

O ponto de mutação determina o local, escolhido aleatoriamente, onde haverá a modificação do material genético.

2.4.7 Critérios de Parada

Após a composição da população intermediária e aplicação dos operadores genéticos têm-se os novos indivíduos (filhos) gerados que irão compor a nova população.

Esta nova população passa pelos mesmos processos de avaliação, seleção e reprodução. A cada nova população diz-se que passou uma geração. Este processo é repetido iterativamente até que a solução (indivíduo ótimo) seja encontrada ou então que o número de gerações pré-

estabelecido foi atingido. Normalmente são estes os critérios que determinam o fim da execução do AG.

2.4.8 Parâmetros Importantes

Para a execução do AG alguns parâmetros fundamentais devem ser especificados. Os mais importantes e aqui apresentados são: o número de indivíduos que irão compor a população inicial, a taxa de crossover, a taxa de mutação e o número máximo de gerações .

Estes parâmetros podem ser ajustados durante a execução do algoritmo, porém estes ajustes em tempo de execução podem comprometer o resultado final, desta forma optou-se por ajustá-los inicialmente e mantê-los fixos durante o processo.

A seguir serão explicados estes parâmetros assim como as modificações causadas por alterações em seus valores.

2.4.8.1 Número de Indivíduos

O número de indivíduos é um dos principais parâmetros e tem grande influência na execução do algoritmo.

Sua escolha é bastante particular, ou seja, em problemas onde o número de variáveis a serem otimizadas for pequeno não há grandes variações em termos de tempo de processamento, porém nos casos em que se deseja otimizar um grande número de variáveis, elevações no número de indivíduos podem implicar em aumentos consideráveis no tempo de processamento.

Desta forma a escolha deve ser feita experimentalmente, levando-se em conta que um número muito pequeno pode conduzir a uma convergência prematura assim como um número muito grande pode aumentar consideravelmente o tempo de execução do algoritmo.

2.4.8.2 Taxa de Crossover

O crossover é o mecanismo que permite a busca da melhor solução através da combinação de porções de cromossomos pais. No entanto este processo deve acontecer de forma controlada, fazendo com que não se perca alguma informação importante contida em alguns indivíduos com altas aptidões. Diversos valores são sugeridos, porém os mais comuns estão no intervalo entre 0.6 e 0.75.

2.4.8.3 Taxa de Mutação

Conforme dito anteriormente, o processo de mutação é responsável por adicionar ao cromossomo características novas, de forma que todas as regiões do espaço de busca possam ser exploradas. Porém, este deve ocorrer de forma controlada, para que não haja perda de material genético e comprometa a aptidão do cromossomo em questão. Assim como a taxa de crossover, diversos valores são sugeridos, sendo os mais comuns entre 0.001 e 0.05.

CAPÍTULO 3
SISTEMAS HÍBRIDOS

Neste capítulo serão apresentados os principais sistemas baseados na fusão das técnicas apresentadas anteriormente. A idéia hibridizá-las tem como objetivo aproveitar as melhores características de cada e, através da cooperação, a construção de poderosos sistemas aplicados na solução de problemas das mais diversas naturezas.

3.1 Redes Neuro-Fuzzy

Assim como todas as técnicas de inteligência artificial, as RNP possuem vantagens e desvantagens. Características como simplicidade, mapeamento de não-linearidades e estrutura compacta justificam a utilização desta em diversas aplicações. Porém, deve-se considerar que na solução de problemas de maior complexidade, a aplicação da RNP pode não ser suficiente, já que a mesma tende a gerar polinômios complexos para sistemas relativamente simples. Além disto, sua estrutura limitada (baseada em polinômios quadráticos com duas variáveis) tende a aumentar e se tornar complexa diante de sistemas com alto grau de não-linearidade, além de exigir uma grande quantidade de dados para sua definição. E mais, não é possível a criação de uma estrutura de RNP versátil em sistemas com três ou menos entradas [20].

Por estes e outros fatores, a necessidade de se criar novos modelos, que suprimam as deficiências dos atuais, sem abrir mão de seus pontos fortes.

Neste sentido diversas opções são propostas, dentre elas a associação de redes neurais artificiais com outras técnicas, de maneira a formar estruturas complementares, mais precisas e robustas.

Uma solução para minimizar os problemas relativos às RNP, apresentados acima, é através da associação com sistemas fuzzy.

Para que isto seja possível, é necessário criar uma nova unidade de processamento capaz de incorporar características dos sistemas difusos. Quando arranjados de forma correta, assumem não só a estrutura de uma rede neural como seu poder de processamento.

Esta pode ser considerada aceitável, pois tem como base os sistemas neuro-fuzzy tradicionais. Sendo assim, são apresentados a seguir, os principais modelos neuro-fuzzy, assim como sua evolução e conceitos necessários à compreensão das redes neuro-fuzzy polinomiais (RNFP).

3.2 Conceitos Fundamentais

O modelo híbrido neuro-fuzzy que obteve maior expressão foi o proposto por Jang [2], denominado Sistema de Inferência Fuzzy baseado em Rede Adaptativa, ou simplesmente ANFIS (Adaptive-Network-Based Fuzzy Inference System). Em seu trabalho, Jang propôs um sistema híbrido capaz de emular um sistema fuzzy, porém com uma estrutura semelhante à das redes neurais artificiais. Após o processo de treinamento é capaz de fornecer um tipo de mapeamento entrada/saída através de regras do tipo se-então, conforme os sistemas fuzzy. Sua arquitetura, similar à das redes neurais, é apresentada a seguir:

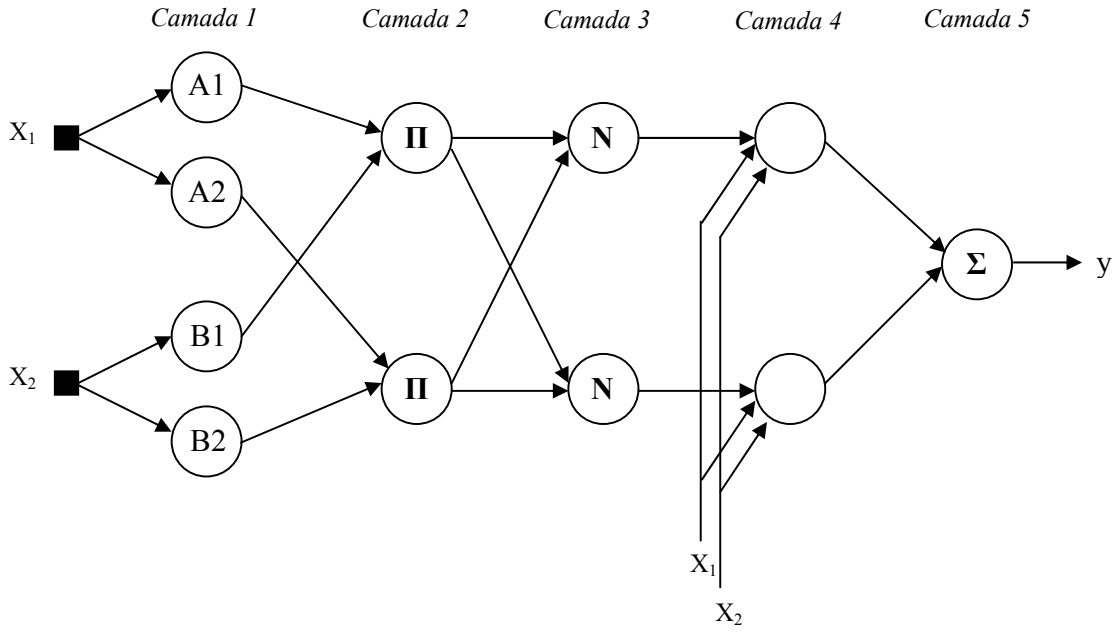


Figura 3.2.1: Topologia básica de uma rede neuro-fuzzy[2]

Onde cada neurônio na camada dois representa uma regra do sistema fuzzy que pode ser escrita da seguinte forma: "Se $x_1=A_1$ e x_2=B1 então $Y=ax_1+bx_2+c$".

É importante notar, no modelo ilustrado, que a topologia é muito semelhante à das RNA do tipo *feedfoward*. A diferença mais importante está no fato desta possuir uma topologia fixa onde o papel de cada neurônio é conhecido, ao contrário das RNA do tipo MLP, cuja estrutura é de difícil definição e não se pode explicar o funcionamento de suas camadas internas.

Segundo a proposta da rede ANFIS, a camada um contém as funções de pertinência do sistema fuzzy, normalmente do tipo triangular ou gaussiana, o que corresponde à parte da premissa da regra. O papel desta é o de agrupar os dados de entrada em subgrupos (ou subespaços fuzzy) e associar cada um ao correspondente subgrupo, de acordo com suas características. Este processo é chamado de "agrupamento" e permite que espaço representativo de entrada seja dividido através de um tipo de particionamento de forma que cada partição contenha dados com características semelhantes.

A figura a seguir exibe um particionamento do tipo grade (do inglês *grid*) sobre um conjunto de entrada qualquer, através de funções de pertinência do tipo gaussiana.

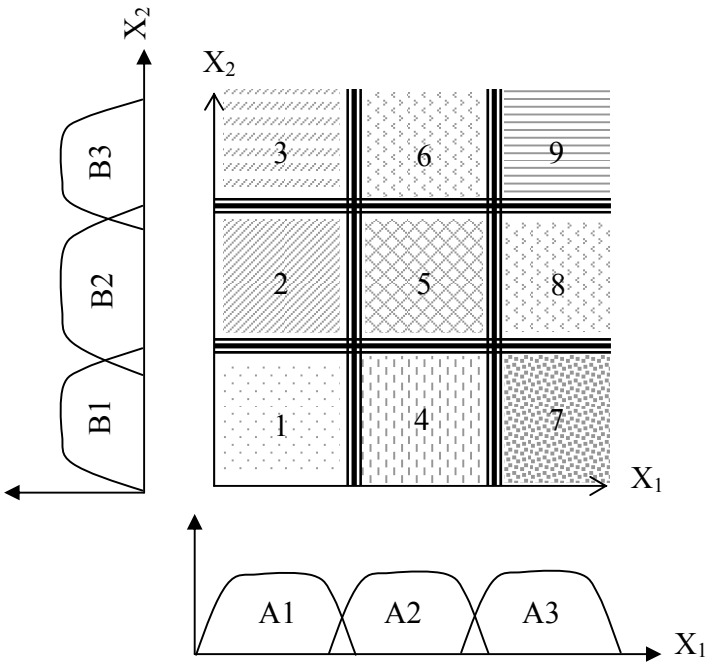


Figura 3.2.2: Exemplo de particionamento realizado por conjuntos fuzzy

A camada dois é responsável por aplicar um tipo de norma triangular, normalmente uma operação lógica "e" entre suas entradas. A camada três tem como função normalizar os dados de suas entradas. Sua saída corresponde ao grau de ativação da parte do conseqüente representado pela camada quatro. Nesta camada, a ativação dos neurônios é realizada por um polinômio, que representa uma combinação linear entre as entradas do sistema. Esta é a parte correspondente ao conseqüente da regra fuzzy, proposto por Sugeno [19], também conhecido como conseqüente de Sugeno. Este se refere ao processo de defuzificação ilustrado a seguir:

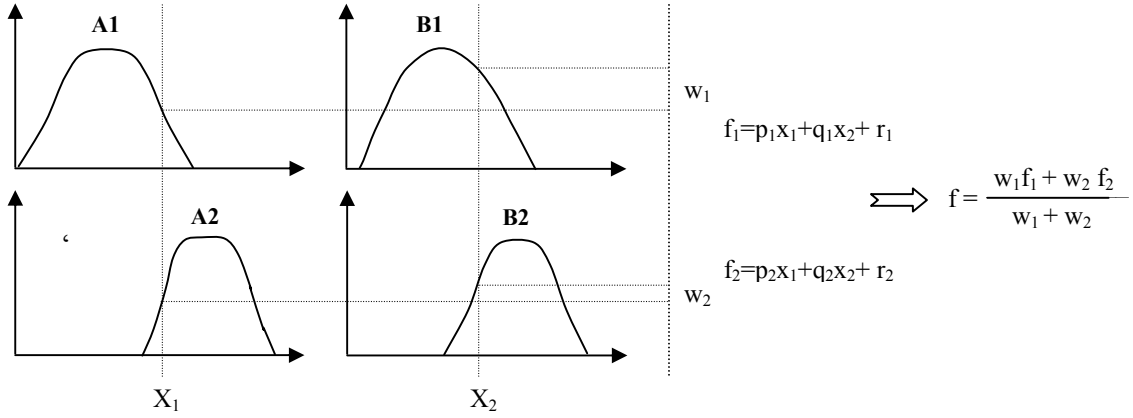


Figura 3.2.3: Método de defuzzificação

Pela figura é possível notar que a saída desta camada é calculada pela soma dos produtos do valor médio dos graus de pertinência pelo valor do polinômio, referente às entradas do sistema.

A camada cinco é responsável por realizar o somatório de todas as saídas da camada quatro e gerar a saída da rede.

Para que a estrutura proposta responda de maneira adequada é necessário que seja ajustado todo o conjunto de parâmetros livres da RNF, como largura e posição das funções de pertinência, além dos parâmetros correspondentes aos polinômios do conseqüente da regra fuzzy. Outras definições como o tipo e o número de funções de pertinência, além da ordem do polinômio correspondente ao conseqüente, ficam a cargo do projetista. Muitas vezes estas são definidas por tentativa e erro, o que pode comprometer todo o desempenho.

O ajuste dos parâmetros da rede ANFIS é realizado em duas etapas. A primeira pelo algoritmo *backpropagation,* típico das RNA, que utiliza o gradiente descendente para definir parâmetros como largura e posição das funções de pertinência e a segunda pelo método dos mínimos quadrados, que permite estimar os coeficientes dos polinômios da camada quatro.

Após o treinamento, a estrutura será capaz de fornecer o conjunto regras, do tipo "*Se...Então*", capaz de modelar os dados de entrada de acordo com a saída desejada, assim como nos sistemas fuzzy tradicionais.

Embora esta proposta tenha sido aplicada com êxito em diversas situações, é importante salientar que muitas vezes a decisão com relação aos tipos de função de pertinência e, principalmente quanto à sua quantidade, não é tarefa fácil.

Além disto, deve-se levar em conta, que sistemas que possuem muitas entradas podem gerar topologias complexas e extremamente grandes, principalmente se for necessário o uso de mais de duas funções de pertinência.

Anos depois, Sung-Kwun [23] propôs um novo modelo, baseado na rede ANFIS de Jang. A proposta de Sung consiste na utilização de uma estrutura muito semelhante à ANFIS para cada variável de entrada, conforme ilustrado a seguir. Ao final, suas saídas são somadas para gerar a "resposta" da rede.

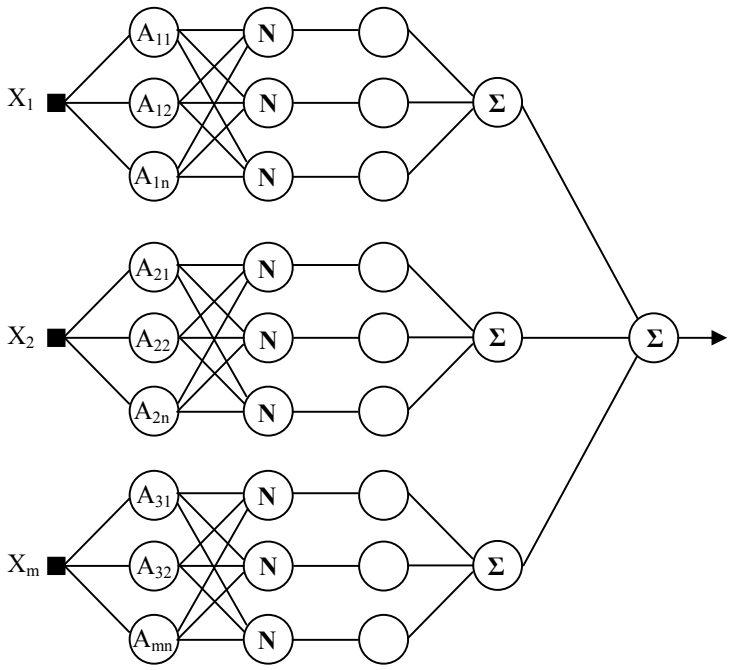


Figura 3.2.4: Modelo de rede neuro-fuzzy [23]

O diferencial deste modelo, aqui chamado apenas de rede neuro-fuzzy (RNF), está associado à capacidade de realizar o "particionamento fuzzy" do espaço entrada/saída, de forma independente para cada variável de entrada da rede.

No processo de treinamento é utilizado, além de métodos baseados em gradiente descendente, o algoritmo genético (AG). Sua capacidade de utilizar informações históricas (anteriormente calculadas) para gerar novas aproximações, é promovida por mecanismos capazes de explorar toda a região no espaço de busca, fazendo com que seja possível lidar com funções-objetivo complexas e não diferenciáveis.

Sendo assim, é aplicado na otimização dos parâmetros relativos ás funções de pertinência.

As aplicações práticas mostram que a utilização de AG favorece a geração de resultados satisfatórios.

A grande desvantagem desta proposta é com relação ao crescimento da estrutura diante de um aumento no número de entradas. Outro ponto fraco é com relação ao número e tipo das funções de pertinência, visto que se utilizam dos mesmos princípios da rede ANFIS. Isto faz com que haja o compartilhamento de suas vantagens e, de forma inerente, das suas dificuldades.

Mais adiante, em [24] Sung introduz o conceito de rede neuro-fuzzy auto-organizável, baseado na fusão entre as RNF e RNP. Esta passa a ser uma tentativa de utilizar os benefícios das RNF, com cada variável de entrada representada por apenas duas funções de pertinência, com a vantagem de auto-organização das RNP.

Neste, o papel da RNF é o de interagir com os dados de entrada, criando grânulos a eles correspondentes. Ou seja, a conversão dos pontos de dados numéricos, em representações abstratas, de acordo com os conjuntos fuzzy. Estas entradas podem ser tratadas separadamente ou em conjunto. Os grânulos gerados são decorrentes do processo de particionamento, explicado anteriormente.

Na estrutura proposta a RNF corresponde à parte da premissa, de forma que a parte do conseqüente, formado pela RNP, visualize apenas os grânulos gerados. Nesta etapa estes são aproximados do ideal, fazendo com que a RNP funcione como um tipo de "ajuste fino".

Para isto, Sung considera dois tipos de arquitetura, apresentadas pelas figuras abaixo e responsáveis por constituir a premissa das regras fuzzy que serão geradas.

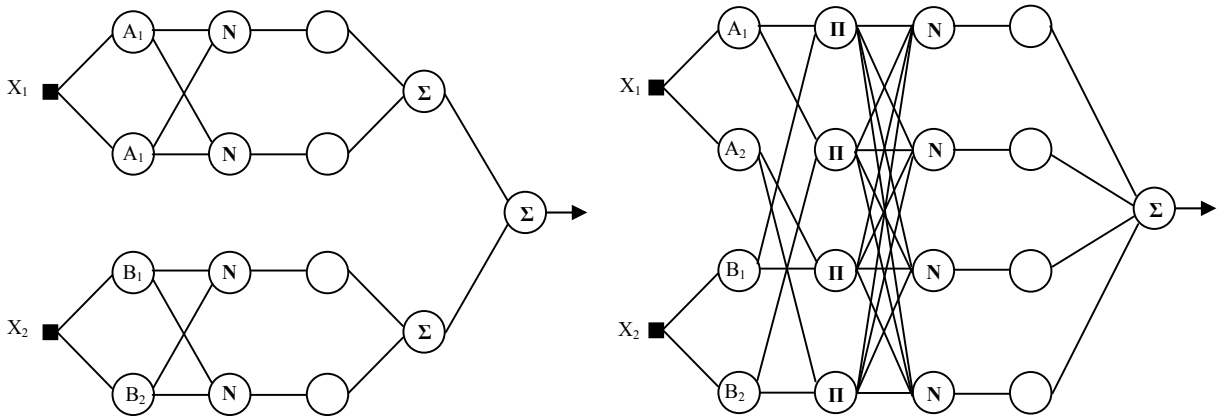


Figura 3.2.5: Diferentes tipos de topologias de rede neuro-fuzzy [24]

Nos dois casos as entradas são mapeadas por duas funções de pertinência: A e B. No primeiro caso o particionamento é realizado de forma independente e a saída corresponde à soma das parcelas de cada um desses grânulos. Neste caso os níveis de ativação da parte dos conseqüentes são determinados apenas pela respectiva entrada, de forma que uma não interfira na ativação da outra. Sendo assim, pode-se ativar apenas a regra correspondente a uma entrada, sem levar a outra em consideração.

Esta estratégia torna a rede simples e de fácil interpretação, mas pode não ser totalmente adequada, pois os conseqüentes das regras serão sempre funções de uma única variável. Isto, na prática, não é suficiente para a representação de comportamentos altamente não-lineares.

No segundo caso, a premissa das regras geradas, leva em consideração a influência das duas entradas. Estas se combinam através de um tipo de norma triangular e ativam, de forma normalizada, os conseqüentes das regras. Estes, por sua vez, correspondem à combinação linear das entradas. Esta estrutura se diferencia da primeira quando considera as relações fuzzy, através das operações de norma realizadas entre as variáveis de entrada e, consequentemente, ao tipo de agrupamento (particionamento) realizado sobre o conjunto de entradas.

As duas metodologias são avaliadas e a elas incorporadas a RNP, responsável por gerar a parte dos conseqüentes das regras. A idéia de incorporá-las é justificada por suas vantagens de auto-organização aliada à capacidade de expansão do espaço de representação das variáveis de entrada, através de descrições parciais de suas entradas, provocadas pelos polinômios em cada neurônio.

Desta forma, tem-se um novo modelo, baseado nos sistemas neuro-fuzzy (RNF) agregados à rede neural polinomial (RNP), denominado rede neuro-fuzzy-polinomial (RNFP).

3.3 Redes Neuro-Fuzzy-Polinomiais

Baseadas nas topologias de redes neuro-fuzzy apresentadas, são concebidos os primeiros modelos de redes neuro-fuzzy-polinomiais. As mesmas são formadas pela adição de uma porção de rede polinomial às estruturas neuro-fuzzy padrão, conforme ilustra a figura seguinte:

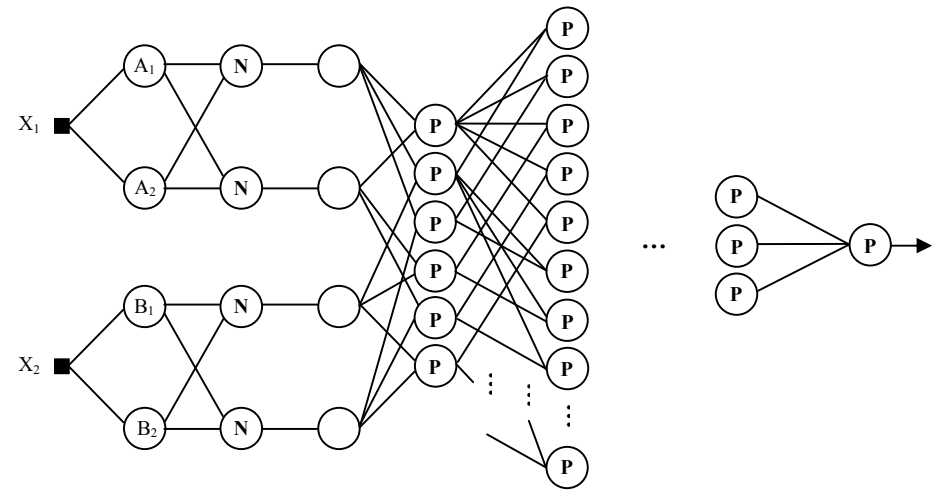


Figura 3.3.1: Exemplo de topologia de uma rede neuro-fuzzy-polinomial [24]

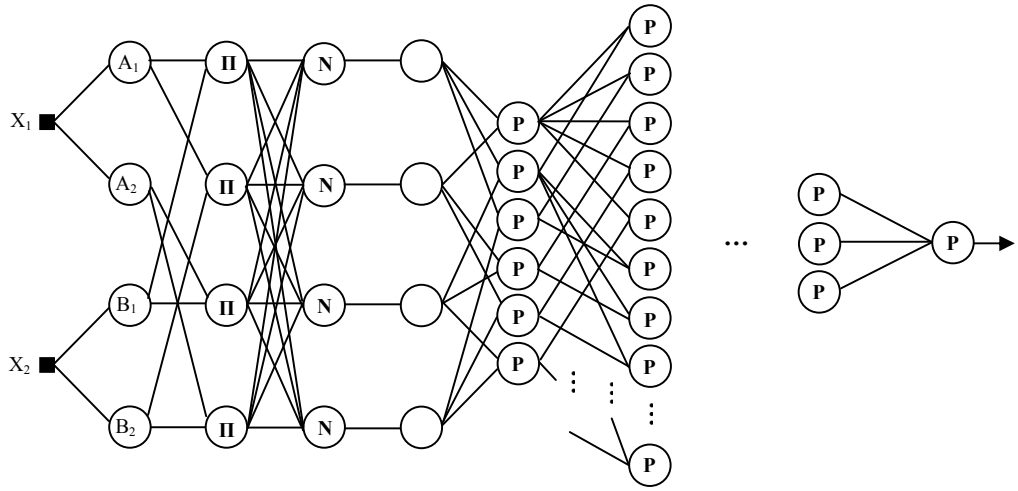


Figura 3.3.2: Exemplo de topologia de uma rede neuro-fuzzy-polinomial [24]

De acordo com as topologias apresentadas, pode-se notar que a estrutura básica da RNF é preservada, e a ela agregada uma RNP, porém agora, a primeira representa a parte das premissas do conjunto de regras do sistema fuzzy, enquanto que a segunda exerce a função de uma RNP, como discutido anteriormente.

Assim, deve-se considerar cada estrutura de RNF como um neurônio da RNFP, capaz de incorporar as características dos sistemas difusos. Este, chamado de neurônio fuzzy-polinomial (NFP), é constituído por dois módulos funcionais. O primeiro contém uma série de conjuntos

difusos, que servem de interface entre as variáveis numéricas de entrada e a parte de processamento realizada pelo neurônio. O segundo formado por polinômios que podem realizar o processamento não-linear. O uso de polinômios é justificado por sua capacidade de generalização. Em particular, também incluem mapeamentos constantes e lineares como casos especiais, que são freqüentes em sistemas baseados em regras.

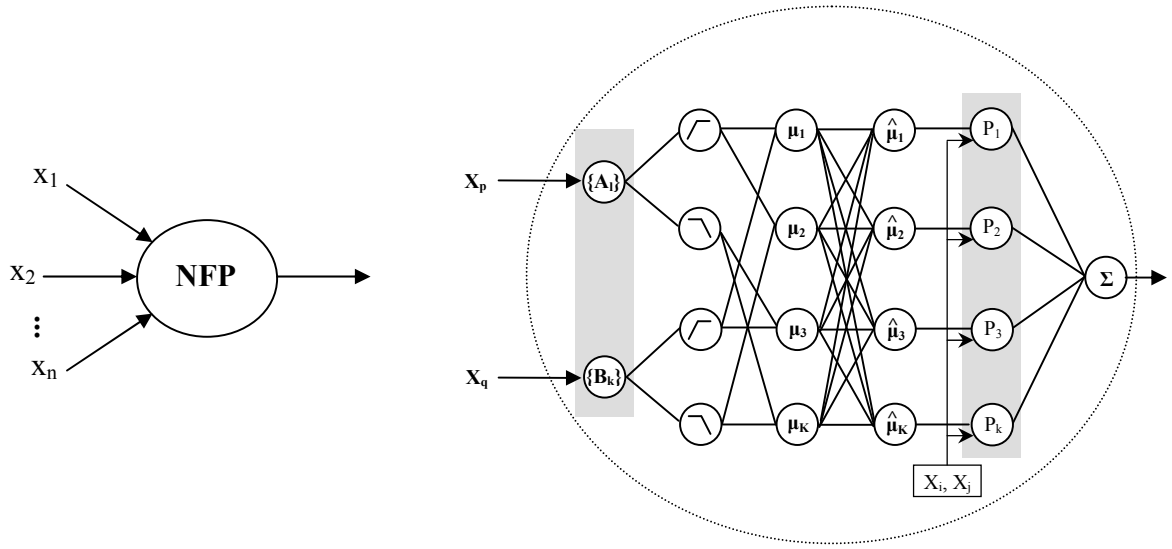


Figura 3.3.3: Arquitetura interna de um neurônio da RNFP [25]

Pela figura anterior nota-se que o número de entradas pode variar, sendo estas transformadas em conjuntos difusos. Estes conjuntos podem ser iguais, diferentes ou parcialmente sobrepostos com variáveis que são processadas pelo módulo polinomial do neurônio. Esta forma de tratamento das entradas adiciona flexibilidade extra a toda unidade de processamento. De posse dos conjuntos difusos usados para transformar as variáveis de entrada, é possível obter regras referentes ao sistema com múltiplas entradas e saída única (MISO).

A função de todos os elementos internos do NFP é semelhante às dos elementos da estrutura ANFIS, apresentada anteriormente. Na primeira camada, as entradas numéricas são "fuzzificadas". A camada seguinte realiza uma operação de norma triangular entre os graus de

pertinências das entradas em relação às suas respectivas funções de pertinência. Na camada posterior o resultado da operação é normalizado, com relação às saídas dos neurônios da camada anterior.

O grau de pertinência normalizado corresponde ao “nível de ativação do conseqüente da regra”, ou seja, a “força” com que o polinômio será “disparado”.

Finalmente, os valores produzidos pelos polinômios são somados. O resultado desta soma corresponde à saída do NFP.

Neste ponto é válido ressaltar que o polinômio é o grande responsável por mapear características de não-linearidades presentes na entrada. Para que isto seja possível é extremamente importante que a escolha de seu grau seja correta. Além disto, devem ser ajustados outros parâmetros como o número e os tipos de funções de pertinências utilizados, o método para a determinação dos coeficientes dos polinômios, e se serão utilizadas todas as entradas da rede ou somente àquelas referentes ao neurônio.

A figura seguinte mostra uma estrutura genérica de RNFP, de acordo com a metodologia em questão.

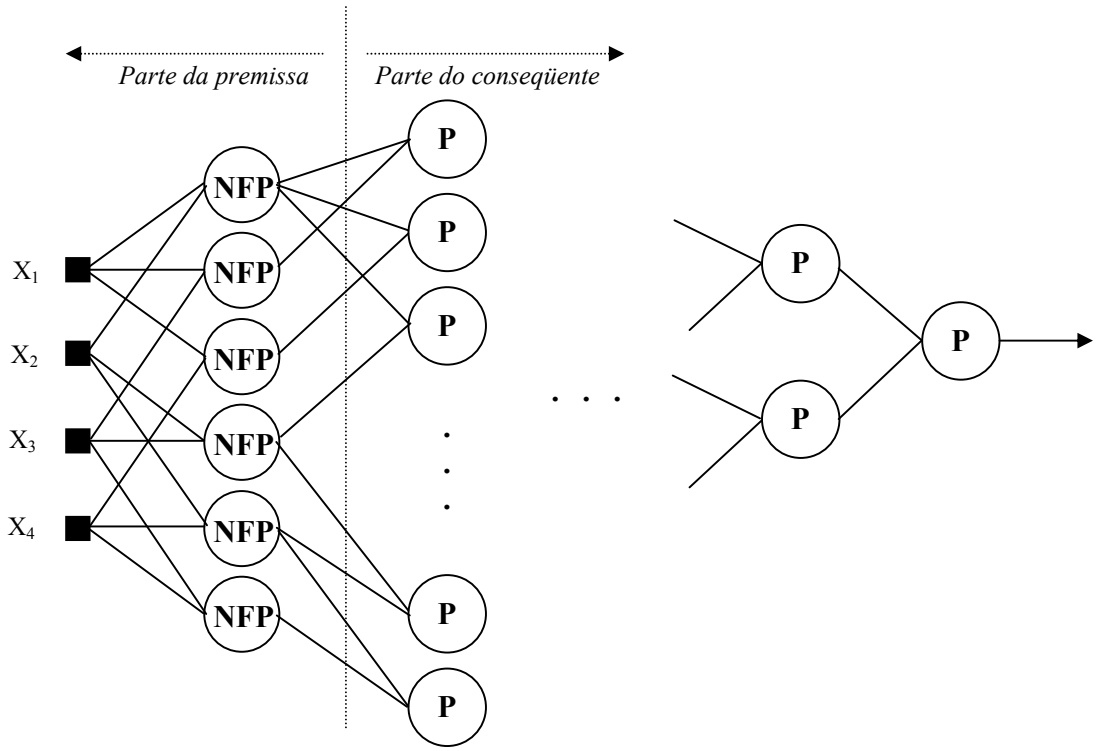


Figura 3.3.4: Regras x topologia, na rede neuro-fuzzy-polinomial

Esta pode ser dividida em duas partes, uma responsável pelas premissas e outra pelos conseqüentes das regras que serão produzidas. Para que a toda esta estrutura adquira a capacidade de "responder" de forma desejada, é necessário que todos os seus parâmetros sejam ajustados.

O processo de se estimar o conjunto ótimo de valores dos parâmetros é chamado de treinamento e será apresentado a seguir.

3.4 Processo de Treinamento

Como sabemos, as RNA são modelos matemáticos inspirados na fisiologia dos sistemas cerebrais, que emulam sua forma de operação.

Para que isto seja possível, algumas características como a capacidade de armazenamento de informações e "raciocínio", são fundamentais para que as consideremos como sistemas inteligentes.

Assim, devemos ter algum tipo de conhecimento para posteriormente a validar e verificar se houve algum tipo de reação próxima da esperada. Desta forma, as mesmas são submetidas ao processo de treinamento.

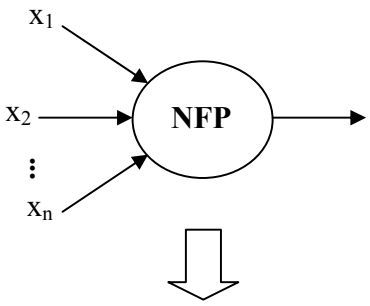
O processo de treinamento, conforme dito anteriormente, é uma das fases mais importante do projeto de RNA e corresponde ao ajuste de todos os parâmetros livres da rede, de forma que esta possa produzir respostas muito próxima ou idêntica àquelas desejadas.

Além desta condição, outras decisões referentes à topologia são também fundamentais, e muitas vezes são definidas por tentativa e erro.

No caso da RNFP isto não é diferente, ou seja, mesmo sendo uma estrutura híbrida, possui um grande conjunto de parâmetros que devem ser ajustados.

Inicialmente, para cada NFP deve-se definir: o tipo e número de funções de pertinência para cada variável de entrada, a quantidade de entradas, o método de inferência e o grau do polinômio. É importante ressaltar que o número de regras que o sistema irá produzir está diretamente associado ao número de NFP.

Em seguida devem ser determinados os graus dos polinômios em cada neurônio polinomial, referente à porção de RNP da RNFP.



Nº de Funções de Pertinência	*Tipo das Funções de Pertinência*	
2	Triangular	
3	Gaussiana	
Estrutura do conseqüente	***Método de inferência***	***Ordem do polinômio***
Todas as entradas	Constante	1
Somente entradas selecionadas	Regressão polinomial	2,3,...

Figura 3.4.1: Parâmetros de um neurônio fuzzy-polinomial

A figura acima apresenta os principais parâmetros relativos aos NFP, que compõe a parte da premissa das regras fuzzy, que o sistema irá gerar. Além disto, alguns outros parâmetros relativos à parte do conseqüente devem ser determinados. Como esta é composta por uma RNP, são necessários os valores para os coeficientes de todos os polinômios de ativação dos neurônios, além da definição do grau de cada um deles.

Assim, torna-se necessário um conjunto de valores ótimos para todos os parâmetros RNFP, que garantirá a geração das saídas desejadas.

A tarefa de encontrar este conjunto ótimo é um problema típico de otimização, onde o objetivo é o de minimizar o erro na saída da rede.

Normalmente, estes valores são atribuídos pelo próprio projetista, que os ajusta de acordo com sua experiência. Aplicações mais recentes têm utilizado métodos de busca heurística, como algoritmos genéticos (AG), para encontrar a solução ótima para o problema em questão.

É muito importante lembrar que a RNFP é uma estrutura auto-organizável, ou seja, tem sua topologia definida ao longo do processo de treinamento. Sendo assim, após este, deve-se ter uma estrutura, como por exemplo, a figura seguinte:

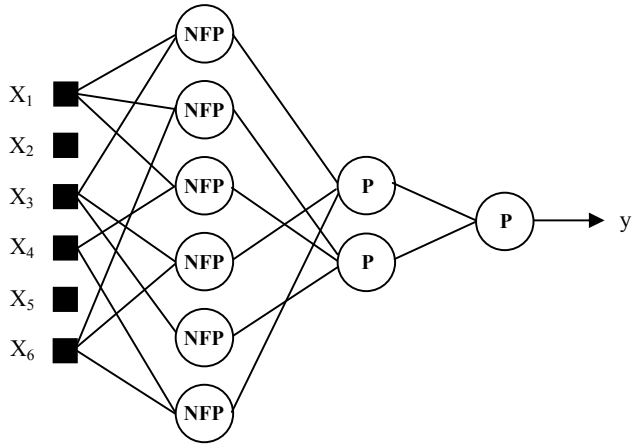


Figura 3.4.2: Topologia de uma RNFP após o treinamento

Onde os NFP correspondem à parte da premissa e os PD à parte do conseqüente das regras fuzzy geradas.

Nota-se, pela figura, que algumas entradas não estão sendo utilizadas. Estas, certamente foram desconsideradas devido à irrelevância em relação às saídas desejadas. Outro aspecto importante relaciona-se ao número de entradas nos neurônios, que foi determinado ao longo do processo de treinamento, realizado, por exemplo, por um AG.

Diversas aplicações envolvendo RNFP associadas ao AG têm sido desenvolvidas e variantes destas propostas com intuito de se obter resultados cada vez melhores.

Em [26] é apresentado um modelo auto-organizável de RNFP, onde a otimização da estrutura é realizada pelo AG e os parâmetros determinados pelo método dos mínimos quadrados padrão.

Uma metodologia de RNFP com neurônios polinomiais baseados em conjuntos fuzzy é proposta em [27]. Nesta o autor propõe que a granulação dos dados de entrada na parte da premissa e estes grânulos fuzzy representados por polinômios, capazes de representar características locais presentes nos padrões de entrada.

Em [28] é apresentado um sistema de previsão, para problemas em sistemas de comunicações móveis, baseado em RNFP. Neste os parâmetros relativos à premissa são definidos pelo projetista e o processo de treinamento é dedicado à estimação dos coeficientes dos polinômios.

Em [29, 30, 31, 32] são apresentadas metodologias semelhantes com aplicações práticas baseadas em diversos conjuntos de dados experimentais.

Na maioria das aplicações citadas utiliza-se algoritmos genéticos para a otimização de toda a estrutura. Em todos os casos o processo de treinamento é *offline*, ou seja, utiliza um conjunto de treinamento fixo e depois de treinada é colocada em operação.

A investigação sobre estes sistemas é apresentada a seguir.

3.5 Redes Neuro-Fuzzy-Polinomiais otimizadas via Algoritmos Genéticos

A evolução das técnicas matemáticas de modelagem tem caminhado no sentido de permitir a criação de representações cada vez melhores do comportamento de complexos sistemas, que operam em função de um grande número de variáveis. Assim, é necessário otimizar o conjunto de parâmetros (ou coeficientes) que estão associados ao modelo representativo, de maneira que este seja capaz de mapear o comportamento altamente não-linear presente na maioria dos sistemas . No entanto, quando o número de variáveis aumenta de forma significativa,

surgem diversos problemas de convergência, principalmente quando são usados métodos baseados em informações de gradiente.

Nestes tipos de aplicações, envolvendo grande número de variáveis, os Algoritmos Genéticos (AG) tem se destacado por sua simplicidade, adaptabilidade, e pela qualidade dos resultados apresentados. Além disto, não utiliza nenhum tipo de informação proveniente de gradiente, o que em muitos casos conduz a convergência a pontos de mínimos locais e trabalham com um conjunto de possíveis soluções candidatas em paralelo que tendem a evoluir através de mecanismos de seleção, cruzamento e mutação.

Por estes motivos os mesmos são, inicialmente, indicados para o ajuste da grande quantidade de parâmetros que envolvem o projeto de uma RNFP.

É importante lembrar que, para o sucesso do AG, alguns parâmetros também devem ser corretamente escolhidos, como: número de indivíduos da população inicial, número de gerações, taxa de cruzamento e mutação, alfabeto de codificação e outros. Por este motivo, é pertinente avaliar os resultados produzidos pelos AGs, quando utilizados no treinamento de RNFP.

3.5.1 Aspectos Práticos

Conforme mostrado anteriormente o projeto de uma RNFP, assim como seu treinamento, envolve uma série de definições que dependem exclusivamente da experiência do projetista. Questões como a topologia, tipos de funções de pertinência, métodos de inferência e muitas outras, determinarão o desempenho da estrutura final. Por este motivo a opção de associar um método capaz de auxiliar na determinação de valores ótimos para as definições de projeto e ajuste é de extrema importância.

As vantagens dos algoritmos genéticos fazem com que este seja o método de solução ideal para o problema em questão. Sendo assim é necessário definir quais parâmetros da RNFP serão

determinados pelo AG. Feito isso, é criado um modelo de indivíduo (ou cromossomo) que contenha genes correspondentes a todos os parâmetros envolvidos.

Após a formação da população inicial, os mecanismos de seleção, cruzamento e mutação irão garantir que a evolução conduza o processo para indivíduos que contenham os valores ótimos desejados. A figura seguinte apresenta os principais parâmetros da RNFP.

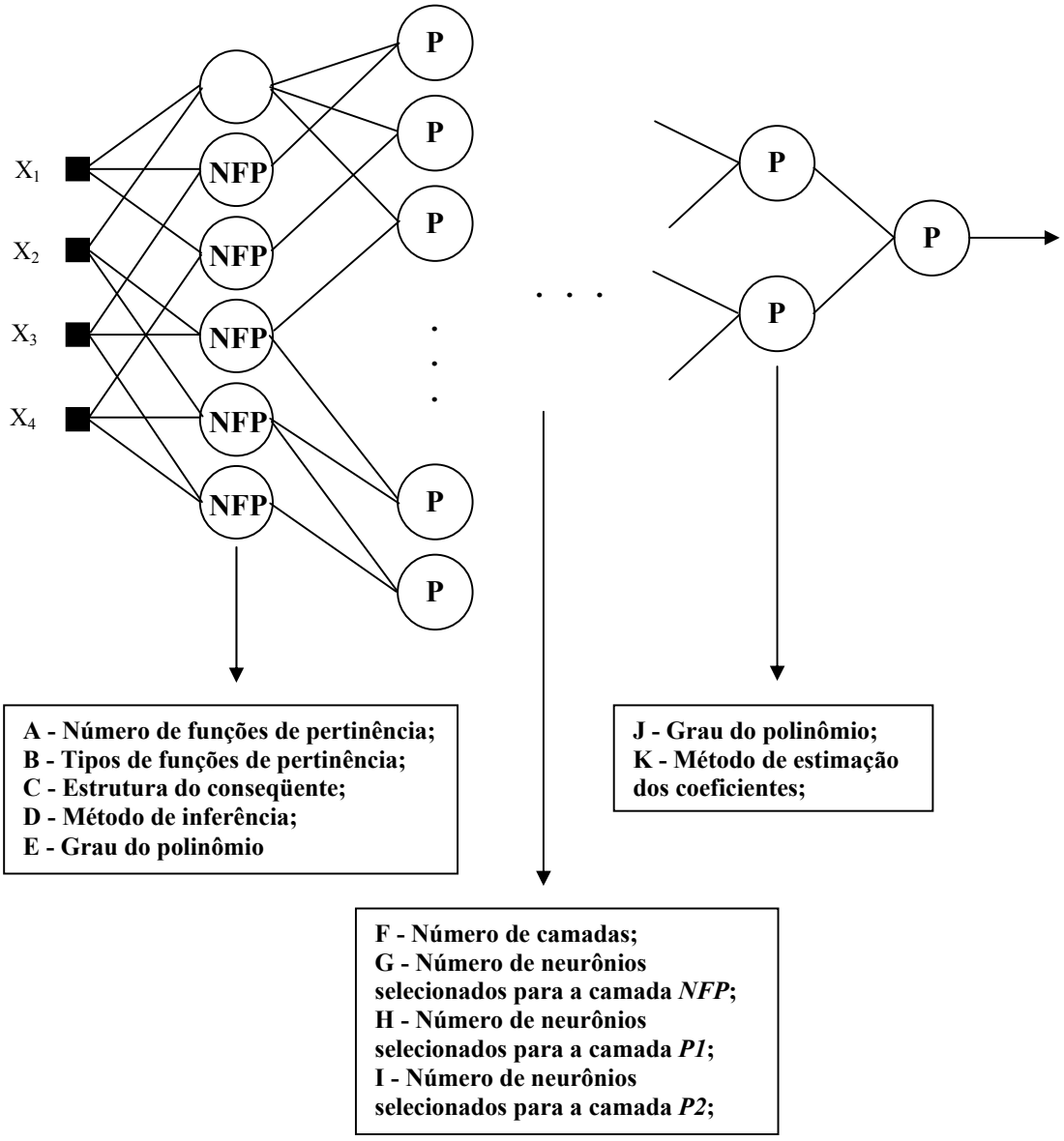


Figura 3.5.1.1: Parâmetros de uma RNFP

De acordo com a estrutura anterior é possível criar um modelo de indivíduo que conterá informações referentes a todos os parâmetros envolvidos. Inicialmente deve-se pensar em buscar um conjunto ótimo contendo valores inteiros. Para isto uma forma simples e capaz de representar as informações necessárias é através da codificação binário. O modelo de indivíduo gerado terá a seguinte forma:

A	B	B	C	D	E	E	E	F	F	G	G	H	H	I	I

Neste temos as letras representando os parâmetros com um determinado número de bits:

A – número de partições do conjunto de entrada para o NFP;

BB – tipo de função de pertinência de entrada do NFP;

C – número de entradas do neurônio NFP;

D – tipo de conseqüente do NFP (constante ou linear);

EEE – número de NFP selecionados na primeira camada;

FF – grau dos polinômios dos neurônios polinomiais P da primeira camada intermediária;

GG - número de neurônios polinomiais P selecionados para a primeira camada intermediária;

HH – grau do polinômio dos neurônios polinomiais P da segunda camada intermediária;

II - grau do polinômio do neurônio polinomial P da camada de saída;

As possibilidades de indivíduos estão representadas a seguir:

Bit	*Valor*	*Representação correspondente*
A	0	Duas funções de pertinência
	1	Três funções de pertinência
BB	00	Triangular
	01	Trapezoidal
	10	Gaussiana Simples
	11	Gaussiana Composta
C	0	Somente entradas selecionadas
	1	Todas as entradas
D	0	Constante
	1	Linear
EEE	000	Quatro neurônios selecionados
	001	Cinco neurônios selecionados
	010	Seis neurônios selecionados
	011	Sete neurônios selecionados
	100	Oito neurônios selecionados
	101	Nove neurônios selecionados
	110	Dez neurônios selecionados
	111	Onze neurônios selecionados
FF	00	Polinômio de grau 1
	01	Polinômio de grau 2
	10	Polinômio de grau 3
	11	Polinômio de grau 4
GG	00	4 neurônios selecionados
	01	5 neurônios selecionados
	10	6 neurônios selecionados
	11	7 neurônios selecionados
HH	00	Polinômio de grau 1
	01	Polinômio de grau 2
	10	Polinômio de grau 3
	11	Polinômio de grau 4
II	00	Polinômio de grau 1
	01	Polinômio de grau 2
	10	Polinômio de grau 3
	11	Polinômio de grau 4

Pelo tipo de indivíduo proposto é fácil perceber que a estrutura da RNFP será formada por quatro camadas (uma com NFP e outras três por neurônios polinomiais - P) e que todos os neurônios em uma dada camada terão características iguais, como o número de partições e tipo de função de pertinência (no caso dos NFP) assim como os graus dos polinômios para os neurônios tipo P. Outra característica implícita é que a camada segunda intermediária é formada por dois neurônios e a camada de saída por apenas um.

Para que os ajustes dos parâmetros da RNFP sejam feitos de forma independente é necessário um indivíduo que contenha todas estas informações ou que o processo seja executado de forma independente para cada elemento. Nos dois casos teremos um inconveniente: o elevado tempo de processamento. Sabe-se que uma das desvantagens da aplicação de algoritmos genéticos está relacionada ao tempo de processamento e à grande ocupação de recursos de hardware. Se o tipo de problema exige a otimização de um número muito grande de variáveis, isto faz com que o AG deixe de ser atraente. Outro fato relevante é que nem sempre o AG produz os melhores resultados, visto que não é possível determinar, inicialmente, os valores ideais para o número de gerações nem para as taxas de cruzamento e mutação, que influenciam diretamente no processo de evolução.

Aplicações que tem o acréscimo constante de novas informações exigem que haja um treinamento periódico da RNFP. Isto pode fazer com que o tempo necessário à convergência do AG torne esta metodologia inviável.

CAPÍTULO 4
PREVISÕES EM SÉRIES TEMPORAIS

Este capítulo tem por objetivo apresentar as informações necessárias à compreensão dos métodos utilizados na previsão de valores futuros em séries de tempo. Tais conceitos são fundamentais para o entendimento dos testes práticos realizados

4.1 Introdução

Há muito tempo os sistemas de previsão estão presentes em nossas vidas, e mesmo sem percebermos o utilizamos com grande freqüência. Um exemplo bastante comum são os sistemas de previsão meteorológica. Constantemente utilizamos suas informações para planejarmos uma viagem ou mesmo um passeio.

A idéia de planejamento está fortemente associada a situações indesejadas que podem vir a acontecer e se estas são antevistas podemos nos preparar para contorná-las. Tal fato é comum nas diversas áreas do conhecimento e por isto se tornou um atraente objeto de pesquisas.

Na área de engenharia elétrica, em específico nos sistemas de geração, transmissão, distribuição e comercialização de energia elétrica, a situação é a mesma. A questão da previsão de cargas elétricas, por exemplo, é um dos pontos mais importantes quando se trata de planejamento e operação de sistemas elétricos de potência.

A carga elétrica total de um determinado sistema é composta pela associação de todos os dispositivos ou aparelhos elétricos a este conectados. As mesmas apresentam em seu comportamento diversas tendências que podem ser estatisticamente previstas, porém, a enorme influência de fatores externos (econômicos, climáticos, temporais, demográficos e esporádicos) faz com que seja extremamente difícil estimar com exatidão um valor futuro.

Outras importantes aplicações que podem ser citadas são: fluxo de potência, análise de estabilidade, planejamento e operação de redes de distribuição, planejamento de manutenção, tarifas e preços, etc.

Para simplificar o entendimento deve-se classificar a questão do planejamento como um problema de otimização, onde o objetivo é o de minimizar as perdas.

Com o mercado cada vez mais competitivo, aquele que consegue o menor nível de perdas certamente se tornará cada vez mais forte. Por estes motivos são abordados os principais conceitos sobre previsão em séries temporais e posteriormente serão apresentados os resultados dos testes realizados com as técnicas anteriormente discutidas para casos de previsão em longo prazo.

4.2 Principais Conceitos

Uma série temporal, por definição, consiste de um conjunto de observações *y(k)* realizadas em um determinado período de tempo, $k=t_1, t_2,..., t_n$, onde $t \in \Re$, $k \in \mathrm{N}$, que podem ser decompostas em diversas componentes como: constante, tendência linear, variação cíclica, impulso, degrau e rampa. Sendo assim, diversos modelos podem ser usados na representação de uma série temporal. No caso de uma série aleatória, pode-se adotar o seguinte:

$$y(k) = b + \varepsilon(k) = b * f(k) + \varepsilon(k)$$

onde b é a média (neste caso constante), ε(k) é uma componente aleatória que representa o ruído inerente ao sistema e f(k)=1.

Num segundo caso podemos considerar que a série apresenta comportamento de tendência linear. Assim, podemos representá-la como:

$$y(k) = b_1 * f_1(k) + b_2 * f_2(k) + \varepsilon(k)$$

Onde $f_1(k)=1$, $f_2(k)=k$ e $\varepsilon(k)$ é o ruído.

Generalizando:

$$y(k) = b_1 * f_1(k) + b_2 * f_2(k) + ... + b_m * f_m(k) + \varepsilon(k)$$

Onde $B=b_1,b_2,...,b_m$ é o conjunto de parâmetros, $F=f_1,f_2,...f_m$ as funções de *K* e *m* o número de funções.

Uma representação alternativa é baseada em valores passados da série, chamados "auto-regressores". Neste caso a representação é feita da seguinte forma, segundo o caso linear:

$$f_i(k) = y(k-i)$$

Na forma generalizada:

$$y(k) = b_1 * y(k-1) + b_2 * y(k-2) + ... + b_m * y(k-m) + b_{q+1}\varepsilon(k)$$

Onde b_{q+1} é um valor constante.

Em geral, pode-se dizer que existe um tipo de correlação entre observações vizinhas e o estudo de uma série temporal busca a criação de um modelo que represente esta dependência. Quando se consegue criar um modelo capaz de representar de forma fiel o comportamento de uma determinada variável se torna possível a predição de um valor futuro.

No entanto, quando lidamos com variáveis como cargas elétricas, fica difícil estabelecer um modelo matemático representativo devido à grande quantidade de fatores que de forma direta influenciam seu comportamento.

A carga elétrica total de um determinado sistema elétrico é composta pela associação de todos os dispositivos ou aparelhos elétricos a este conectados. As mesmas apresentam em seu comportamento diversas tendências que podem ser estatisticamente previstas, mas como dito

anteriormente, fatores externos influenciam constantemente de forma direta. Os mais comuns são:

- Fatores climáticos: são fatores que ocorrem normalmente em um período curto de tempo e que podem impactar instantaneamente ou após algum tempo. Como exemplos podem-se considerar: chuvas, temperatura, umidade relativa do ar, etc. Destes, o mais importante é a temperatura, que sob variações bruscas pode forçar o acionamento de sistemas de refrigeração ou aquecimento.

- Fatores temporais: são situações ocasionadas pela ocorrência de eventos cíclicos e sazonais. Nestes observa-se a variação da carga em determinados períodos do ano como férias, feriados, início do ano escolar, etc. Nos períodos cíclicos são observadas reduções significantes na carga como nos finais de semana, por exemplo.

- Fatores econômicos e demográficos: são fatores que ocorrem por longos períodos de tempo e causam impacto considerável em termos de carga elétrica. Estes estão associados a situações de aumento do parque industrial em uma determinada região, às tendências econômicas, variações bruscas no valor das tarifas de energia, etc.

- Fatores esporádicos: são fatores que influenciam a carga elétrica por um pequeno período de tempo. É o caso de eventos esportivos, eleições, greve de industriários, etc.

Os diversos fatores externos que influenciam na carga elétrica total de um determinado sistema, e que podem provocar determinados comportamentos, possibilitam classificá-las de duas formas: estacionária e não-estacionária.

Uma série temporal é dita estacionária se seus valores oscilam em torno de uma média constante ao longo do tempo. No caso de uma série temporal não-estacionária, seus valores sofrem variações no tempo de forma que estes apresentem padrões tendenciosos, cíclicos, sazonais ou aleatórios.

4.3 Previsões em séries temporais

A previsão de um valor futuro em uma série temporal depende fundamentalmente de duas condições:

- que as informações sobre valores passados possam ser numericamente quantificadas;

- que o comportamento passado irá, de certa forma, se repetir.

As principais classificações dos tipos de previsão estão associadas ao período de predição desejado. Este pode ser chamado de "horizonte de previsão" e estabelece a que tempo futuro o valor previsto se refere. Outro conceito importante é o que define o número de amostras de valores passados que serão utilizadas no processo. A este chamamos de "janela de tempo". A figura seguinte ilustra estes dois conceitos:

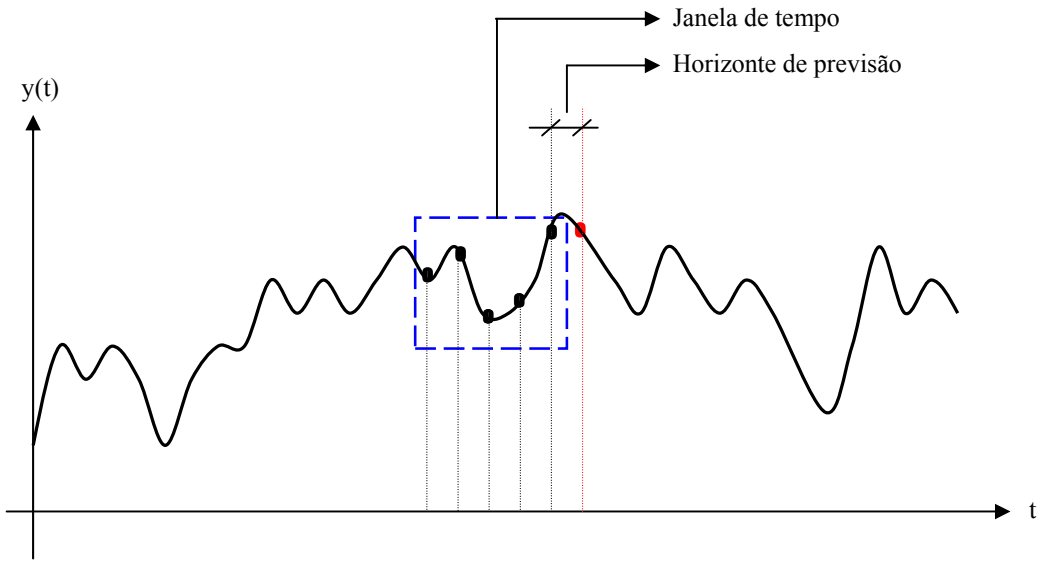


Figura 4.3.1:Características importantes em uma série temporal

De acordo com as definições, podemos classificar os sistemas de previsão da seguinte forma:

- Previsão em longo prazo: é realizada para um longo horizonte de previsão, normalmente dado em anos. Este tipo de previsão é aplicado, por exemplo, no planejamento da expansão de sistemas elétricos.

- Previsão em médio prazo: considera um período de tempo de algumas semanas ou meses. Pode ser utilizada, por exemplo, em planejamento de programas de manutenção.

- Previsão em curto prazo: neste caso o horizonte de previsão pode ser de meia hora a poucas horas. É utilizado, por exemplo, para questões de segurança, nos casos de avaliação de picos de carga elétrica.

- Previsão em curtíssimo prazo: o horizonte de predição pode variar de segundos a quinze minutos. Normalmente estes tipos de sistemas são utilizados em planejamento e controle instantâneos (*on-line*).

4.4 Modelos para análise de séries temporais

Os modelos mais comuns utilizados na representação de séries temporais são aqueles que estão associados a determinados parâmetros (ou coeficientes), denominados modelos paramétricos. Os principais são apresentados a seguir:

- Modelo Auto-Regressivo (AR)

É método capaz de representar uma determinada observação através da combinação linear de *p* valores passados: *y(k-1), y(k-2),... , y(k-p),* da mesma série temporal:

$$y(k) = a_1 y(k-1) + a_2 y(k-2) + ... + a_p y(k-p) + \varepsilon(k)$$

Onde a_1, a_2,..., a_p são os parâmetros do modelo e ε(k) é o erro da representação.

O modelo auto-regressivo pode ser usado para a representação de séries temporais estacionárias e não-estacionárias.

- Modelo Média Móvel (MA)

Difere-se do modelo AR, pois considera o valor médio da série temporal. O mesmo pode ser escrito da seguinte forma:

$$y(k) = \mu + \varepsilon(k) + c_1\varepsilon(k-1) + c_2\varepsilon(k-2) + ... + c_p\varepsilon(k-p)$$

Onde c_1, c_2, ..., c_p são os parâmetros do modelo e ε(k) é uma variável com média zero e variância constante e μ é a média da série.

- Modelo Auto-Regressivo - Média Móvel (ARMA)

É uma associação dos modelos anteriores e pode ser escrito da seguinte forma:

$$y(k) = ay(k-1) + \varepsilon(k) + c\varepsilon(k-1)$$

Neste, pode-se dizer que o processo será estacionário nos casos em que $|a| < 1$.

- Modelo Auto-Regressivo – Integrado - Média Móvel (ARIMA)

É um dos modelos mais utilizados, e com o propósito de melhor representar séries não-estacionárias. Este permite a transformação de séries não-estacionárias em estacionárias por meio de diferenças entre observações sucessivas. O mesmo pode ser representado da seguinte forma:

$$y(k) = y(k-1) + a_1(y(k-1) - y(k-2)) + a_2(y(k-2) - y(k-3)) + \varepsilon(k)$$

A breve abordagem dos modelos anteriores tem propósito apenas de apresentação da diversidade de metodologias voltadas à análise de séries temporais.

CAPÍTULO 5
TESTES PRÁTICOS

Este capítulo marca o início do estudo de casos e apresenta uma série de testes realizados com as técnicas anteriores, operando isoladamente ou em conjunto, sob a aplicação de diversos conjuntos de dados.

5.1 Conjuntos de dados

Após a apresentação dos principais tipos de técnicas de inteligência artificial que podem ser aplicados em sistemas de previsão de séries temporais, faz-se necessário a realização de testes com diferentes conjuntos de dados para a avaliação dos resultados produzidos por cada uma delas. Todas as simulações foram realizadas no software de simulação Matlab 7.4.0, em um microcomputador com processador AMD® Sempron 1.80 GHz e 1GB de memória RAM.

Sendo assim, são utilizadas duas séries de dados. A primeira se trata de uma série temporal contendo valores reais da demanda de energia elétrica no estado de Minas Gerais pelo período de um ano. O gráfico seguinte apresenta o comportamento da demanda, medida em intervalos de tempo iguais a quinze minutos, durante todo o ano de 2004. Desta forma, cada curva é representada por 96 pontos, espaçados de quinze minutos, ao longo das 24 horas.

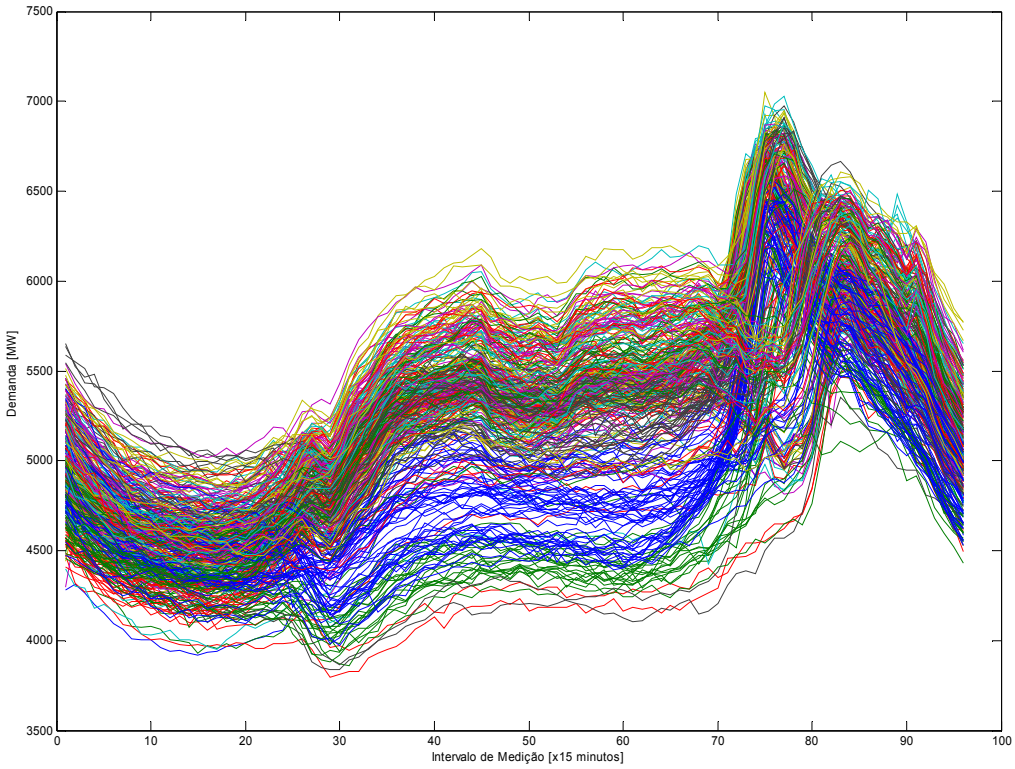


Figura 5.1.1: Demanda diária, registrada no ano de 2004, medida em intervalos de 15 minutos

A segunda, produzida por Box e Jenkins [33], é formada por dados provenientes de uma fornalha onde a entrada é a taxa de alimentação de gás metano e a saída, a concentração de dióxido de carbono (% CO_2) em uma mistura de gases.

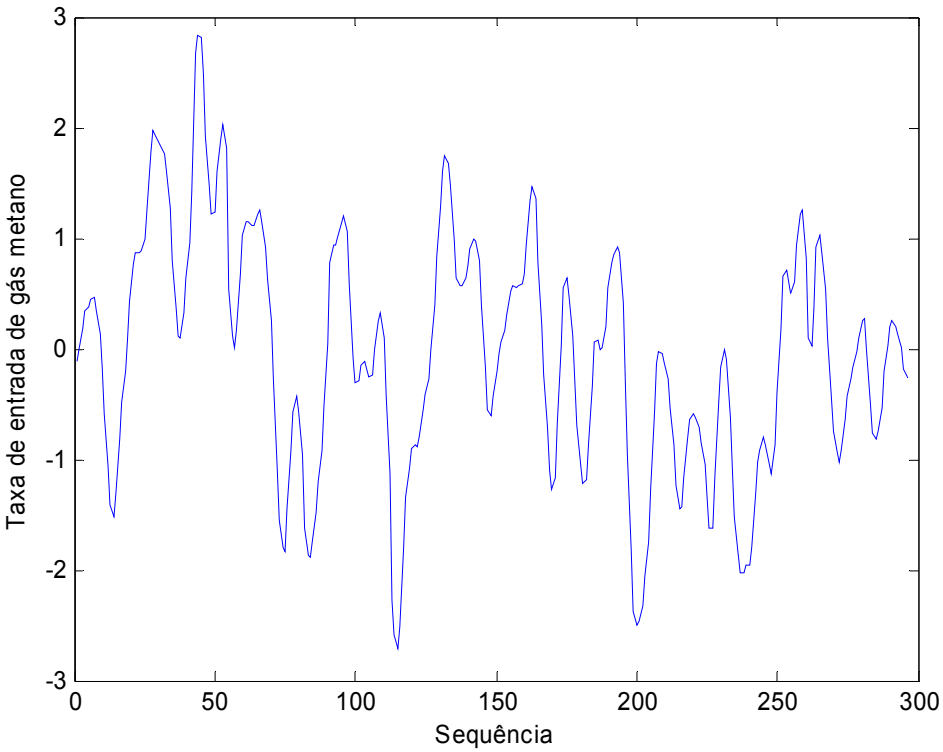


Figura 5.1.2: Taxa de entrada de gás metano na fornalha de Box e Jenkins

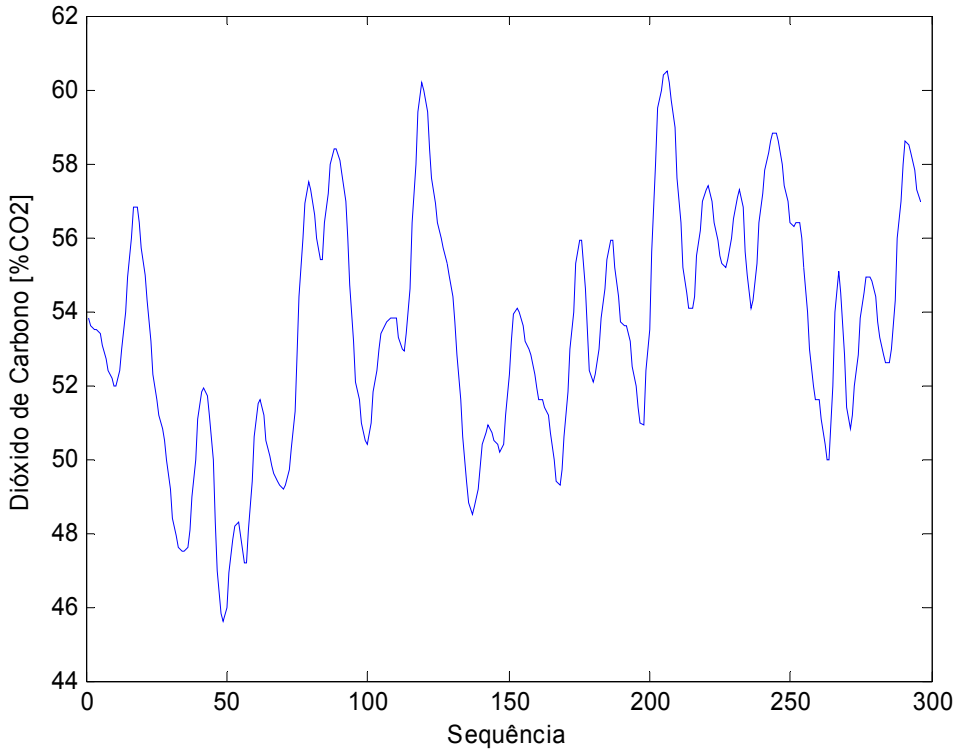


Figura 5.1.3: Porcentagem de emissão de dióxido de carbono em uma mistura de gases

Esta série é composta por uma seqüência de 296 pares de entrada-saída. Destes, foram utilizadas 148 para o conjunto de treinamento e o restante para o conjunto de validação. Ambos estão organizados de acordo com a tabela a seguir:

Entradas	Saída
u(t-3)	y(t)
u(t-2)	
u(t-1)	
y(t-3)	
y(t-2)	
y(t-1)	

Onde:

- u(t-n) corresponde a entrada atrasada *n* instantes de tempo;

- y(t-m) é a saída atrasada *m* instantes de tempo.

Para a realização dos testes referentes à primeira série de dados foram criados quatro conjuntos para os processos de treinamento e validação. O primeiro contendo amostras de medições realizadas em diferentes horários de um dia específico da semana. O dia escolhido foi o de quarta-feira. O gráfico a seguir apresenta o comportamento da demanda registrada em intervalos de quinze minutos, durante todas as quartas-feiras do ano de 2004:

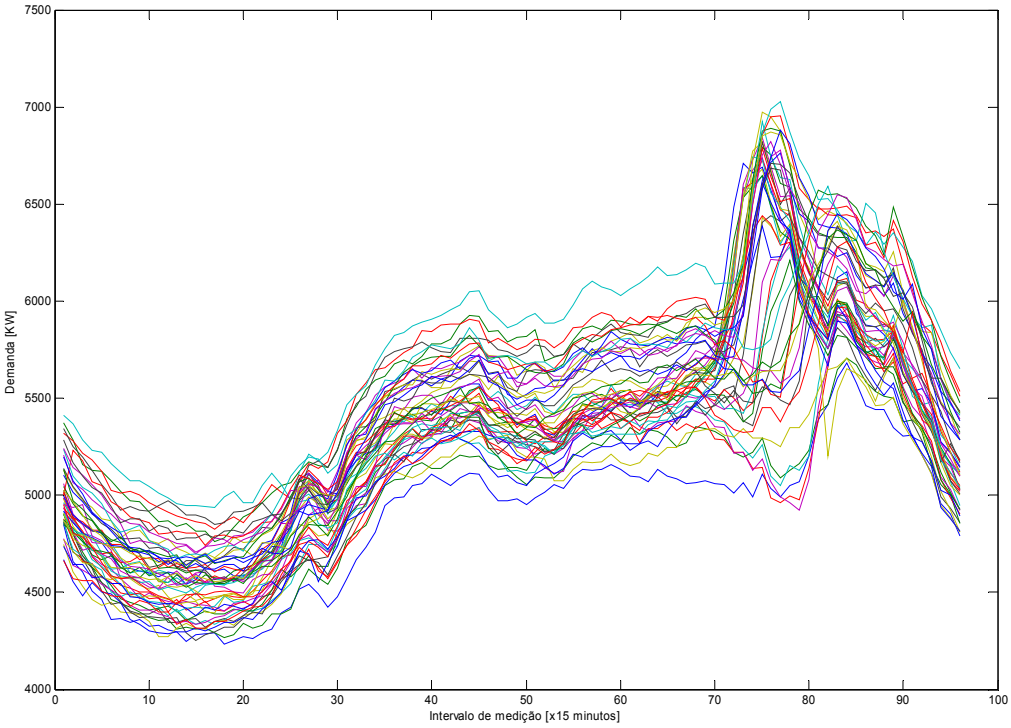


Figura 5.1.4: Demanda registrada em todas as quartas-feiras do ano de 2004

Neste caso, os dados estão organizados de forma que na entrada sejam utilizados valores temporalmente espaçados de quinze minutos. A tabela a seguir apresenta o esquema de elaboração dos conjuntos de treinamento e validação utilizados nos testes práticos.

PERÍODO	HORÁRIO	ENTRADA
07/01/2004 a 29/12/2004	(hh_a:mm_a) - 00:45	Demanda registrada a quarenta e cinco minutos do horário atual.
	(hh_a:mm_a) - 00:30	Demanda registrada a trinta minutos do horário atual.
	(hh_a:mm_a)	Demanda registrada no horário atual.
	(hh_a:mm_a) - 168:00	Demanda registrada no horário atual a uma semana.
	(hh_a:mm_a) - 167:45	Demanda registrada quinze minutos após o horário atual, na semana anterior.
	(hh_a:mm_a) - 168:45	Demanda registrada a quarenta e cinco minutos do horário atual, na semana anterior.
	(hh_a:mm_a) – 168:30	Demanda registrada a trinta minutos do horário atual, na semana anterior.
	(hh_a:mm_a) - 168:15	Demanda registrada a quinze minutos do horário atual, na semana anterior.

Onde:

hh_a:mm_a = representa o horário atual.

Para formação do conjunto de treinamento são utilizadas 3500 medições realizadas em diferentes horários, de acordo com o esquema acima e para o conjunto de validação outras 500 amostras.

Abaixo um esquema que mostra as entradas e saídas do sistema de previsão proposto:

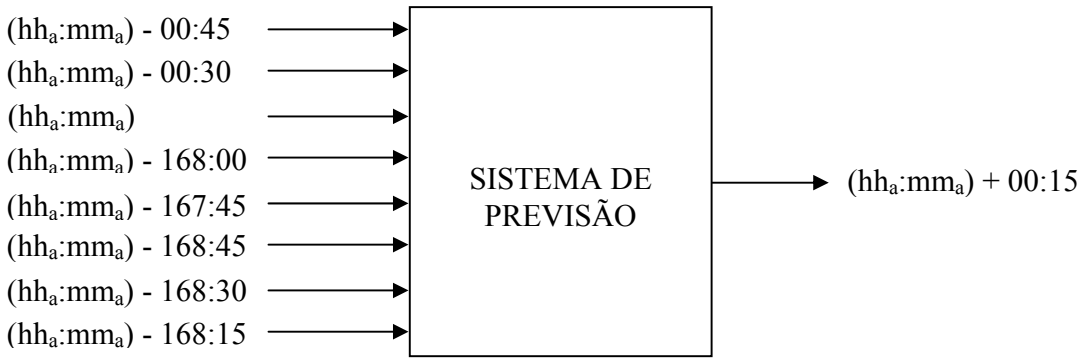


Figura 5.1.5: Padrão adotado de entradas do sistema de previsão para a realização dos testes práticos

Ainda com a primeira série temporal foram construídos três outros conjuntos de treinamento. Para isto foram utilizadas amostras de medições realizadas em um determinado horário, independente do dia da semana. Escolheu-se, arbitrariamente, o horário de doze horas.

A organização dos dados seguiu a metodologia anterior, respeitando-se os intervalos de tempo. O gráfico a seguir mostra o comportamento da demanda medida ao longo do ano de 2004 no horário de doze horas:

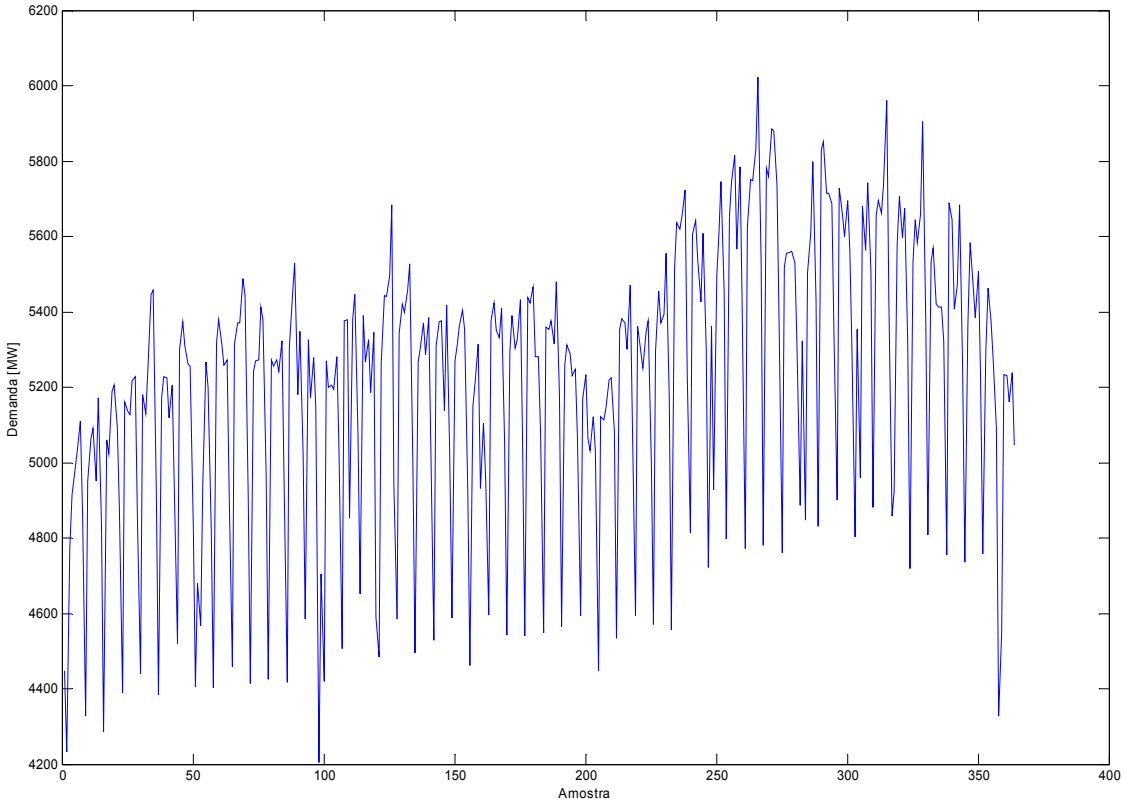


Figura 5.1.6: Demanda registrada no horário de 12:00 horas ao longo do ano de 2004

Para efeito de diversidade nos testes, o conjunto contendo 364 amostras foi subdivido em três outros, cada um com 100 amostras utilizadas no treinamento e 20 o processo de validação, como mostrados nos gráfico a seguir:

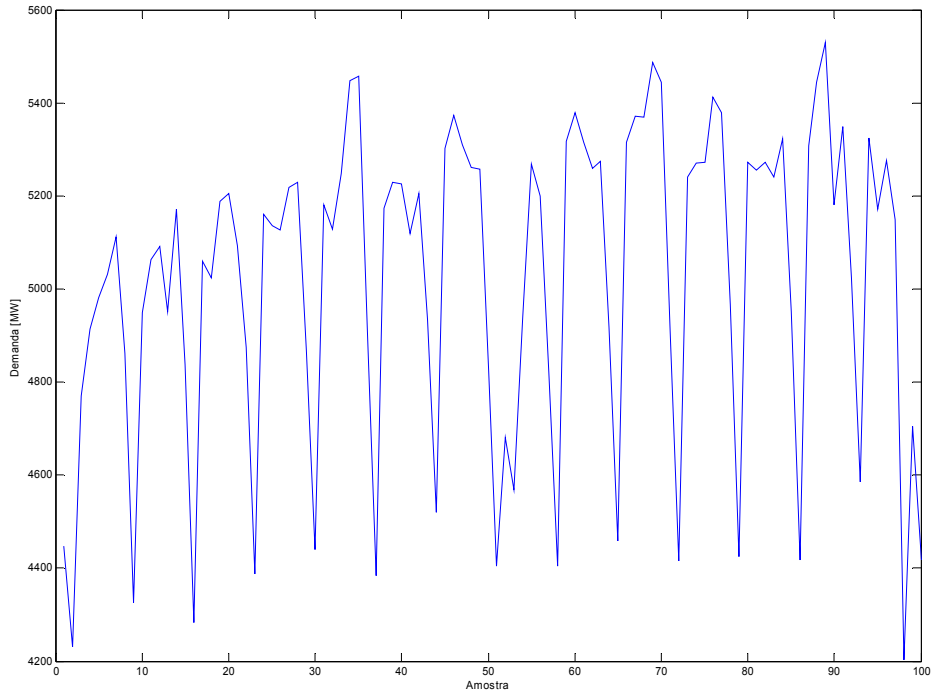


Figura 5.1.7: Primeiro conjunto de treinamento com valores de medições realizadas às 12:00

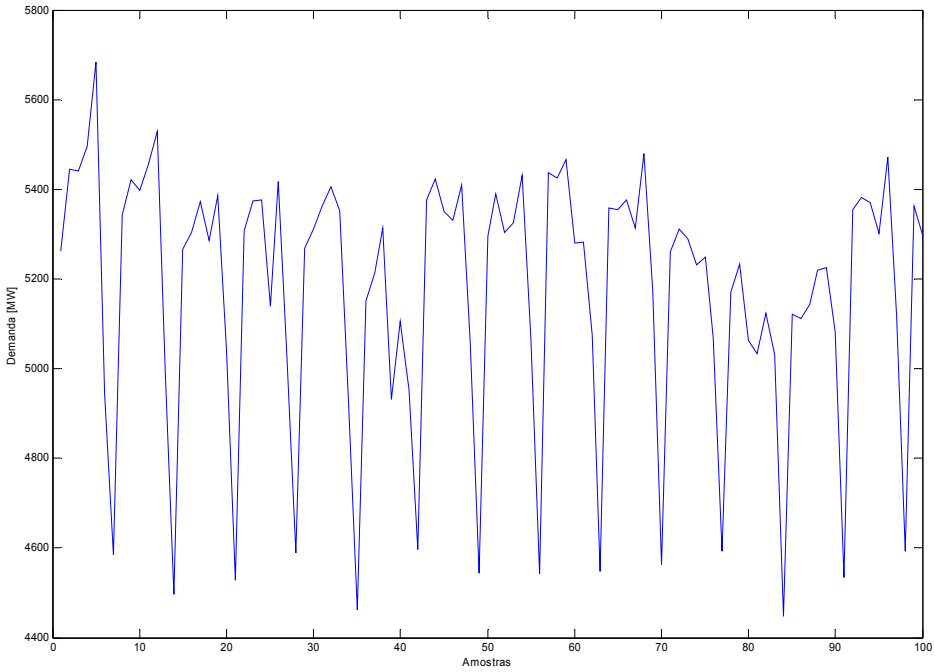


Figura 5.1.8: Segundo conjunto de treinamento com valores de medições realizadas às 12:00

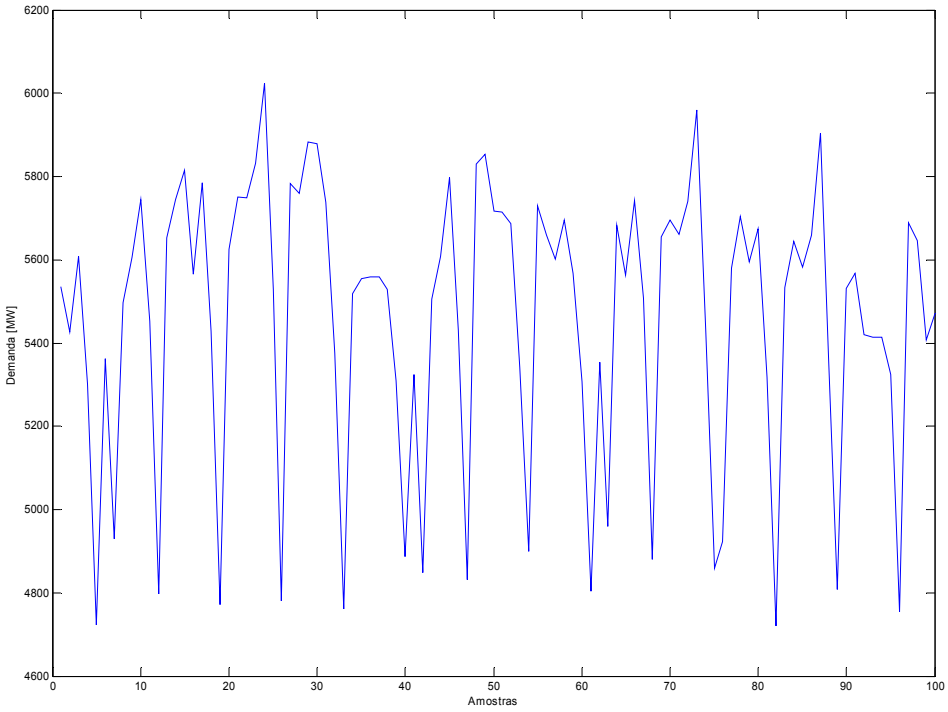


Figura 5.1.9: Terceiro conjunto de treinamento com valores de medições realizadas às 12:00

Para avaliação das RNA quanto aos processos de treinamento e validação, normalmente é utilizado o erro médio quadrático - MSE (*Mean Squared Error*) ou a soma dos erros quadráticos - SSE (*Sum Squared Error*) como medida de performance.

Neste trabalho foi escolhida a soma dos erros quadráticos como parâmetro de avaliação do desempenho de todas as RNA. O mesmo é calculado da seguinte forma:

$$SSE = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (T_i - y_i)^2$$

Onde:

e = corresponde à diferença entre o valor desejado (T) e o valor produzido na saída do sistema (y)

n = número de observações

A partir deste modelo e dos conjuntos de dados apresentados são iniciados os testes com os diversos tipos de RNA apresentadas.

5.2 Testes realizados com redes neurais artificiais do tipo MLP

Os primeiros testes realizados têm o objetivo de verificar os resultados produzidos por RNA do tipo *Multilayer Perceptron Network (MLP)*. A importância de avaliá-las deve-se ao fato de serem altamente difundidas e pelo baixo tempo necessário para o treinamento.

Para isto é utilizado o algoritmo de retropropagação de erros, ou *backpropagation*. Como topologia de referência é utilizada aquela onde os elementos de processamento estão dispostos em camadas, conectados a todos os outros da camada seguinte e com o fluxo de sinal partindo da entrada em direção à saída, como apresentado a seguir:

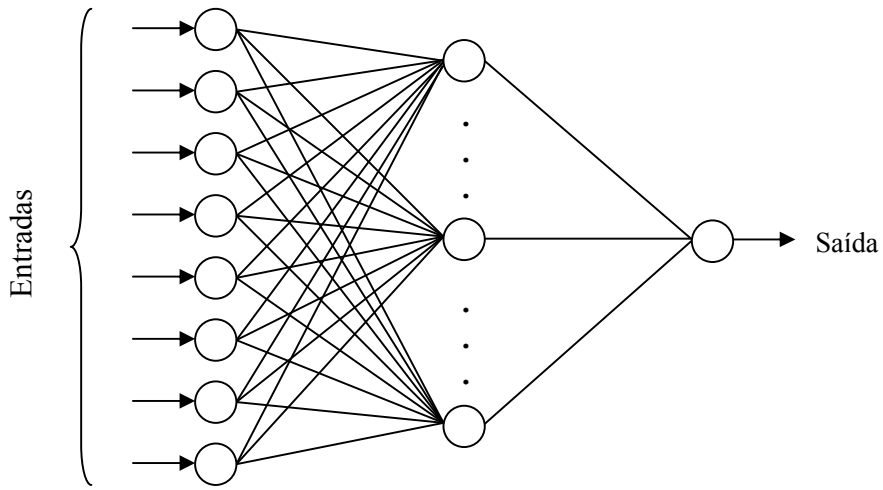


Figura 5.1.10 : Topologia de referência para RNA do tipo MLP

Vale relembrar que um inconveniente deste tipo de estrutura é o de não ser possível determinar o número ótimo de elementos de processamento (neurônios) na camada intermediária. Sendo assim é fundamental a realização de diversos testes com diferentes topologias. Os melhores resultados são apresentados a seguir, assim como as topologias otimizadas para cada caso. Nos gráficos apresentados adotou-se como "saída real" aquela obtida na aplicação das técnicas e como "saída desejada" o valor esperado para o padrão de entrada em questão.

5.2.1 Resultados Obtidos

Os gráficos a seguir apresentam os resultados dos processos de treinamento e validação realizados com todos os conjuntos de dados, utilizando RNA do tipo MLP.

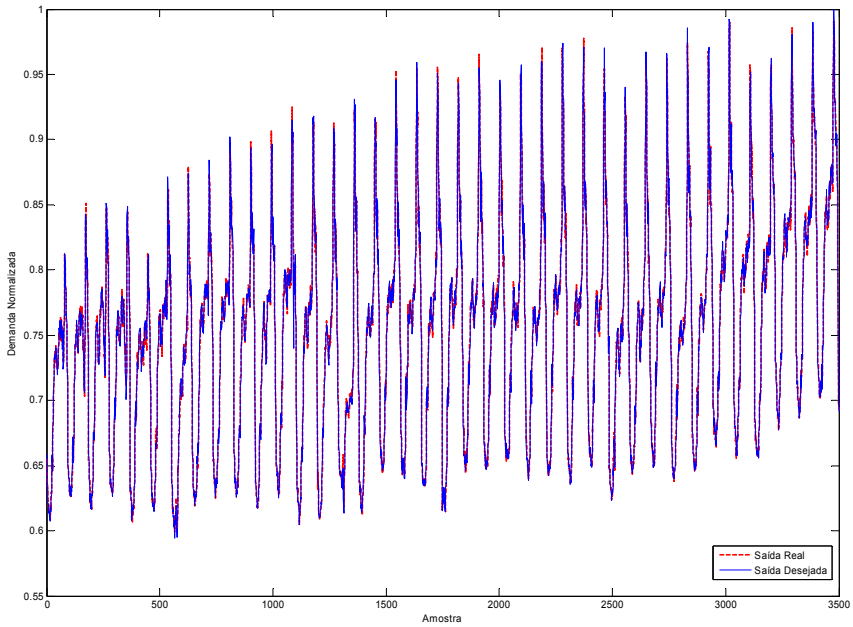


Figura 5.2.1.1: Resultado do processo de treinamento para o primeiro conjunto de dados

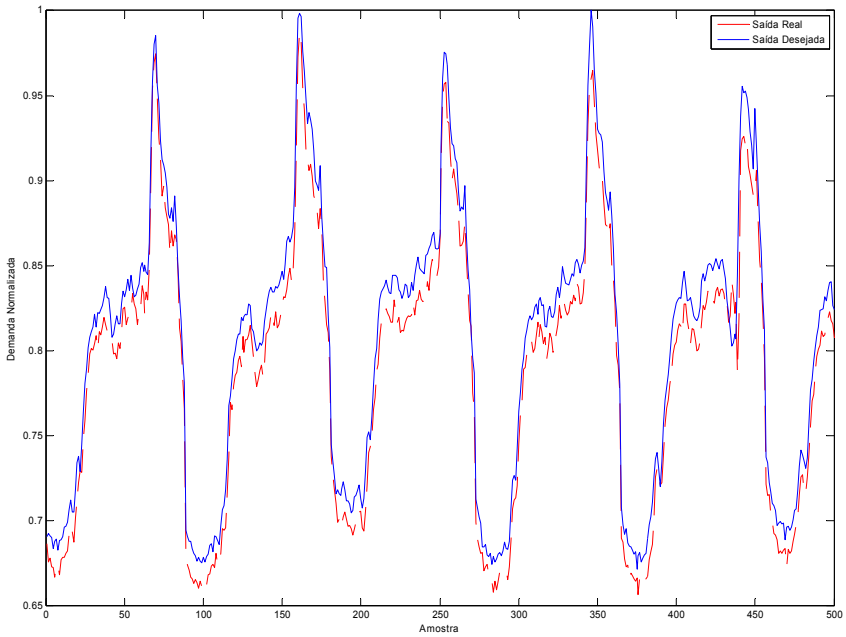


Figura 5.2.1.2: Resultado do processo de validação para o primeiro conjunto de dados

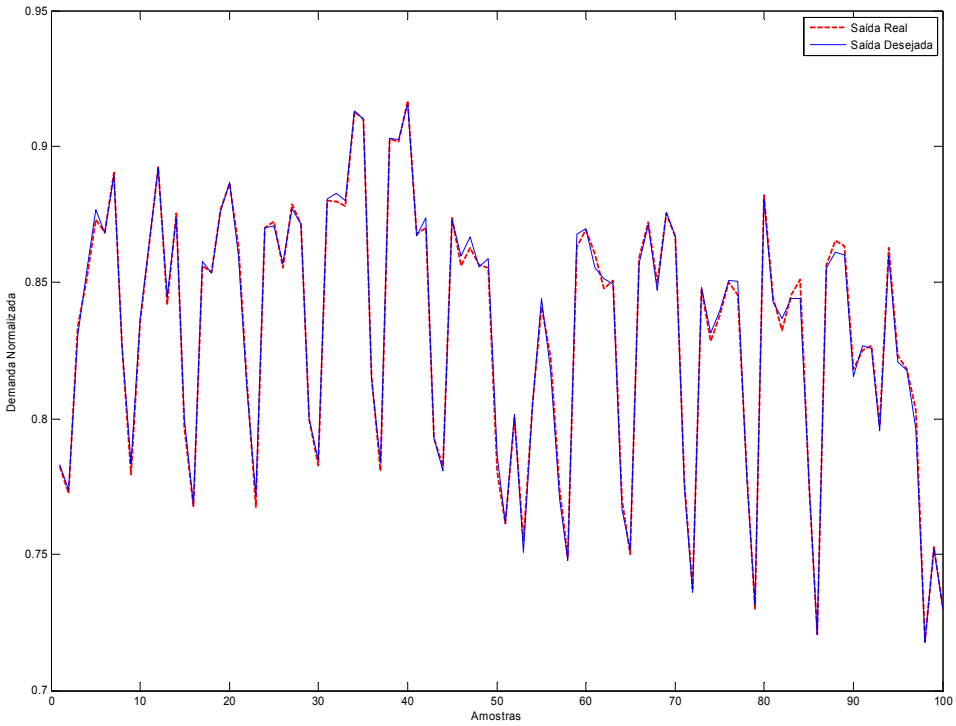


Figura 5.2.1.3: Resultado do processo de treinamento para o segundo conjunto de dados

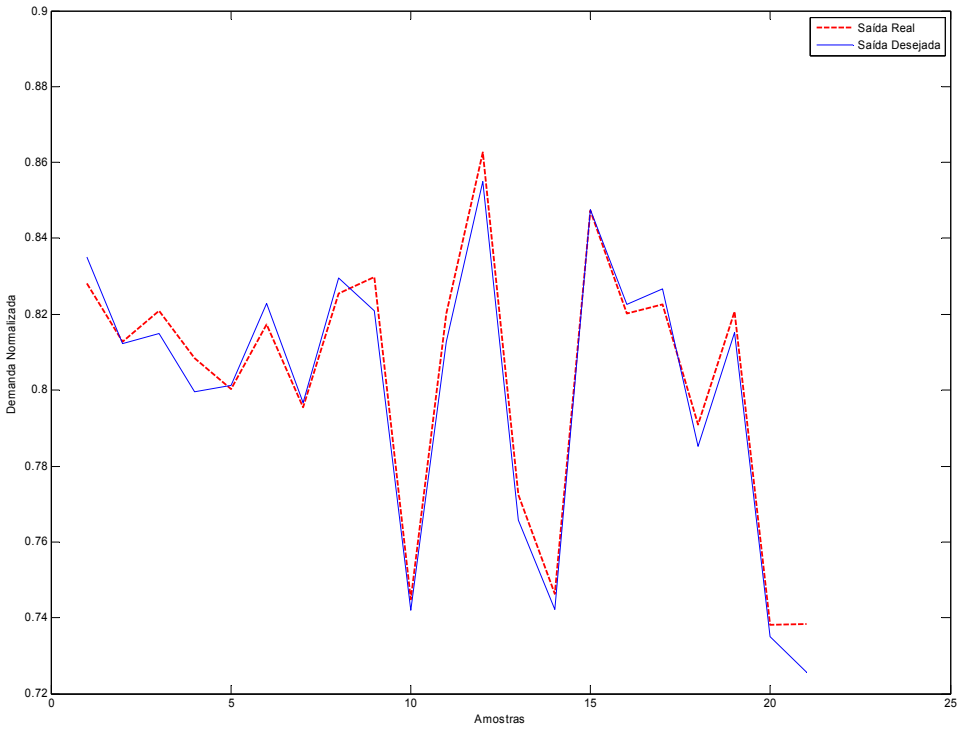


Figura 5.2.1.4: Resultado do processo de validação para o segundo conjunto de dados

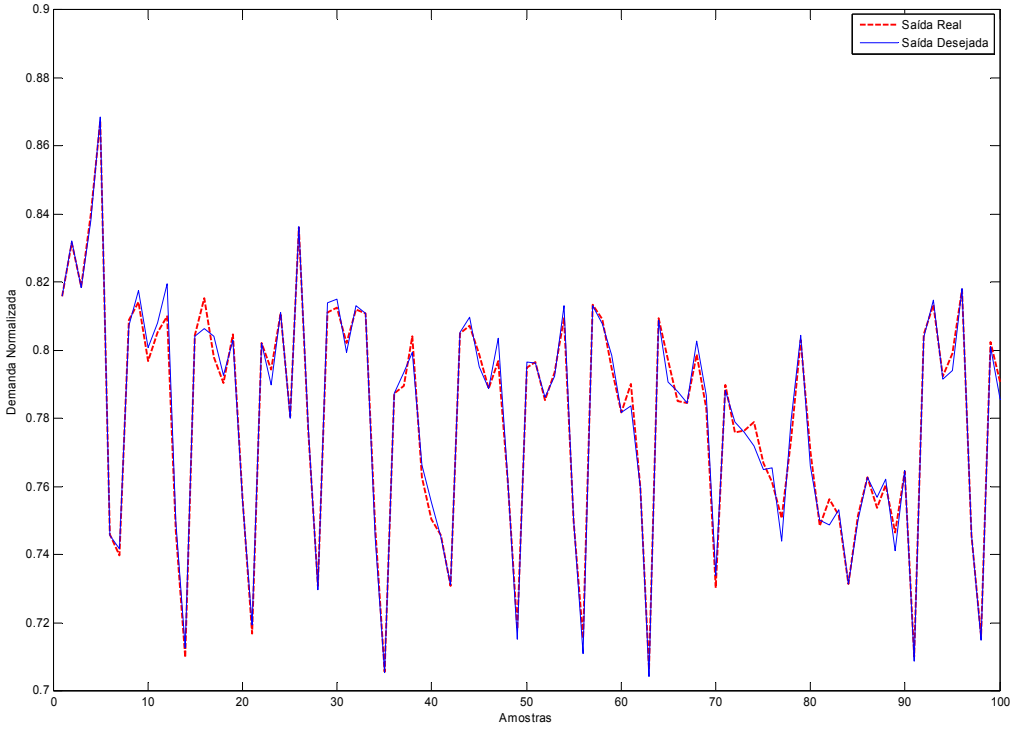


Figura 5.2.1.5: Resultado do processo de treinamento para o terceiro conjunto de dados

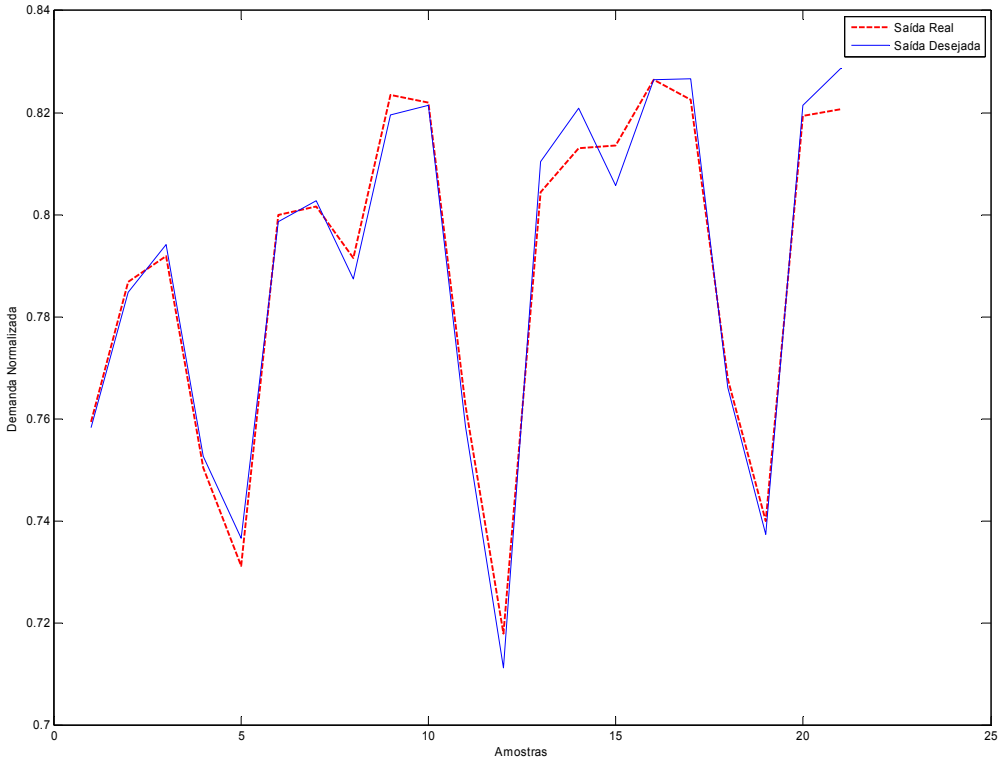


Figura 5.2.1.6: Resultado do processo de validação para o terceiro conjunto de dados

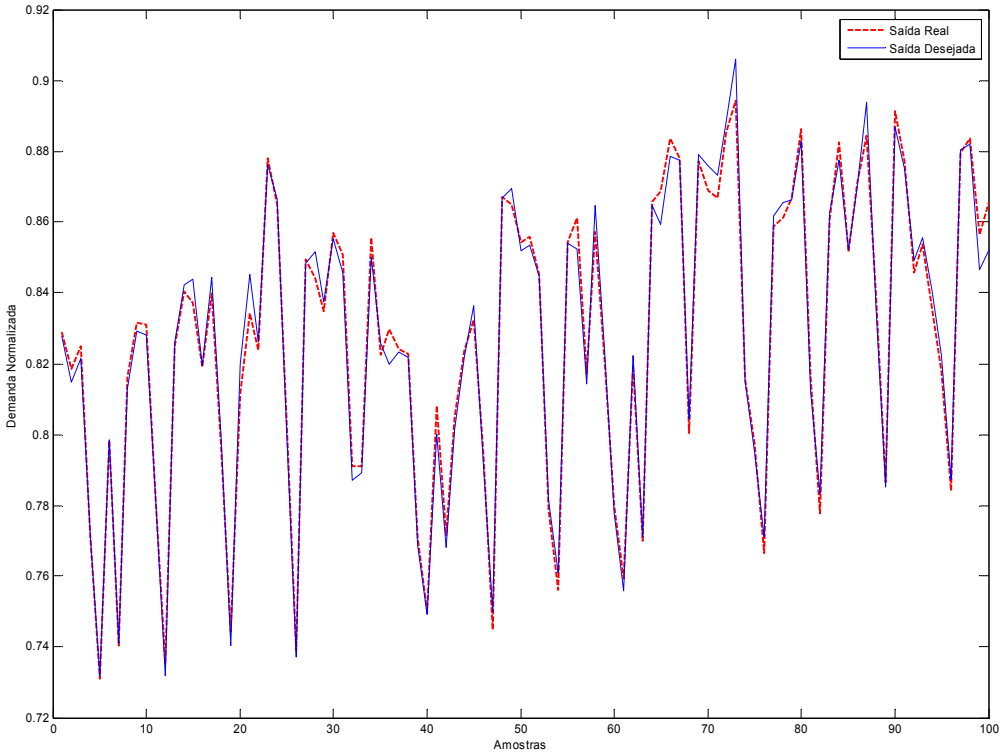


Figura 5.2.1.7: Resultado do processo de treinamento para o quarto conjunto de dados

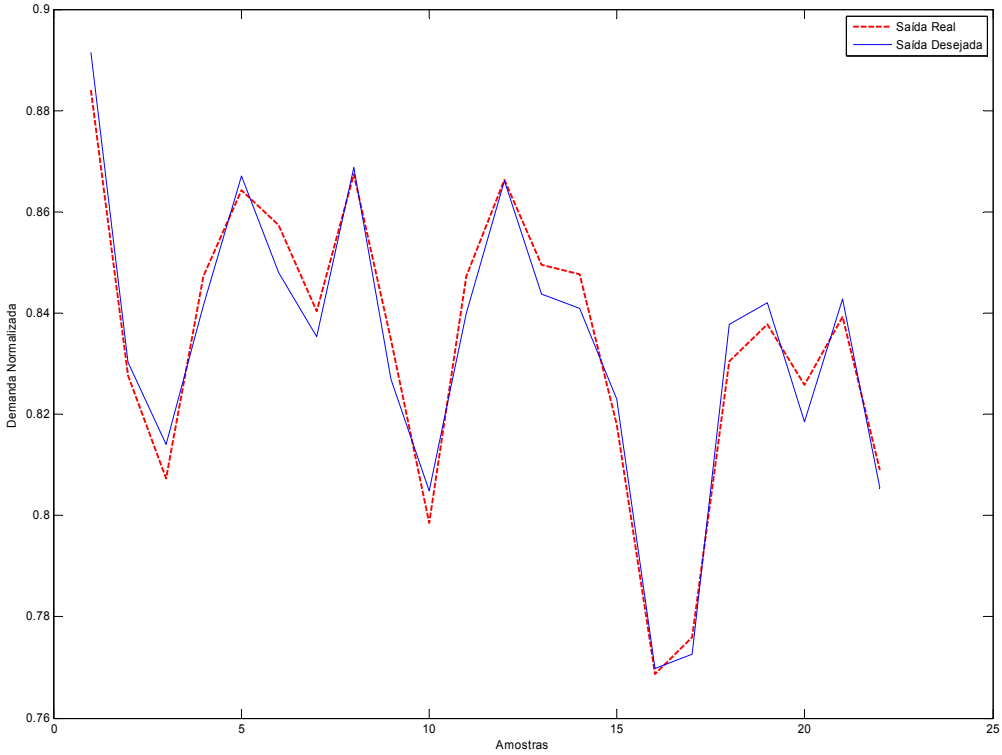


Figura 5.2.1.8: Resultado do processo de validação para o quarto conjunto de dados

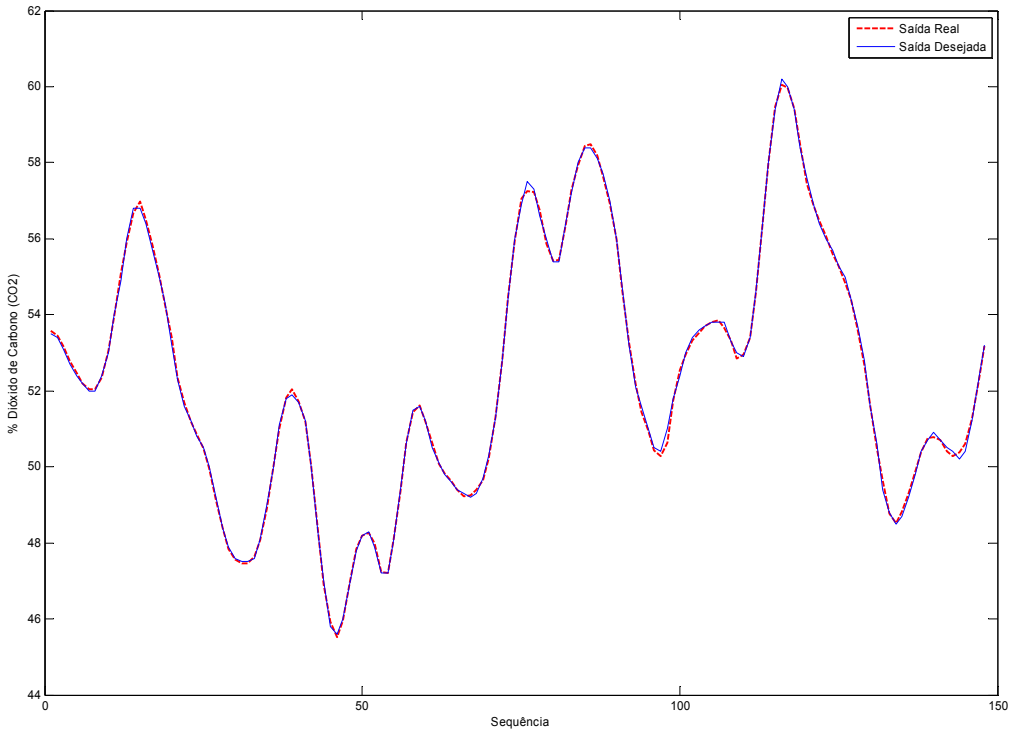


Figura 5.2.1.9: Resultado do processo de treinamento para a série de dados Gas Furnace

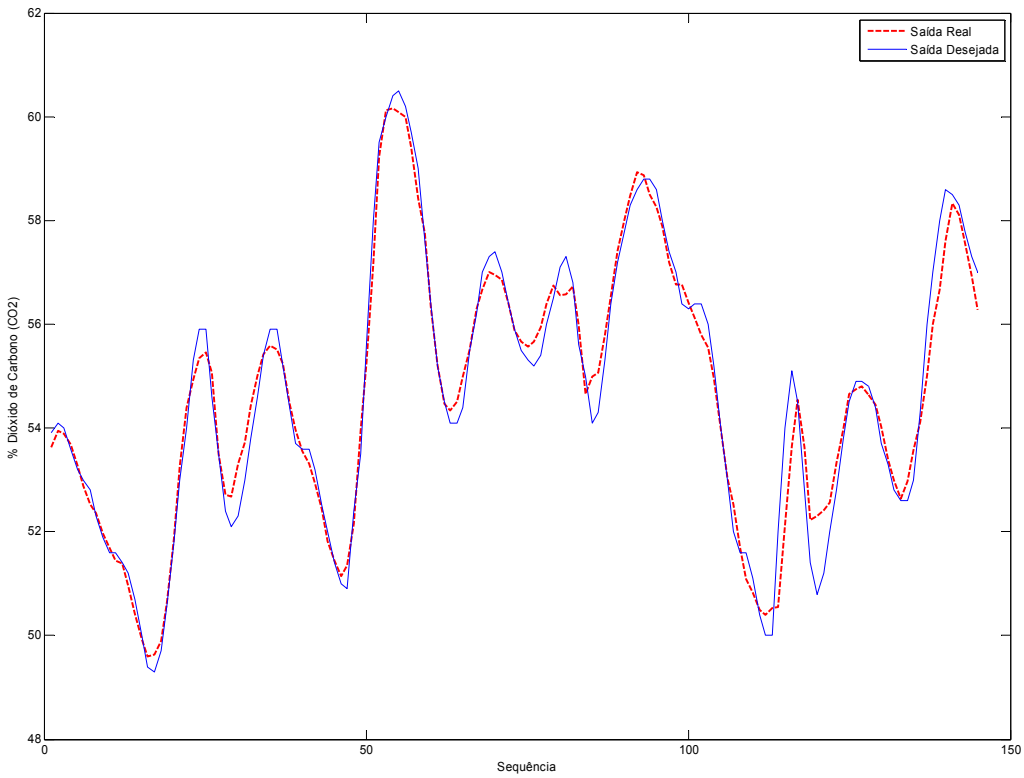


Figura 5.2.1.10: Resultado do processo de validação para a série de dados Gas Furnace

Adiante é apresentada uma tabela contendo dados referentes às topologias finais e os valores da soma dos erros quadráticos (SSE) para todos os casos:

CONJUNTO DE DADOS	TOPOLOGIA	$SSE_{TREINAMENTO}$	$SSE_{VALIDAÇÃO}$
1	8:4:1	0,14	0,1668
2	8:3:1	6,55e-004	7,21e-004
3	8:2:1	0,0014	3,92e-004
4	8:3:1	0,0020	6,83e-004
Gas Furnace	6:4:1	1,2102	33,6402

Tabela 5.2.1.1: Resultados obtidos com RNA do tipo MLP, em relação a todos os conjuntos de dados

5.2.2 Avaliação dos Resultados

Pelos resultados apresentados pode-se considerar como razoável a performance das RNA em todos os casos.

É importante ressaltar que a medida de desempenho adotada apresenta um valor referente à soma dos erros quadráticos ocorridos ao longo dos processos de treinamento e validação, ou seja, o valor final é cumulativo. Desta forma, quando a RNA responde de forma insatisfatória à apresentação de um dado padrão, isto impacta de forma negativa na avaliação global. Justamente por este motivo foi escolhido este tipo de parâmetro.

5.3 Testes realizados com redes neurais polinomiais

Os testes realizados a seguir envolvem um tipo de RNA classificada como auto-organizável ou plástica. Isto se deve ao fato da topologia ser definida ao longo do processo de treinamento. A importância de avaliá-las se deve ao fato da produção de bons resultados com topologias compactas e do baixíssimo tempo de processamento no processo de treinamento. Isto faz com que a mesma possa ser reprojetada para se adaptar a novos padrões sem elevado esforço.

Como dito anteriormente a camada 1 é formada através da combinação das entradas, duas a duas. Nesta são selecionados os *r* melhores neurônios, que serão combinados dois a dois para formar a camada 2. Finalmente nesta são selecionados os dois melhores elementos que formarão a camada 3. A fase de seleção, assim como a topologia final estão exemplificadas através das figuras a seguir:

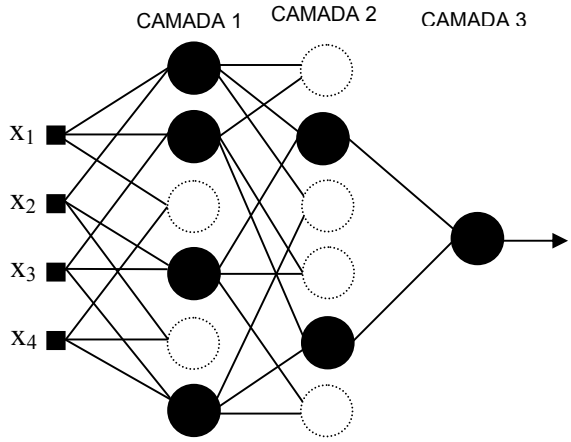


Figura 5.3.1: Processo de formação da topologia da RNP durante o processo de treinamento

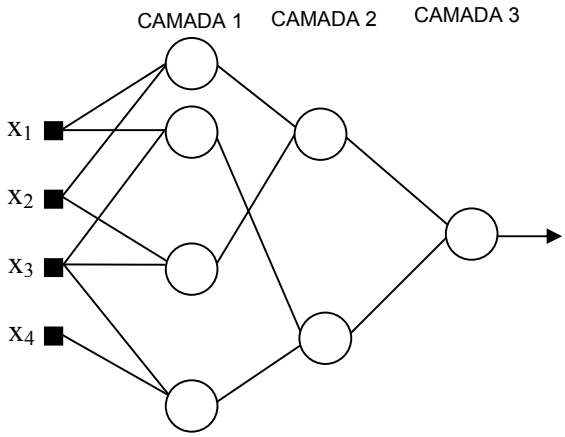


Figura 5.3.2: Exemplo de topologia final da RNP após o treinamento

5.3.1 Resultados Obtidos

Os gráficos a seguir apresentam os resultados dos processos de treinamento e validação realizados com todos os conjuntos de dados, utilizando redes neurais polinomiais.

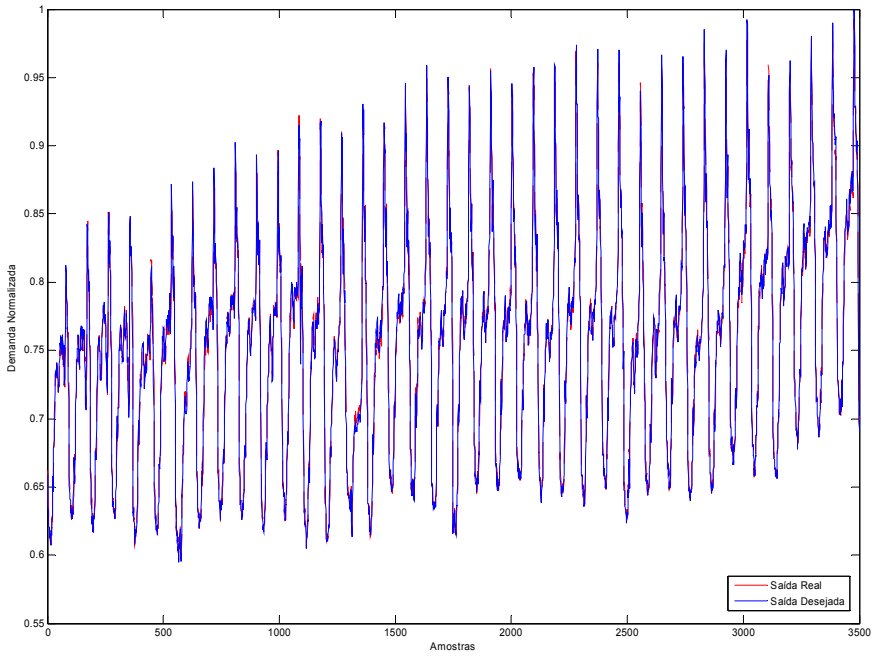


Figura 5.3.1.1: Resultado do processo de treinamento para o primeiro conjunto de dados

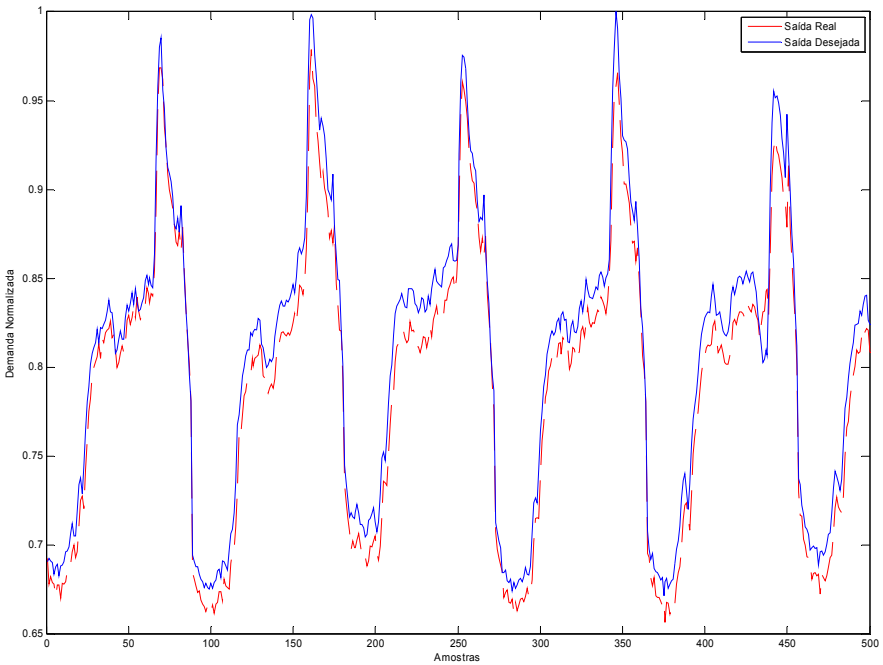


Figura 5.3.1.2: Resultado do processo de validação para o primeiro conjunto de dados

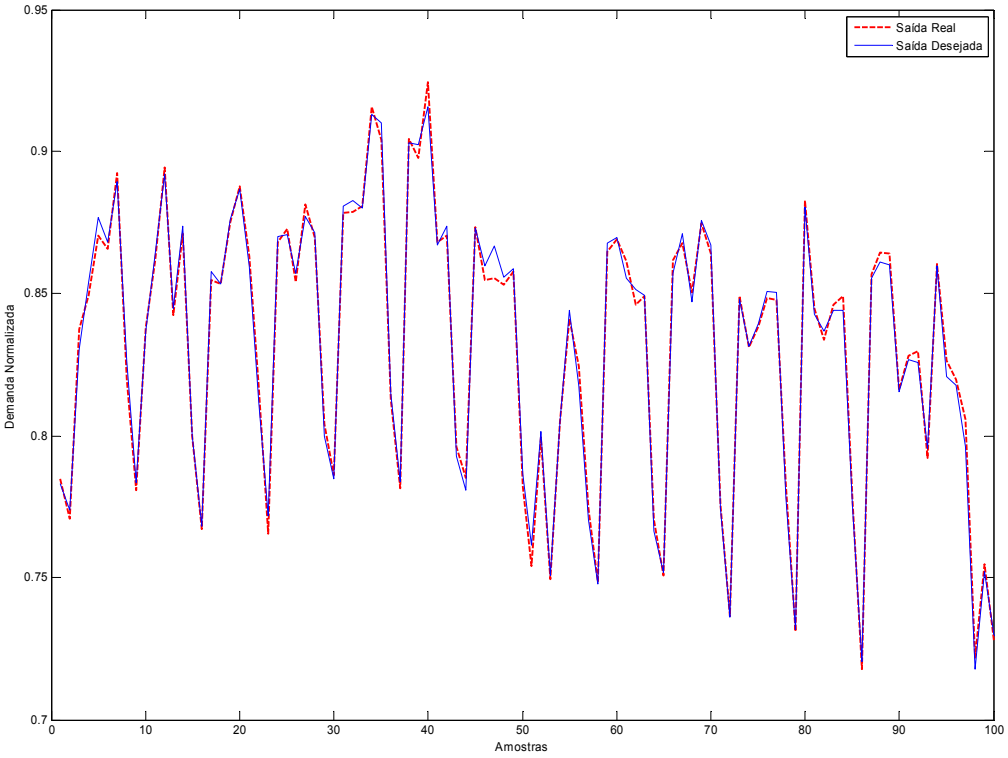


Figura 5.3.1.3: Resultado do processo de treinamento para o segundo conjunto de dados

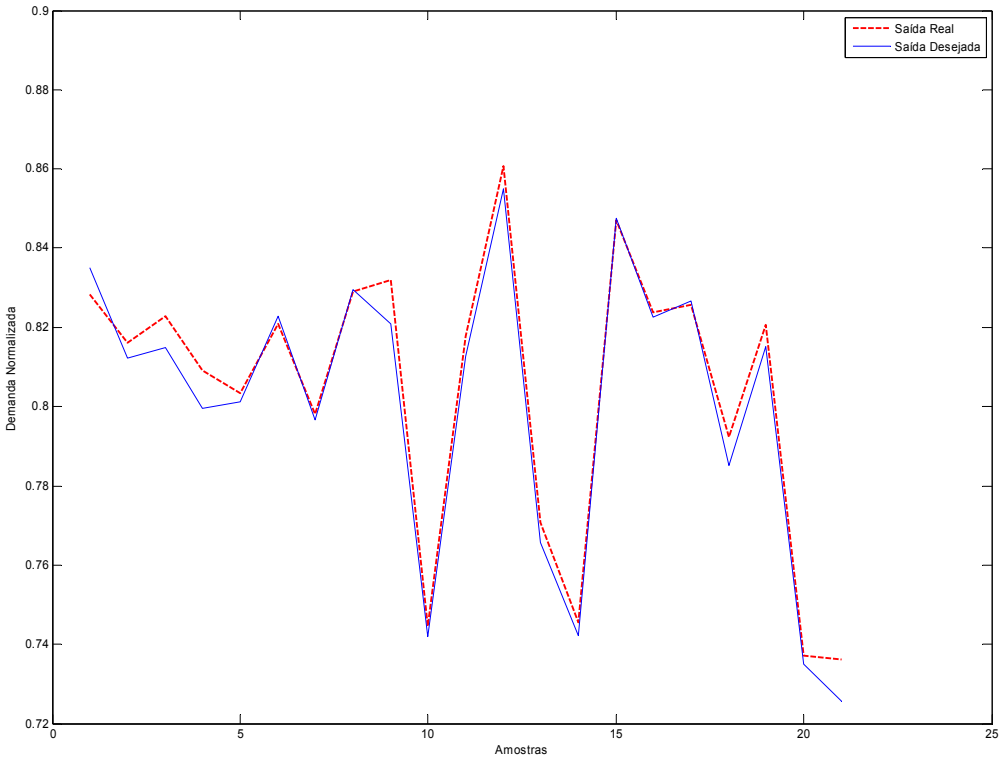


Figura 5.3.1.4: Resultado do processo de validação para o segundo conjunto de dados

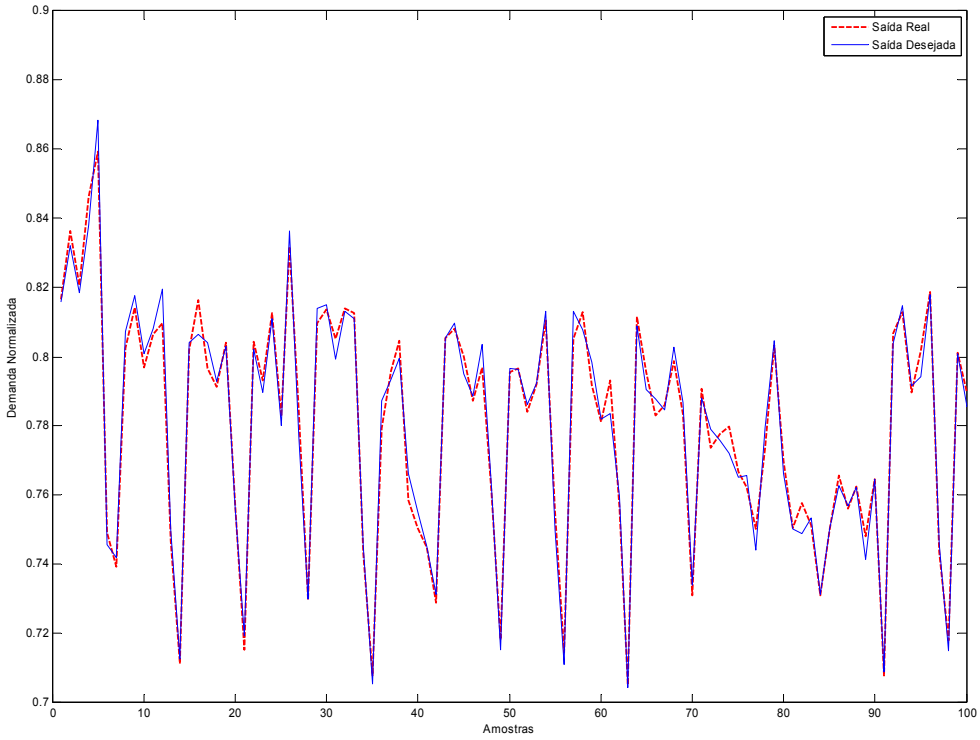


Figura 5.3.1.5: Resultado do processo de treinamento para o terceiro conjunto de dados

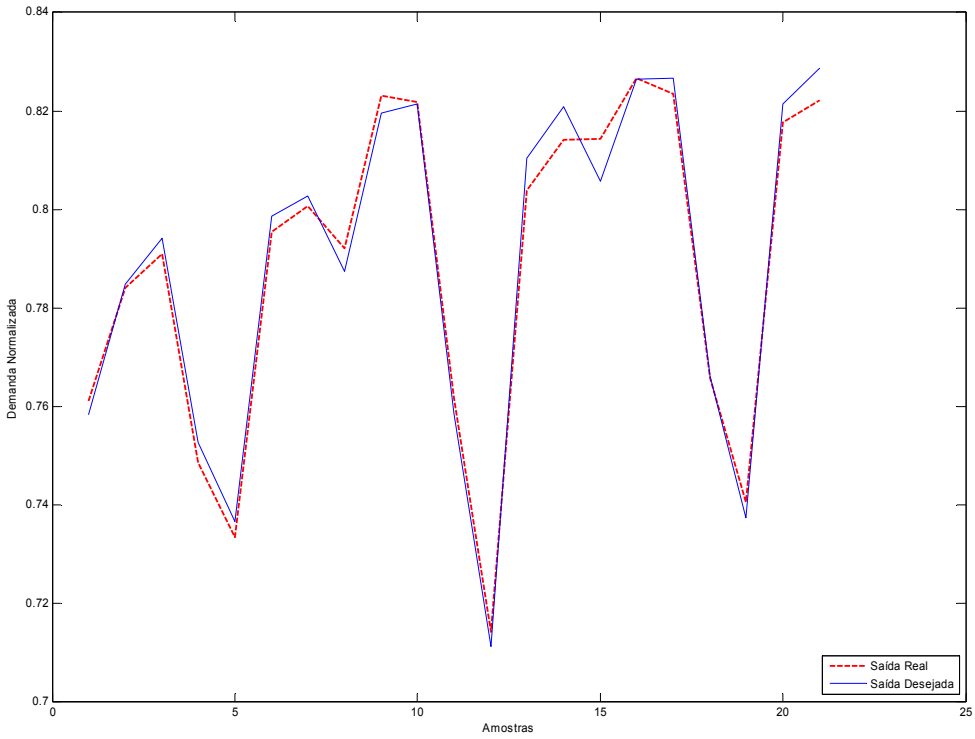


Figura 5.3.1.6: Resultado do processo de validação para o terceiro conjunto de dados

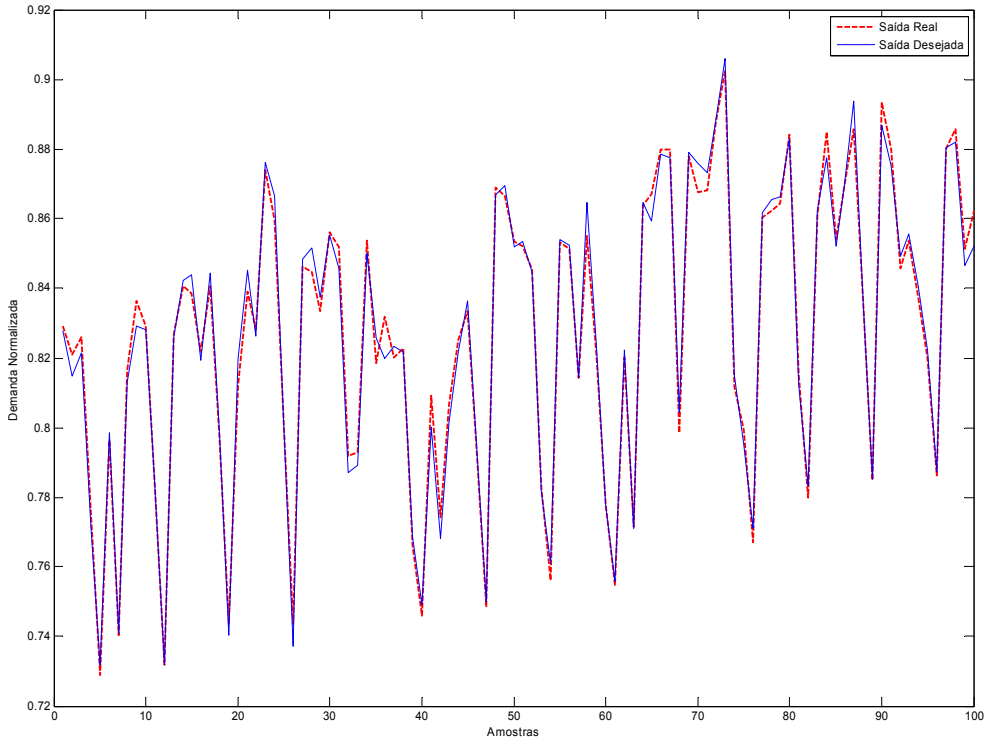


Figura 5.3.1.7: Resultado do processo de treinamento para o quarto conjunto de dados

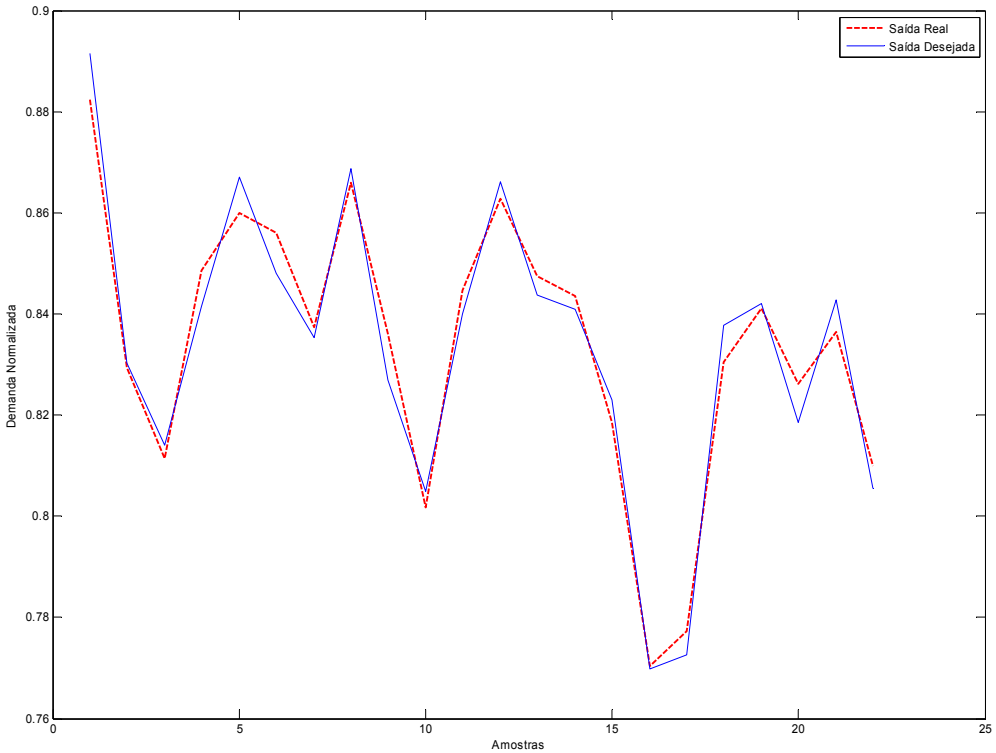


Figura 5.3.1.8: Resultado do processo de validação para o quarto conjunto de dados

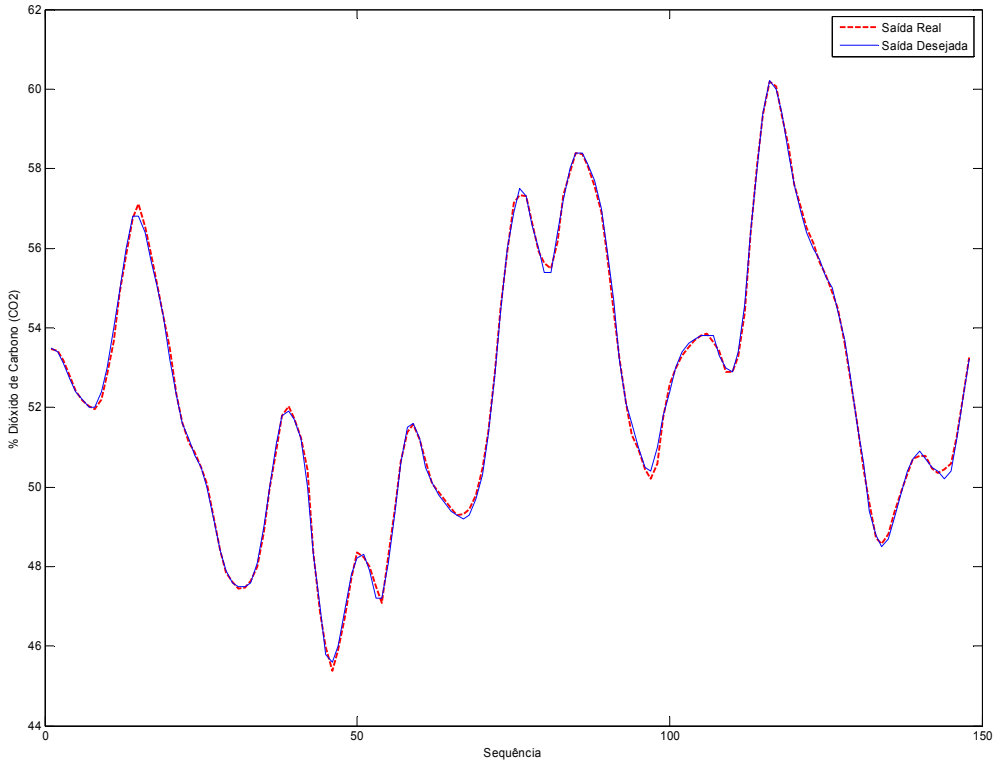


Figura 5.3.1.9: Resultado do processo de treinamento para a série de dados Gas Furnace

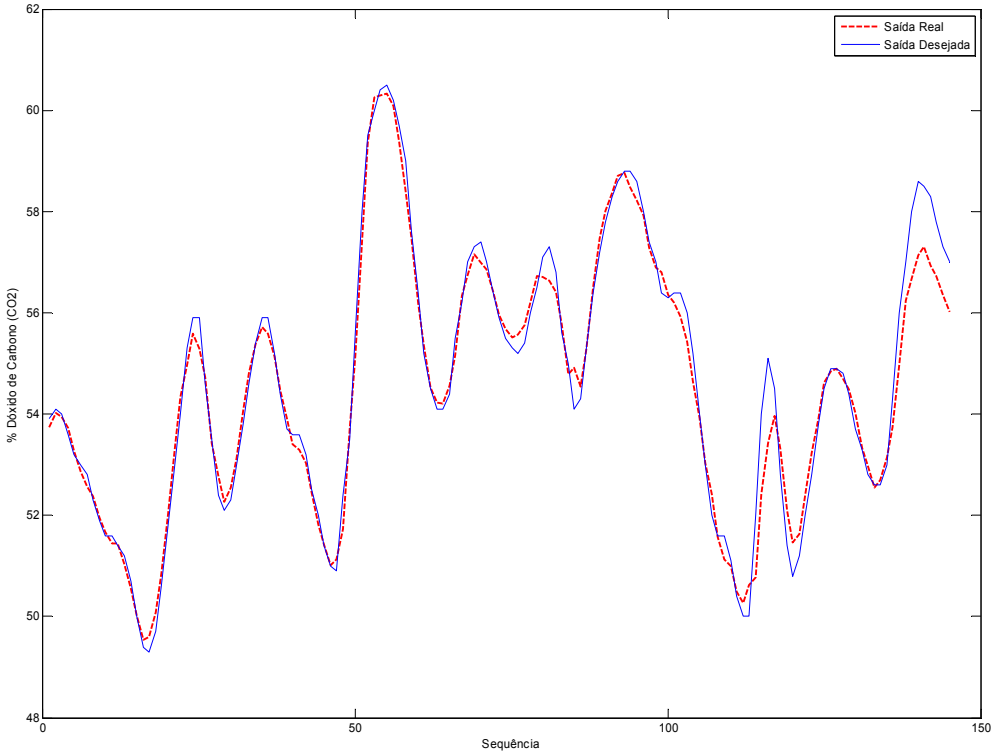


Figura 5.3.1.10: Resultado do processo de validação para a série de dados Gas Furnace

A tabela seguinte mostra os resultados obtidos nos testes realizados com redes neurais polinomiais:

CONJUNTO DE DADOS	$SSE_{TREINAMENTO}$	$SSE_{VALIDAÇÃO}$
1	0,2662	0,1628
2	0,00139	6,4551e-004
3	0,0018	3,5085e-004
4	0,00179	6,3026e-004
Gas Furnace	2,5207	29,9934

Tabela 5.3.1.1: Resultados obtidos com redes neurais polinomiais, em relação a todos os conjuntos de dados

5.3.2 Avaliação dos Resultados

Pelos resultados apresentados é possível comprovar o bom desempenho das redes neurais polinomiais. Um fato relevante é o de que, em todos os casos, obteve-se sucesso com uma topologia final compacta. Os tempos necessários ao treino são normalmente muito pequenos. A única desvantagem deste tipo de estrutura é a de não se poder determinar um número ótimo de neurônios, de forma que para a definição da topologia ótima são necessários vários testes. Isto a torna de certa forma inviável nos casos onde há a necessidade de novos treinamentos, devido à mudanças no conjunto de dados.

5.4 Testes realizados com redes neuro-fuzzy

As redes do tipo neuro-fuzzy, como dito anteriormente, buscam identificar um conjunto de regras do tipo “Se..Então”, capazes de representar relações entre pares entrada-saída. Por este motivo torna-se pertinente a realização de testes que comprovem seu comportamento diante de sistemas de previsão.

No projeto deste tipo de estrutura alguns parâmetros devem ser definidos. Os principais são os seguintes:

- número de conjuntos difusos por variável de entrada;

- tipo de função de pertinência de entrada;

- tipo de função de pertinência de saída;

- método de treinamento

É muito importante lembrar que quanto maior o número de entradas, mais complexa a topologia, muito maior o número de parâmetros a serem ajustados além do tempo necessário ao processamento.

A topologia padrão é a seguinte:

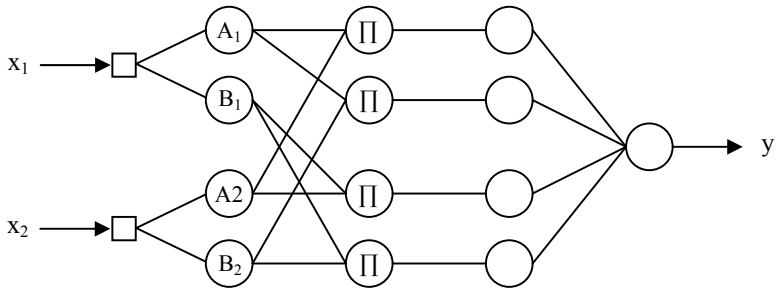


Figura 5.4.1: Topologia padrão para os testes práticos

A partir desta são realizados os testes, como mostrado a seguir.

5.4.1 Resultados Obtidos

Os gráficos a seguir apresentam os resultados dos processos de treinamento e validação realizados com todos os conjuntos de dados, utilizando redes neuro-fuzzy.

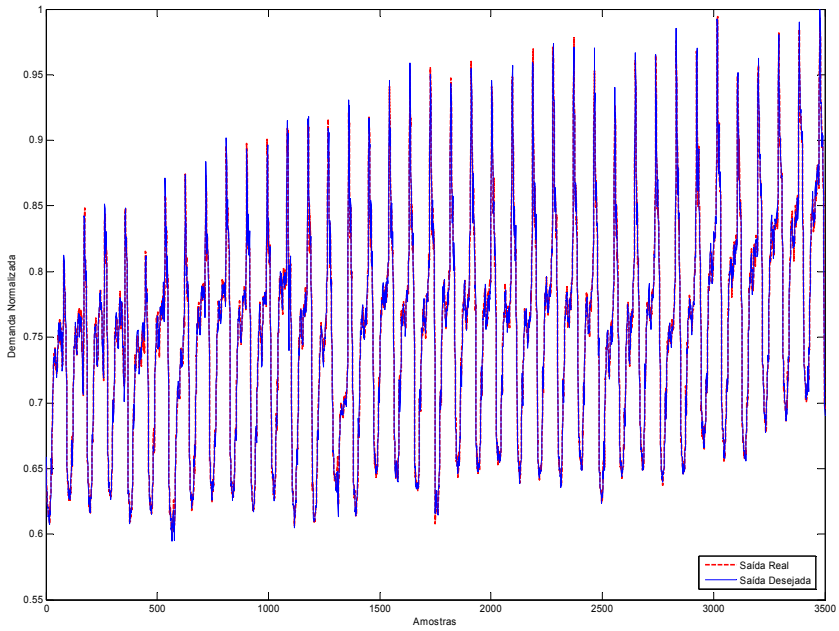


Figura 5.4.1.1: Resultado do processo de treinamento para o primeiro conjunto de dados

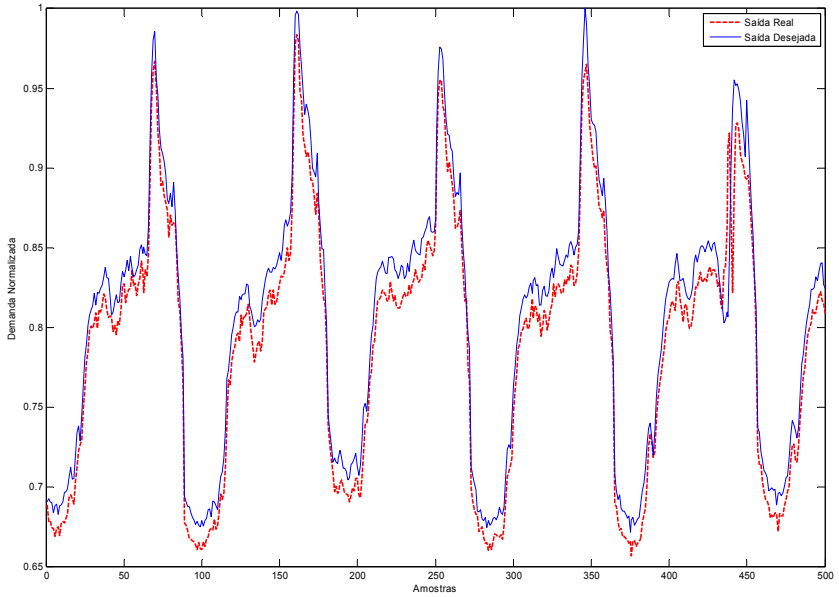


Figura 5.4.1.2: Resultado do processo de validação para o primeiro conjunto de dados

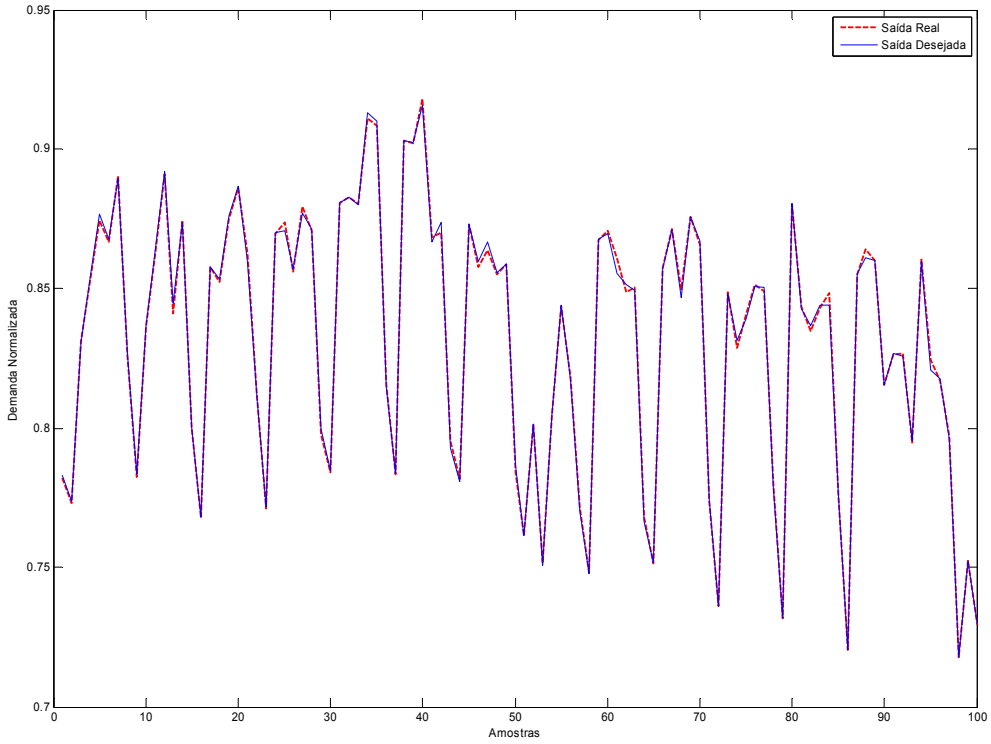


Figura 5.4.1.3: Resultado do processo de treinamento para o segundo conjunto de dados

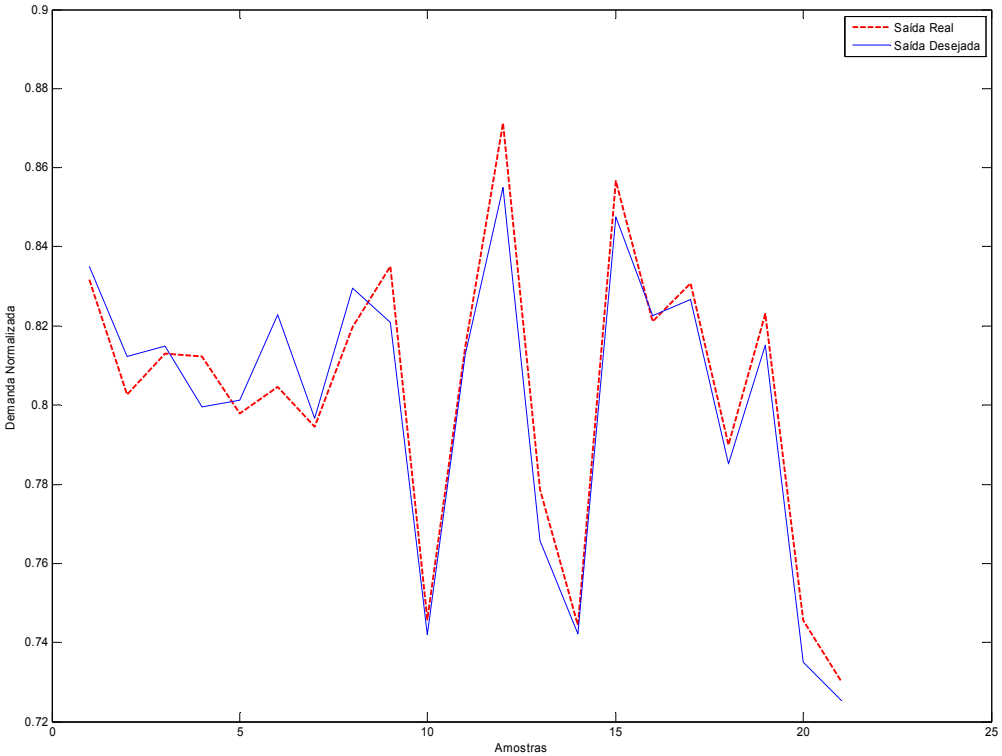


Figura 5.4.1.4: Resultado do processo de validação para o segundo conjunto de dados

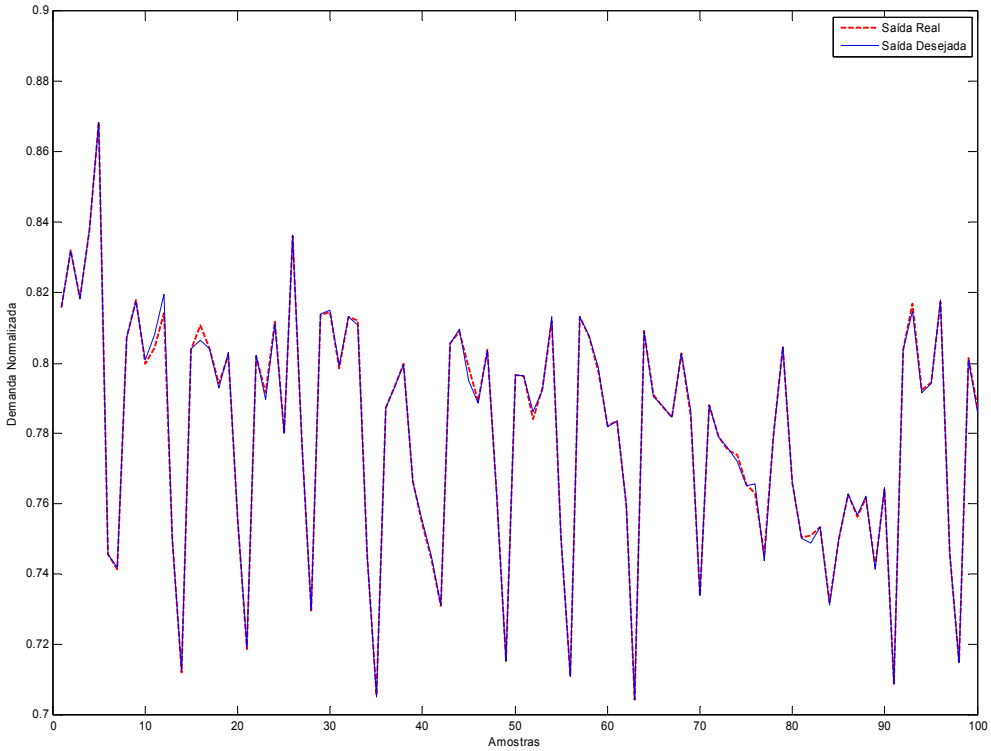


Figura 5.4.1.5: Resultado do processo de treinamento para o terceiro conjunto de dados

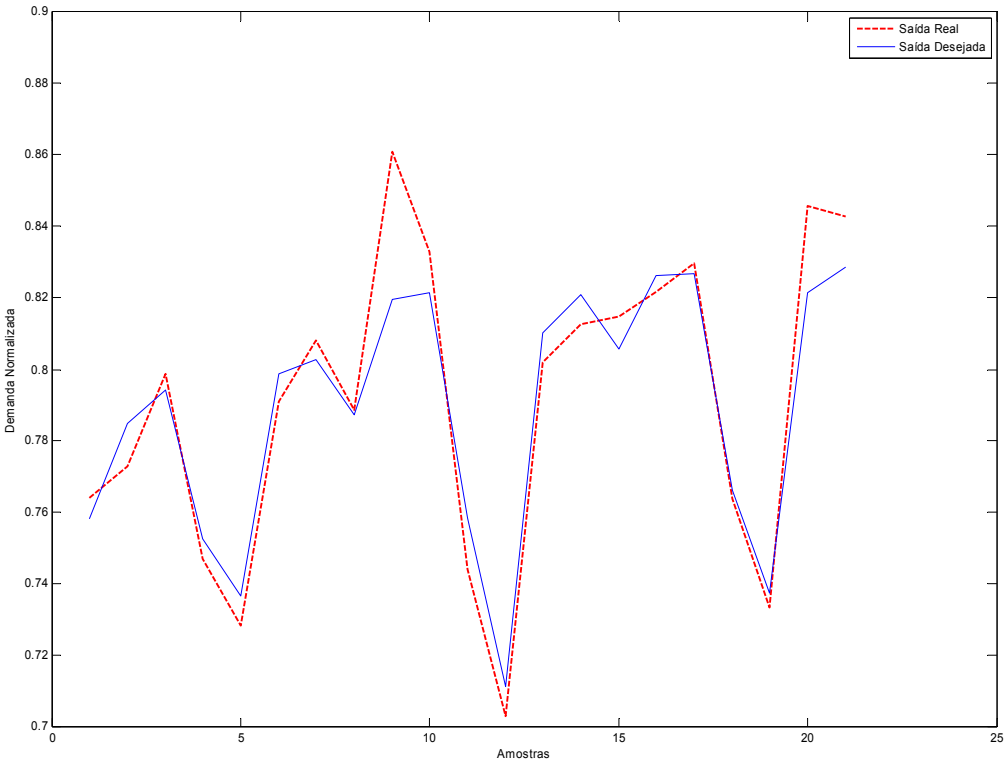


Figura 5.4.1.6: Resultado do processo de validação para o terceiro conjunto de dados

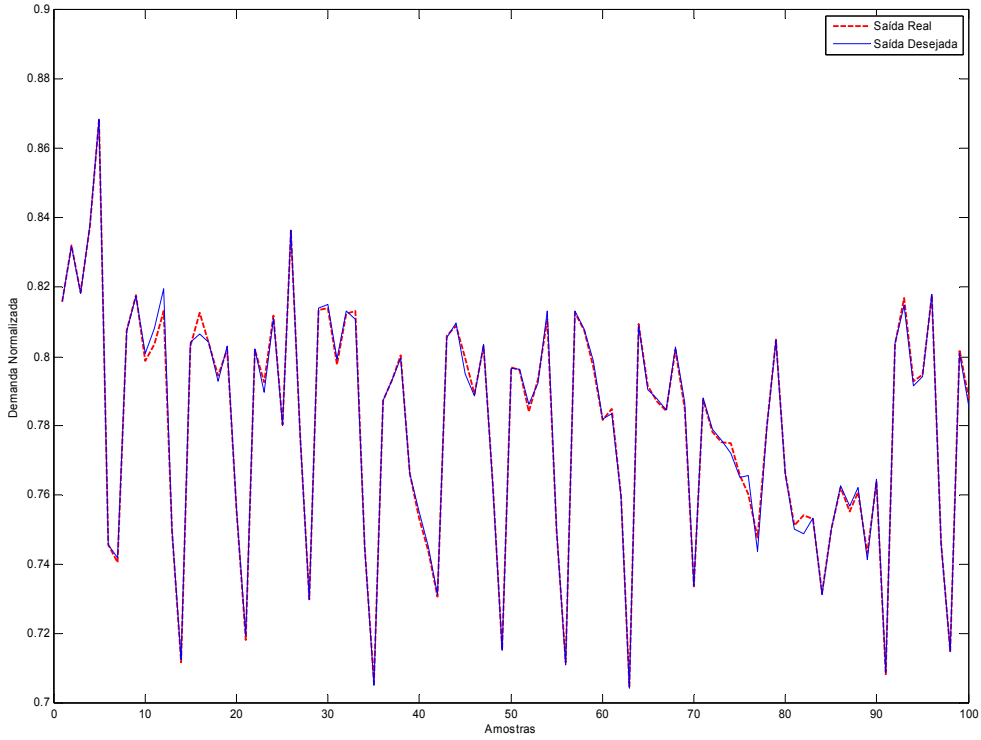


Figura 5.4.1.7: Resultado do processo de treinamento para o quarto conjunto de dados

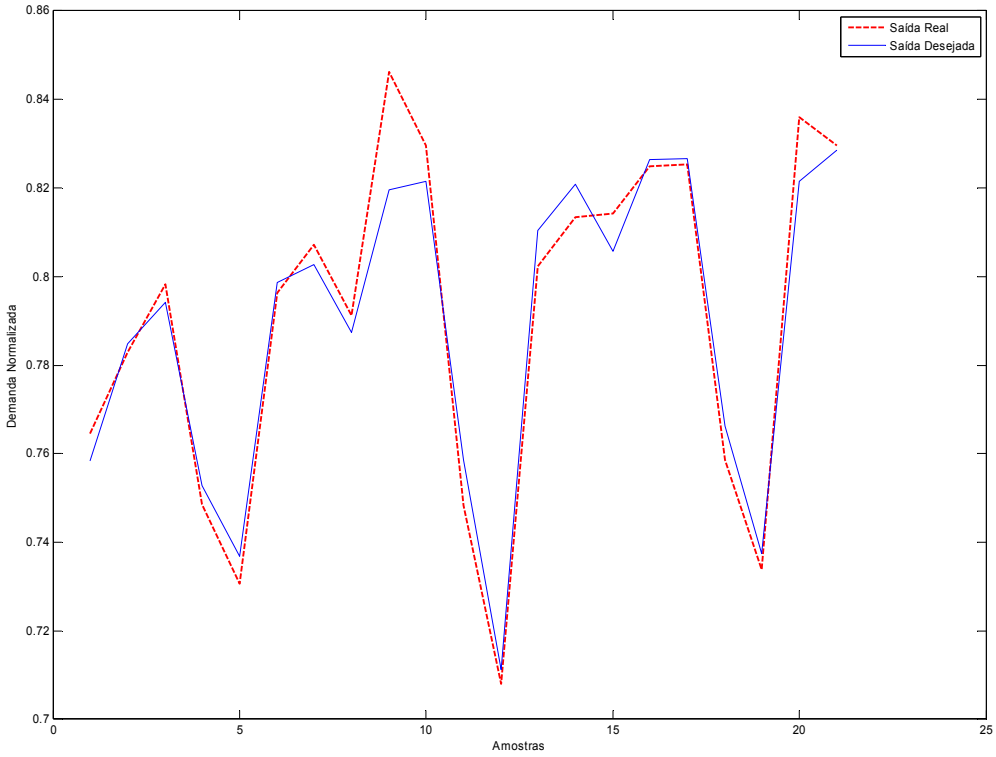


Figura 5.4.1.8: Resultado do processo de validação para o quarto conjunto de dados

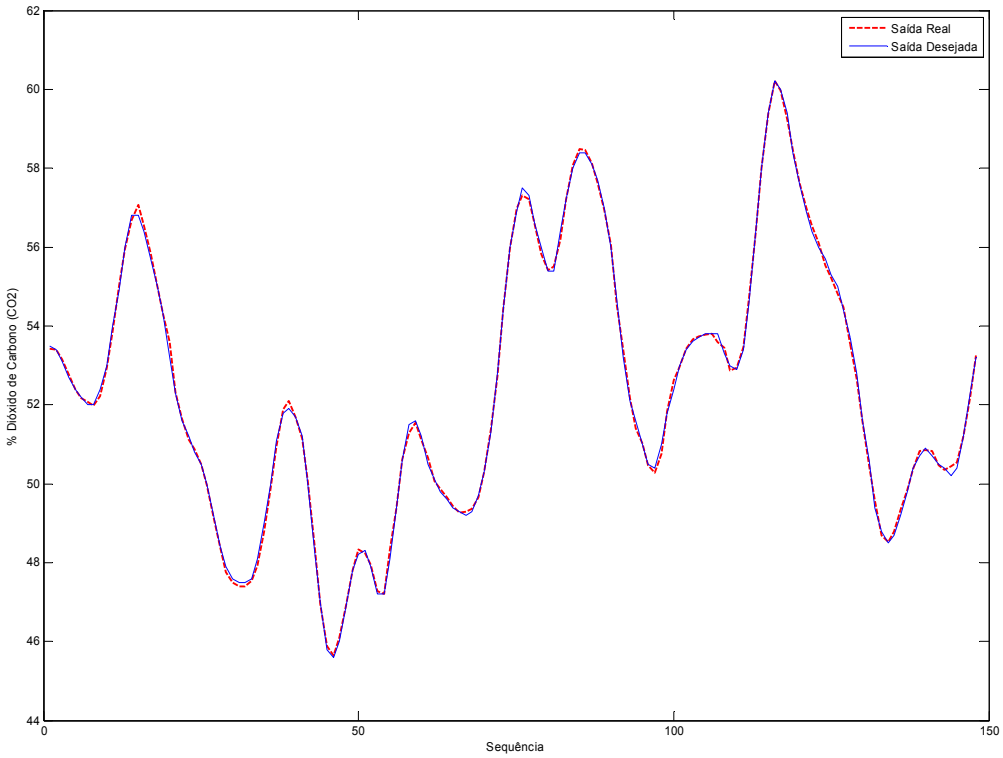


Figura 5.4.1.9: Resultado do processo de treinamento para a série de dados Gas Furnace

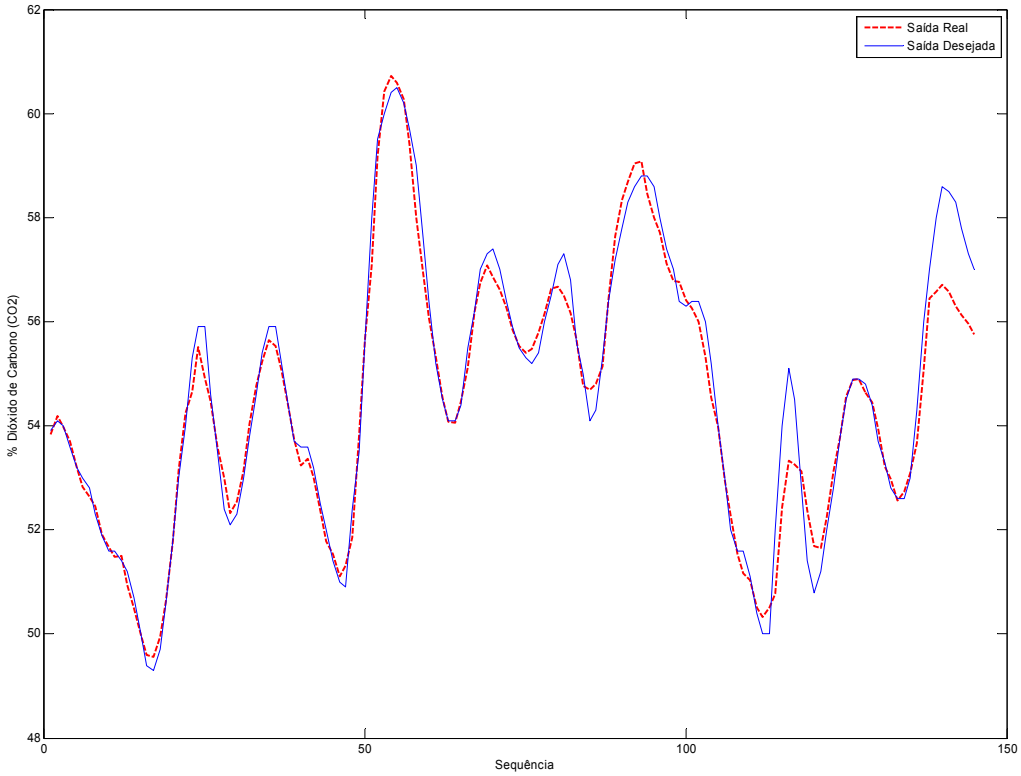


Figura 5.4.1.10: Resultado do processo de validação para a série de dados Gas Furnace

A tabela seguinte mostra os resultados obtidos nos testes realizados anteriormente com redes neuro-fuzzy:

CONJUNTO DE DADOS	$SSE_{TREINAMENTO}$	$SSE_{VALIDAÇÃO}$
1	0.1325	0.1914
2	0.0002	0.0017
3	0.0003	0.0015
4	0.0003	0.0019
Gas Furnace	1.7967	45.0636

Tabela 5.4.1.1: Resultados obtidos com redes neuro-fuzzy, em relação a todos os conjuntos de dados

5.4.2 Avaliação dos Resultados

Pelos resultados apresentados pode-se concluir que as redes do tipo neuro-fuzzy apresentaram resultados satisfatórios na maioria dos casos. O grande inconveniente deste tipo de estrutura é o elevado número de parâmetros a serem ajustados e mais do que isso, o elevadíssimo tempo de processamento. Como comentado, a topologia cresce exponencialmente conforme o número de entradas do sistema. Sendo assim, para caso onde se excede três entradas, as RNF tornam-se inviáveis. Os resultados apresentados só foram conseguidos com funções de pertinência do tipo “constante” nos neurônios de saída. O uso do conseqüente de Sugeno fez com que o treinamento consumisse aproximadamente uma hora de processamento e na maior parte dos casos apresentou resultados indesejáveis.

5.5 Testes realizados com redes neuro-fuzzy-polinomiais

Até o momento foram analisadas estruturas bem definidas e comuns na literatura. As redes neuro-fuzzy-polinomias constituem uma metodologia relativamente nova e, de certa forma, com poucas aplicações publicadas em relação às anteriores.

Espera-se que os resultados por elas produzidos sejam compatíveis com os anteriores, visto que a RNFP agrega estruturas capazes de mapear tanto características lineares quanto não-lineares presentes no conjunto de treinamento.

A topologia adotada para a realização dos testes práticos, já apresentada anteriormente é a seguinte:

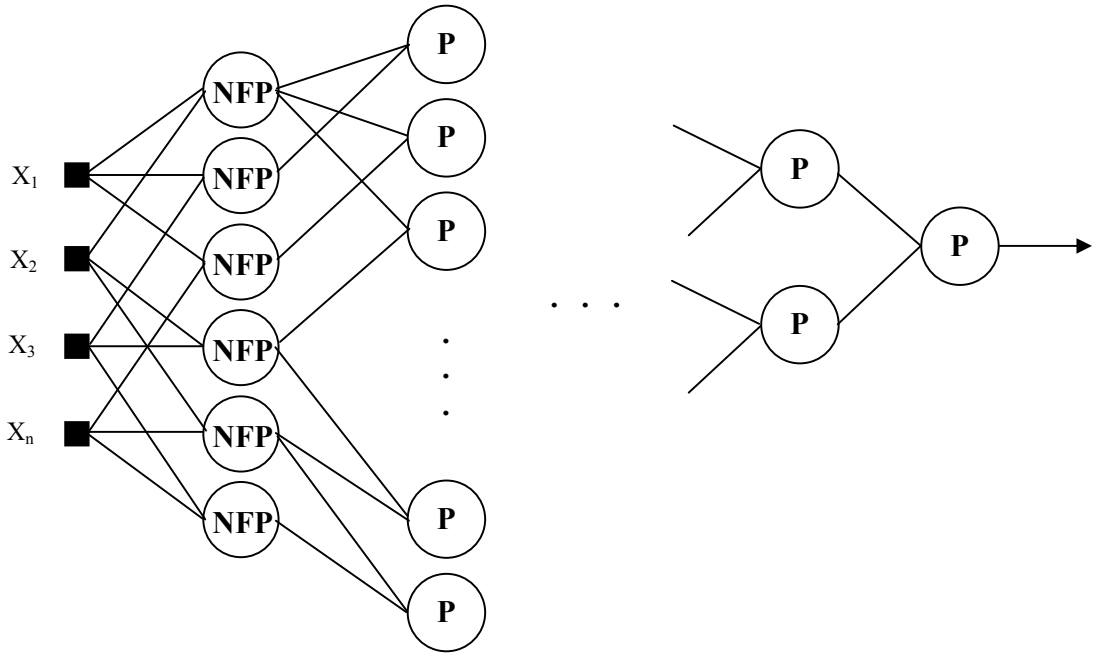


Figura 5.5.1: Topologia adotada como referência para os testes com RNFP

Fixou-se em dois o número máximo de camadas polinomiais.

5.5.1 Resultados Obtidos

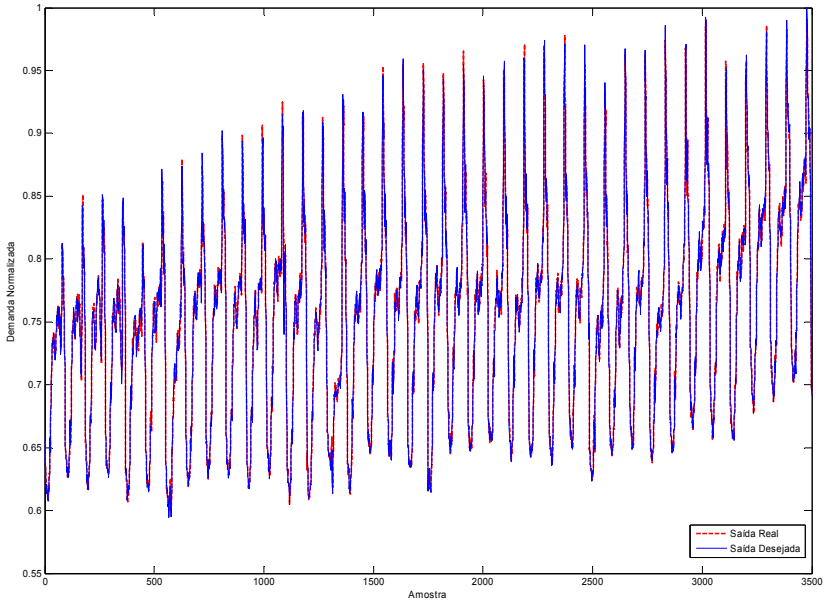


Figura 5.5.1.1: Resultado do processo de treinamento para o primeiro conjunto de dados

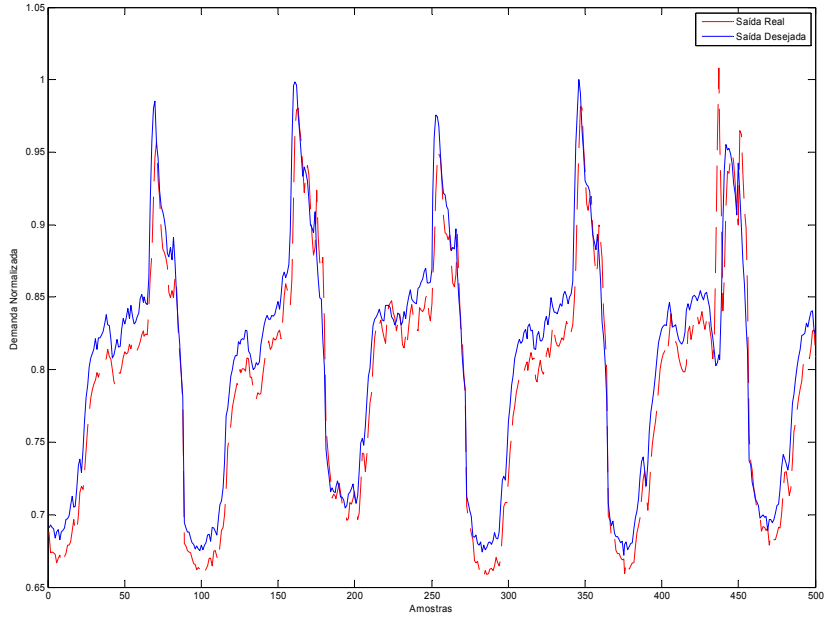


Figura 5.5.1.2: Resultado do processo de validação para o primeiro conjunto de dados

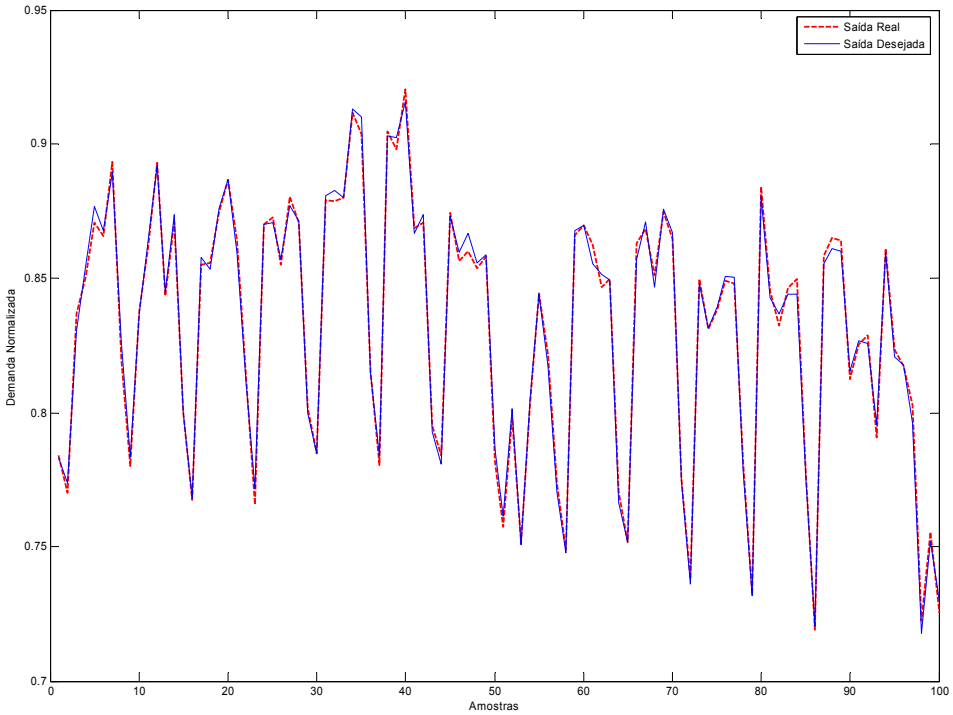


Figura 5.5.1.3: Resultado do processo de treinamento para o segundo conjunto de dados

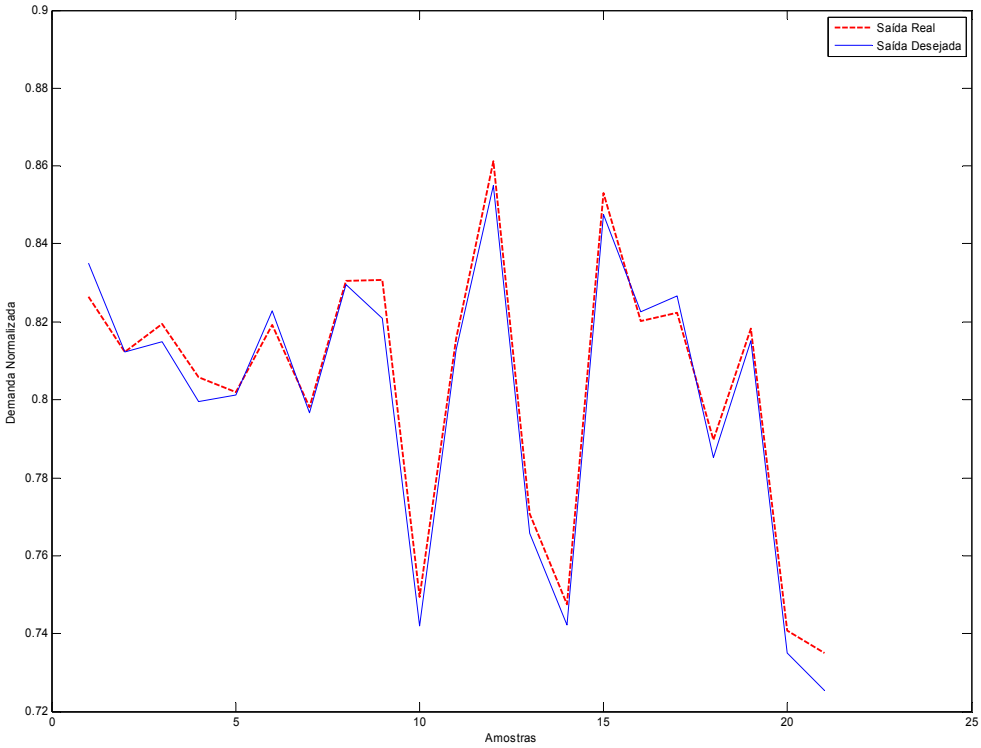


Figura 5.5.1.4: Resultado do processo de validação para o segundo conjunto de dados

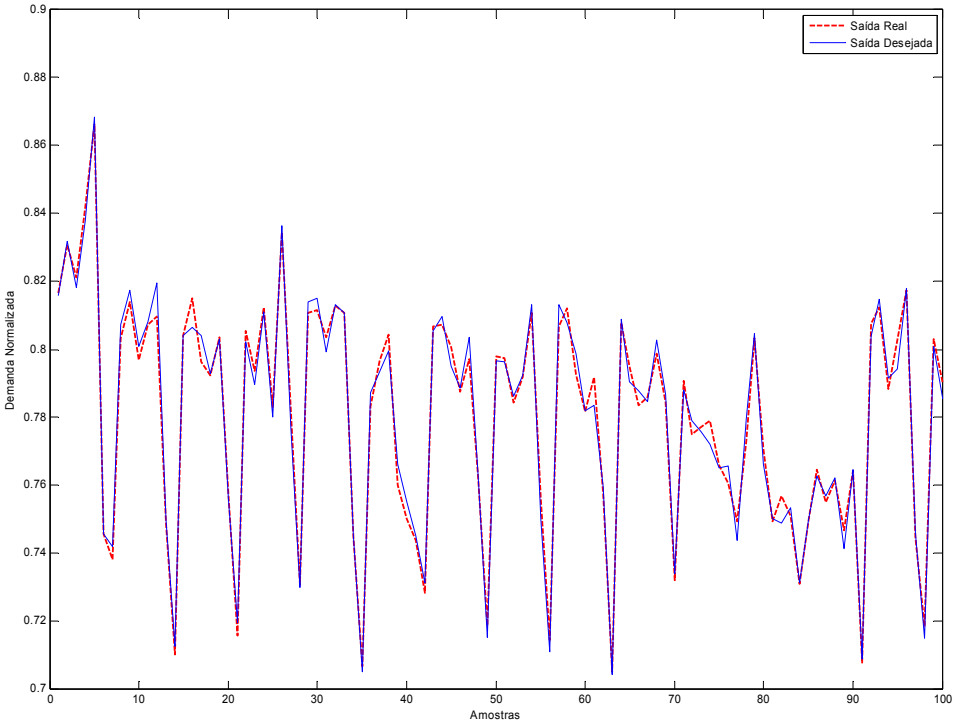


Figura 5.5.1.5: Resultado do processo de treinamento para o terceiro conjunto de dados

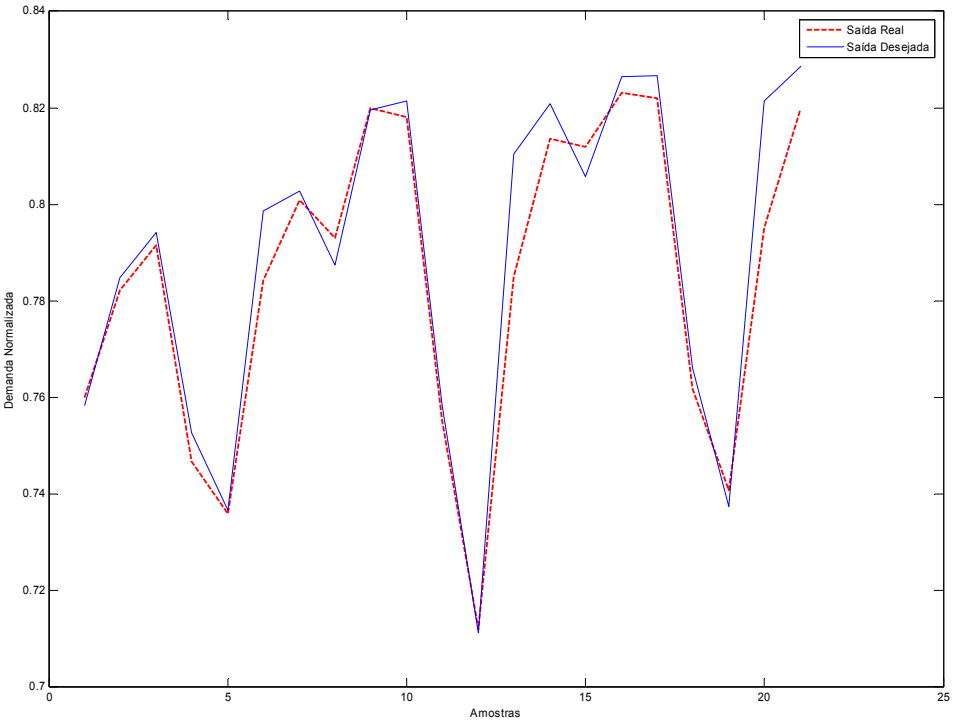


Figura 5.5.1.6: Resultado do processo de validação para o terceiro conjunto de dados

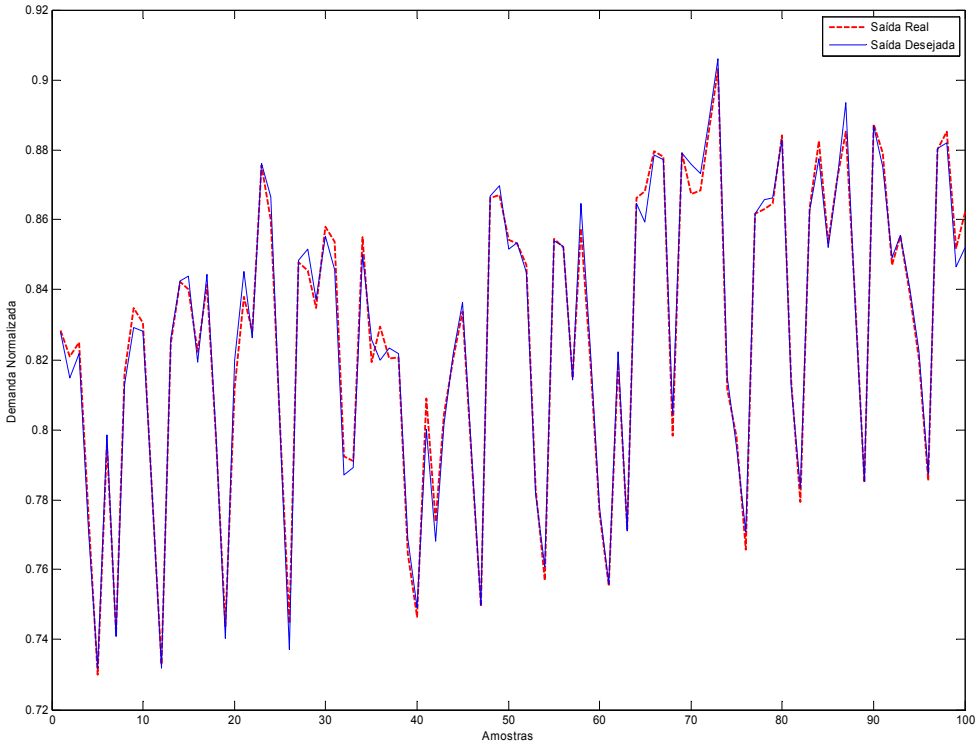


Figura 5.5.1.7: Resultado do processo de treinamento para o quarto conjunto de dados

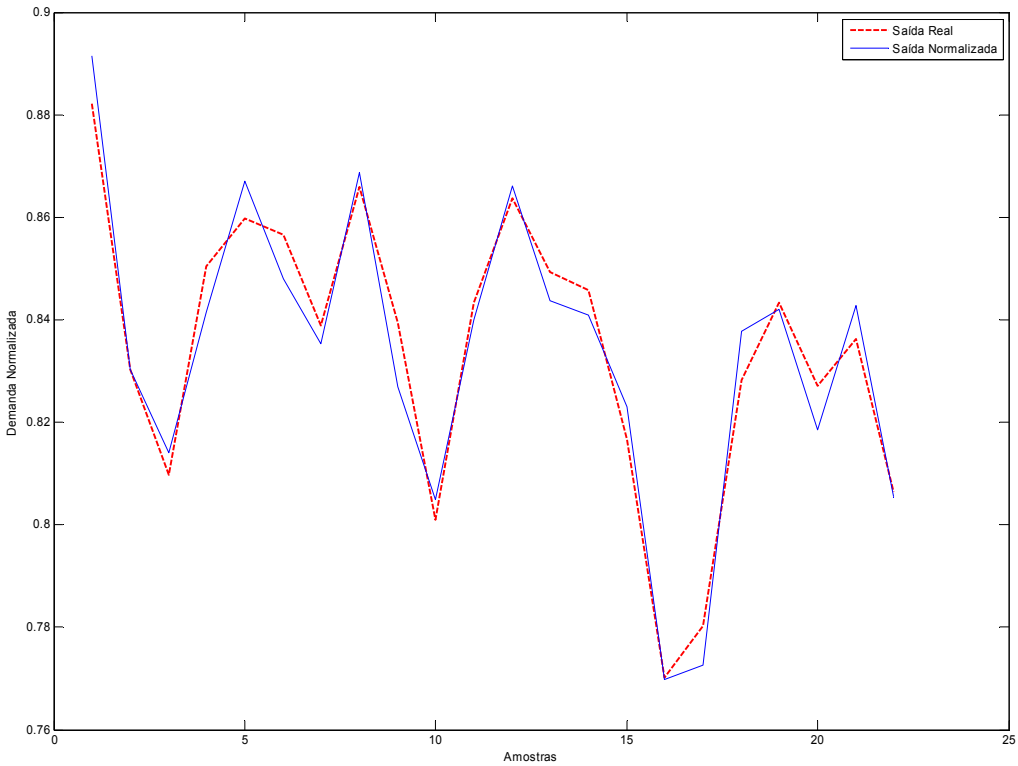


Figura 5.5.1.8: Resultado do processo de validação para o quarto conjunto de dados

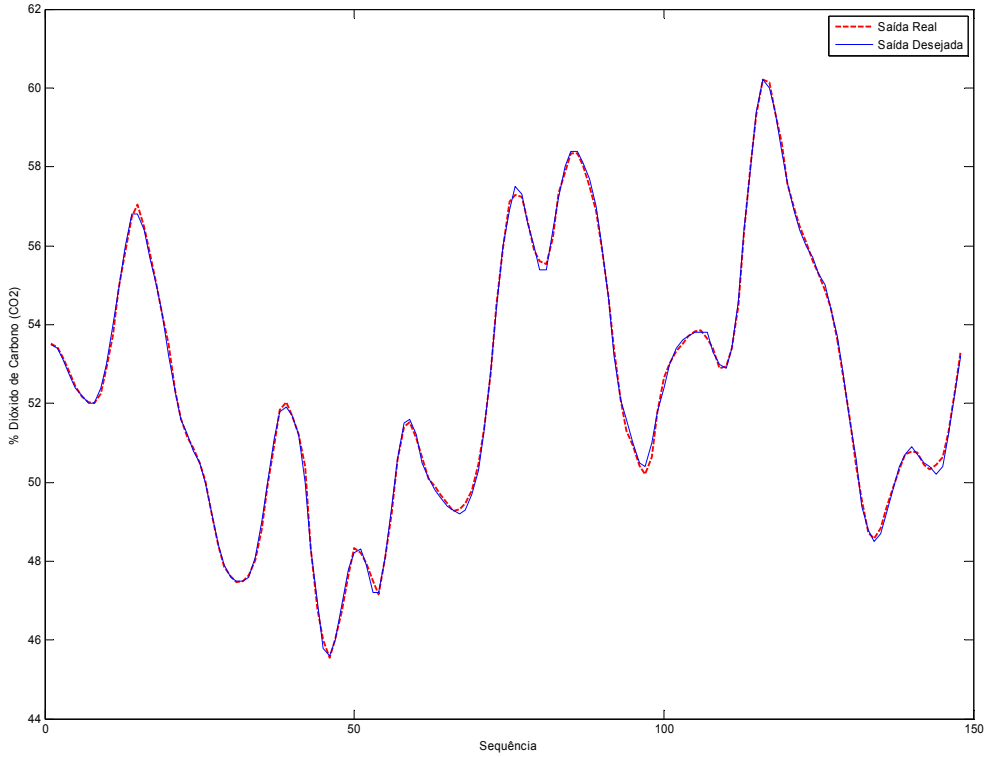


Figura 5.5.1.9: Resultado do processo de treinamento para a série de dados Gas Furnace

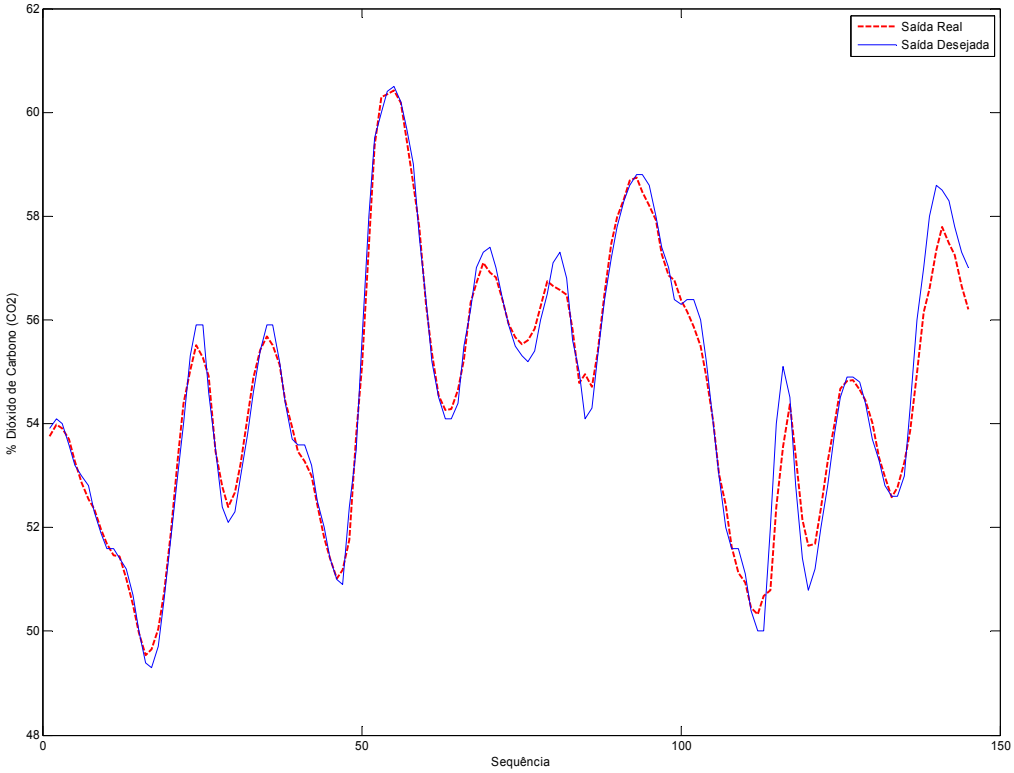


Figura 5.5.1.10: Resultado do processo de validação para a série de dados Gas Furnace

A tabela seguinte mostra os resultados obtidos nos testes realizados anteriormente com redes neuro-fuzzy-polinomiais:

CONJUNTO DE DADOS	$SSE_{TREINAMENTO}$	$SSE_{VALIDAÇÃO}$
1	0,2585	0,3898
2	0,0010	5,7918e-004
3	0,0014	0,0013
4	0,0016	8.8465e-004
Gas Furnace	2,2284	26,4209

Tabela 5.5.1.1: Resultados obtidos com redes neuro-fuzzy, em relação a todos os conjuntos de dados

5.5.2 Avaliação dos Resultados

Conforme explicitado no início dos testes, esperava-se que os resultados fossem próximos daqueles obtidos nos testes anteriores e isto se verificou.

Os resultados obtidos foram muito bons em todos os casos. Isto comprova o sucesso na associação de redes neuro-fuzzy com redes polinomiais. A grande vantagem é a geração de uma estrutura compacta e capaz a responder a diversos tipos de não-linearidades. Por outro lado apresentam as desvantagens inerentes às RNF e RNP, como o elevado número de parâmetros a serem ajustados e o desconhecimento inicial do número ótimo de camadas, muito menos a quantidade de neurônios em cada uma delas.

5.6 Testes realizados com redes neuro-fuzzy-polinomiais otimizadas via AG

A idéia de associar algoritmos genéticos (AG) às RNFP provém da dificuldade de ajustar corretamente todos os parâmetros que envolvem estas estruturas. Para comprovar a viabilidade da hibridização proposta são realizados testes assim como nos casos anteriores.

Para simplificação do AG adotou-se um modelo de cromossomo simplificado, contendo apenas informações sobre os parâmetros mais importantes, conforme ilustrado a seguir:

N1	N1	N1	N2	N2	FP	FP	NC	NC	GP1	GP2

Onde:

N1 = número de neurônios selecionados para a camada 1;

N2 = número de neurônios selecionados para a camada 2;

FP = tipo de função de pertinência;

NC = número de conjuntos fuzzy por variável;

GP1 = grau do polinômio na camada polinomial 1;

GP2 = grau do polinômio na camada polinomial 2.

É importante ressaltar que esta representação foi adotada apenas para efeitos de avaliação da metodologia proposta e que uma alternativa é a de incluir no cromossomo todos os parâmetros (citados no item 7.1) a serem ajustados. Isto poderá produzir melhores resultados.

As simulações executadas utilizaram as seguintes configurações para o AG:

- número de indivíduos = 20;

- tipo de cruzamento = único corte;

- taxa de cruzamento = 0.85;

- taxa de mutação = 0.04;

- número de gerações = 100.

5.6.1 Resultados Obtidos

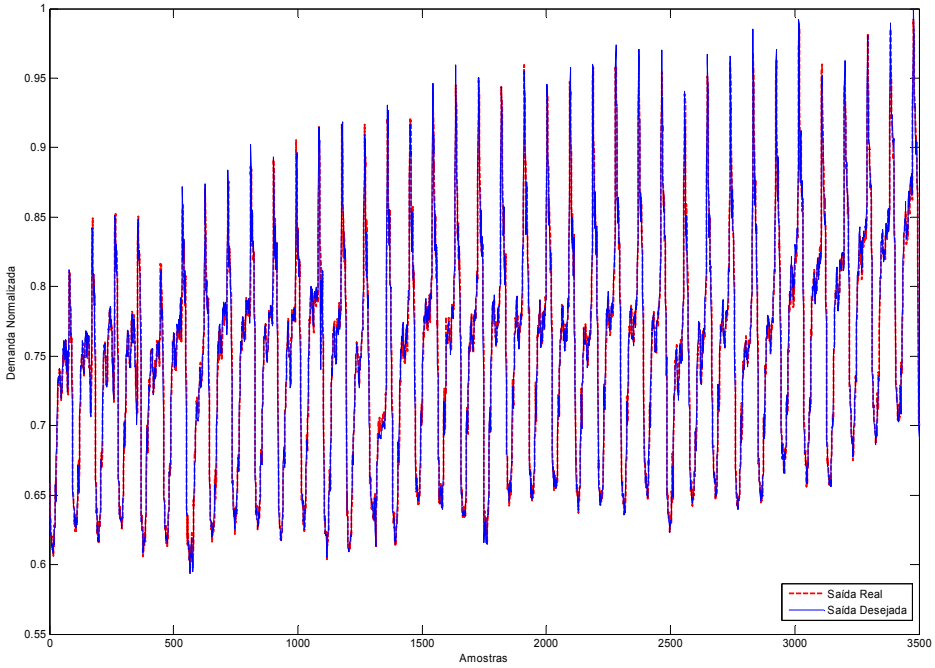


Figura 5.6.1.1: Resultado do processo de treinamento para o primeiro conjunto de dados

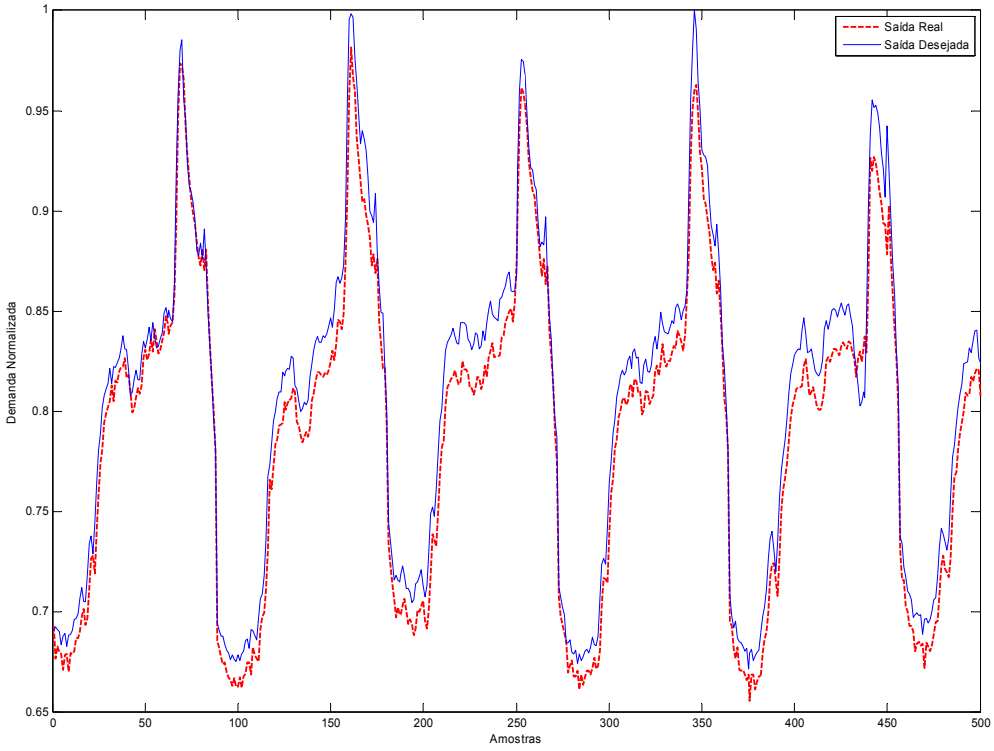


Figura 5.6.1.2: Resultado do processo de validação para o primeiro conjunto de dados

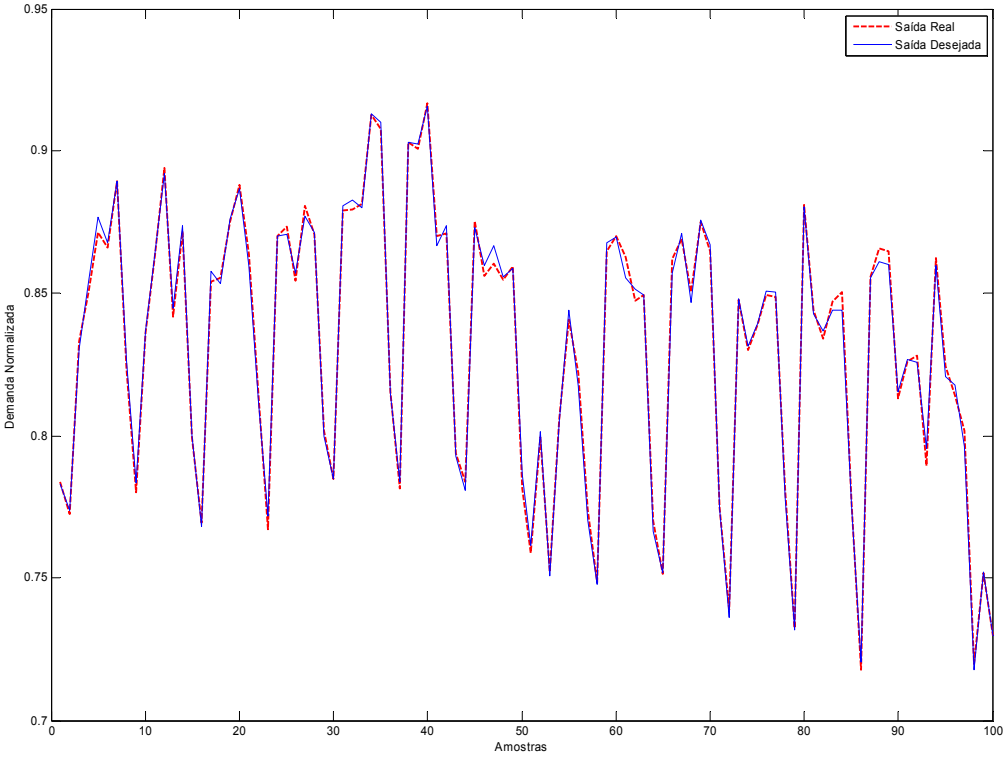


Figura 5.6.1.3: Resultado do processo de treinamento para o segundo conjunto de dados

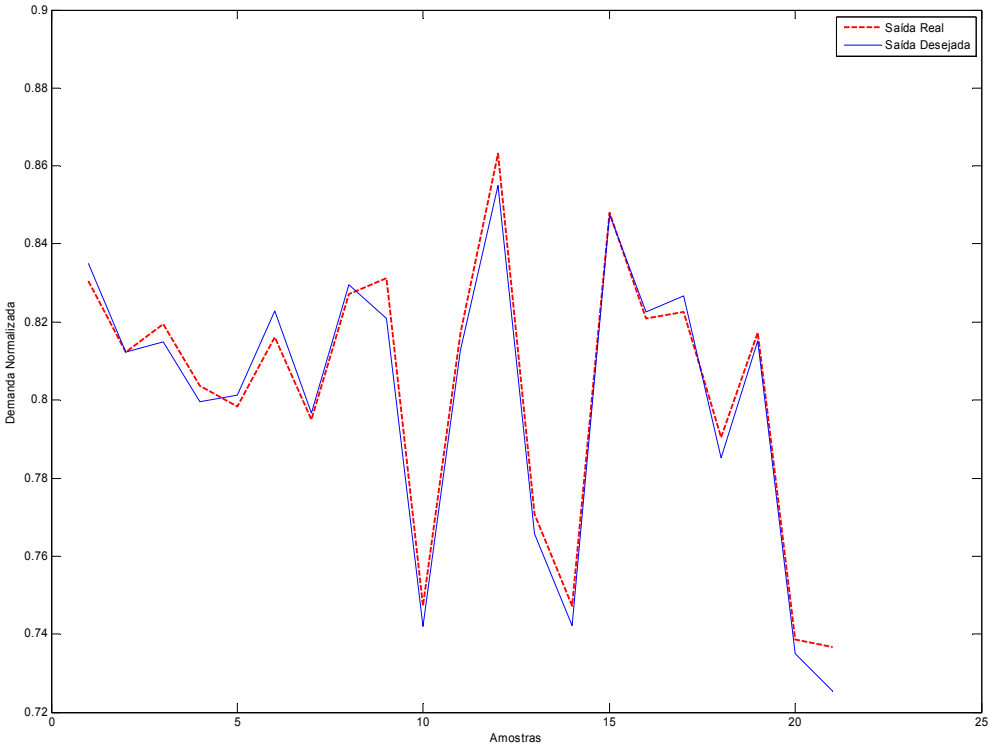


Figura 5.6.1.4: Resultado do processo de validação para o segundo conjunto de dados

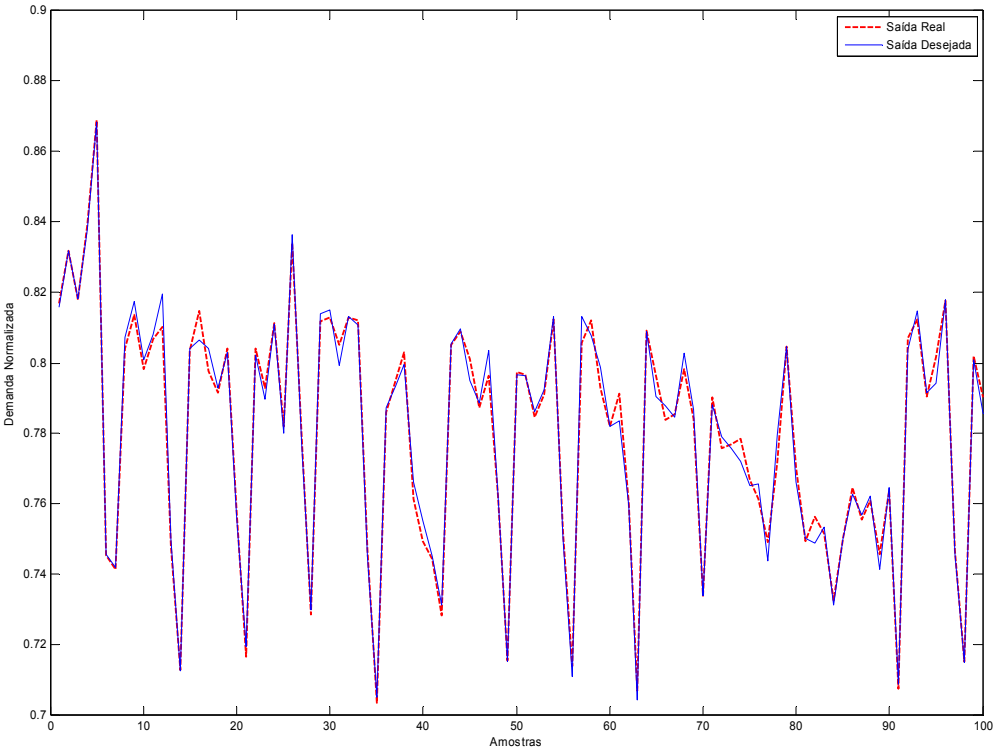


Figura 5.6.1.5: Resultado do processo de treinamento para o terceiro conjunto de dados

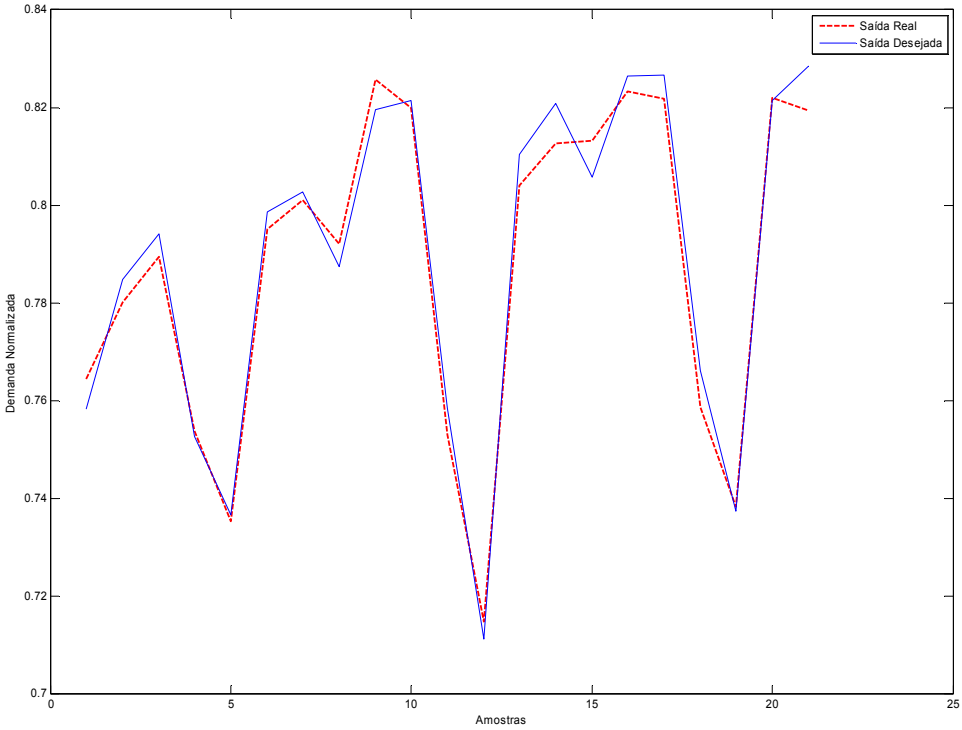


Figura 5.6.1.6: Resultado do processo de validação para o terceiro conjunto de dados

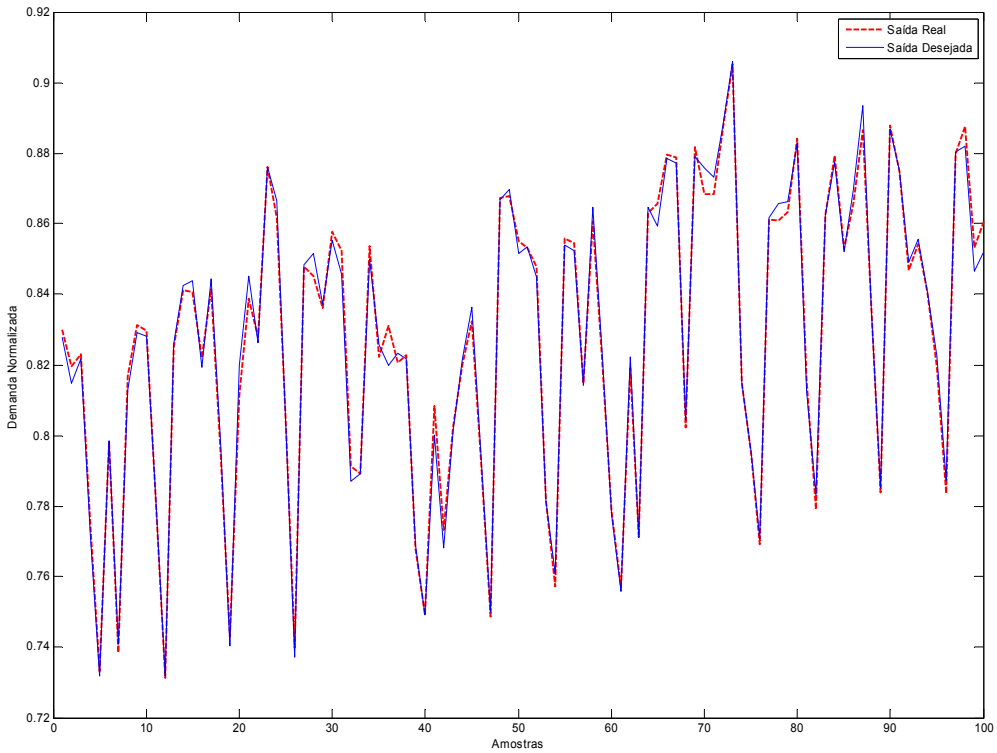


Figura 5.6.1.7: Resultado do processo de treinamento para o quarto conjunto de dados

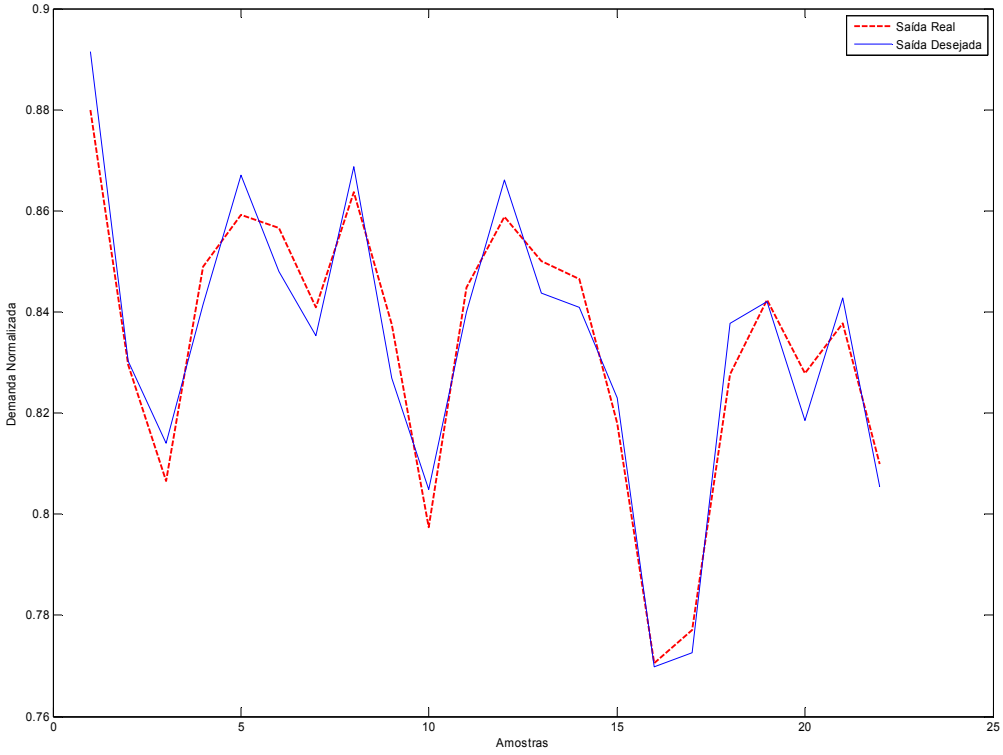


Figura 5.6.1.8: Resultado do processo de validação para o quarto conjunto de dados

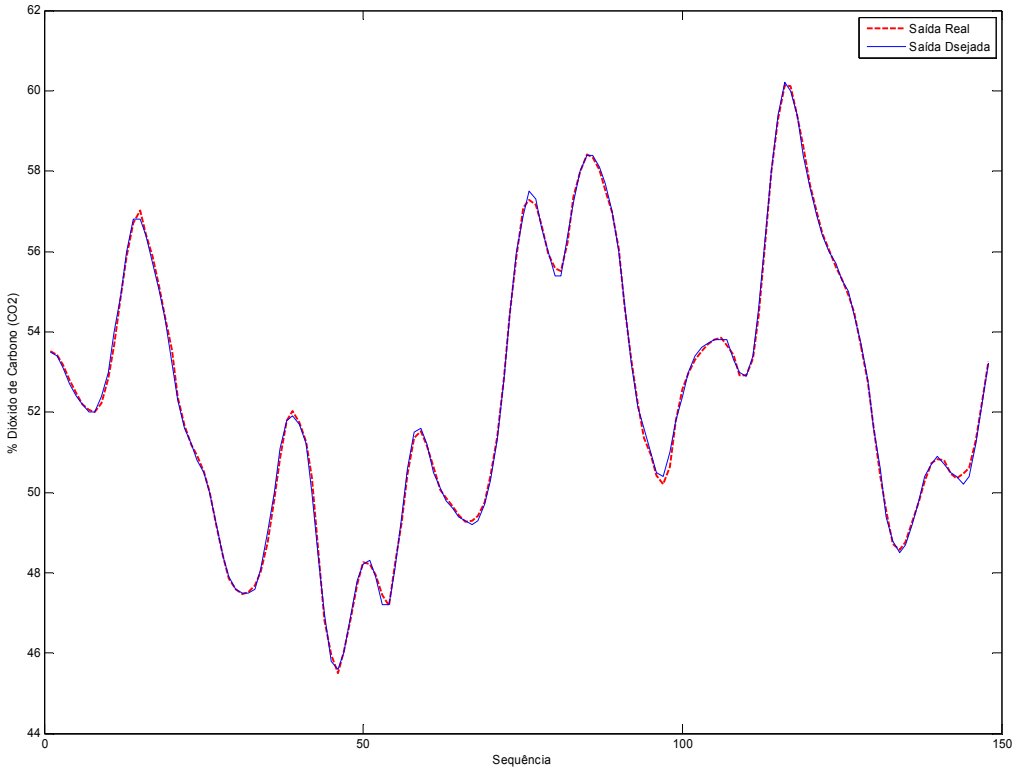


Figura 5.6.1.9: Resultado do processo de treinamento para a série de dados Gas Furnace

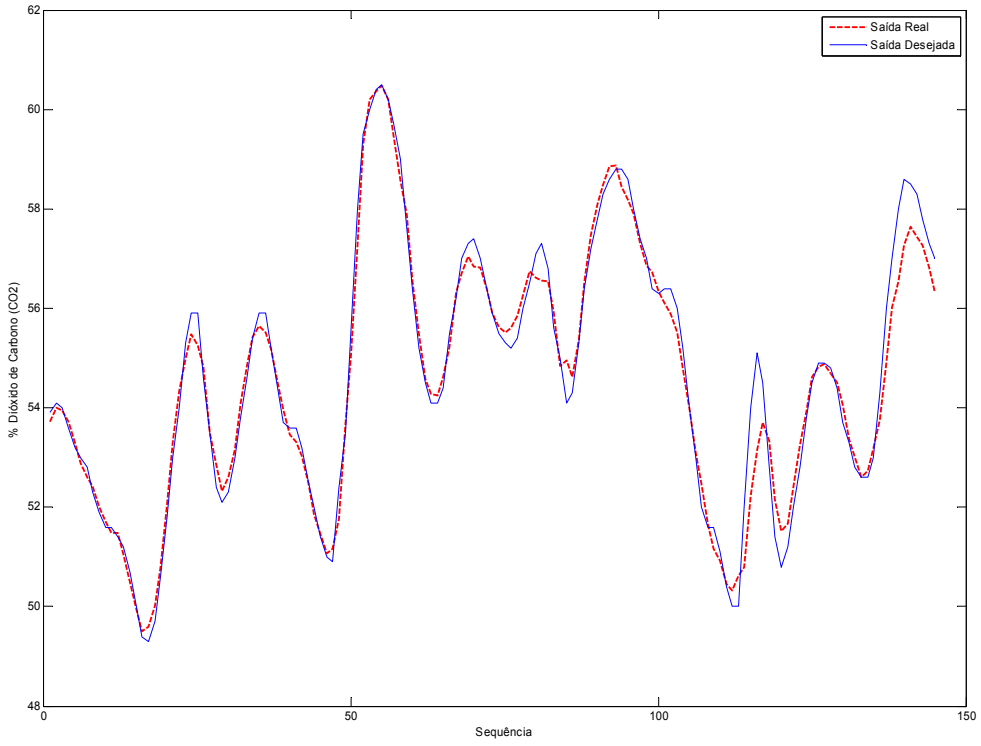


Figura 5.6.1.10: Resultado do processo de validação para a série de dados Gas Furnace

A tabela seguinte mostra os resultados obtidos nos testes realizados com redes neuro-fuzzy-polinomiais otimizadas por AG:

CONJUNTO DE DADOS	$SSE_{TREINAMENTO}$	$SSE_{VALIDAÇÃO}$
1	0.2570	0.1584
2	7.9236E-004	5.7492e-004
3	0.0011	5.4575E-004
4	0.0012	0.0010
Gas Furnace	2.2177	29.8329

Tabela 5.6.1.1: Resultados obtidos com redes neuro-fuzzy polinomiais, em relação a todos os conjuntos de dados

5.6.2 Avaliação dos Resultados

Pelos resultados obtidos pode-se avaliar como válida a idéia da utilização de algoritmos genéticos no auxílio ao processo de treinamento de redes neuro-fuzzy-polinomiais.

Aparentemente não há problemas quanto aos resultados apresentados, porém os mesmos encobertam uma informação importante. Na grande maioria dos testes o algoritmo considerou como promissores apenas os cromossomos que continham nível lógico alto nos primeiros bits. Isto tem sentido e confirma o fato de que um número elevado de neurônios nas camadas polinomiais pode conduzir a bons resultados. Por outro lado foi percebido também, nos testes com RNFP com ajuste manual, que uma pequena quantidade de neurônios nestas camadas não compromete de forma significativa o resultado final. Percebeu-se também que o ajuste nos parâmetros referentes às funções de pertinência provoca modificações muito mais interessantes. Isto quer dizer que o sucesso no processo de treinamento está muito mais associado a ajustes nas funções de pertinência do que no na topologia.

Outro grande inconveniente é o tempo necessário ao treinamento e o fato de que nem sempre há a convergência para o indivíduo ótimo. Isto irá depender diretamente dos ajustes iniciais que devem ser feitos para a execução do AG. Daí a necessidade de avaliação de novos

métodos de otimização de forma a reduzir o tempo de treinamento e tempo de processamento sem comprometimento dos resultados finais.

Sendo assim, optou-se pelo estudo da técnica conhecida por *PSO* (*Particle Swarm Optimization*), ou otimização por enxame de partículas, que será apresentada em detalhes no capítulo seguinte.

CAPÍTULO 6
ALGORITMOS DE ENXAME

Muito recentemente, diversas técnicas da chamada computação evolucionária, com características similares tem sido propostas. Estas se baseiam no comportamento de animais, como bando de pássaros, enxames de abelhas e colônias de formigas. Dentre elas, a de maior destaque é conhecida por otimização por enxame de partículas (OEP), que vem sendo aplicadas em diversos tipos de problemas de otimização, com a geração de resultados satisfatórios em tempos de processamento menores do que os AG.

6.1 Introdução

O elevado nível de evolução dos sistemas de processamento de dados tem permitido o desenvolvimento e aplicação de diversas técnicas de Inteligência Artificial (IA) na solução de problemas complexos.

Dentre as sub-áreas da IA pode-se destacar aquela conhecida como Computação Evolucionária (CE), que compreende um conjunto de técnicas, inspiradas na natureza e aplicadas à otimização.

Em muitos casos o modelo matemático que representa o comportamento de uma ou mais variáveis, pode conduzir a funções descontínuas, não-lineares e complexas.

O propósito da criação destes modelos é justamente o de avaliar as condições em que estas assumem valores extremos, de mínimo ou de máximo.

Desta forma, a CE disponibiliza várias técnicas que tentam esboçar características como raciocínio, percepção, adaptação e evolução, com intuito de buscar um conjunto ótimo de valores que satisfaçam as restrições impostas.

Dentre estas técnicas pode-se citar: Algoritmos Genéticos (AG), Recozimento Simulado (RS) e Otimização por Enxame de Partículas (OEP). Todas estas técnicas são classificadas como heurísticas, pois partem de um conjunto de possíveis soluções, atribuídas aleatoriamente e, a partir destas, utilizam um mecanismo capaz de explorar o espaço de busca atrás de um valor ótimo ou sub-ótimo.

Os algoritmos genéticos correspondem a um método de busca, baseado na teoria da evolução e seleção natural de Charles Darwin e Gregor Mendel.

Um conjunto inicial de soluções candidatas, denominadas indivíduos, é gerado aleatoriamente e posteriormente avaliado. Os melhores indivíduos correspondem aos pontos localizados em região promissora no espaço de busca. Estes são submetidos a operadores genéticos que tentam conduzi-los a locais cada vez melhores. O processo é repetido, iterativamente, até que seja encontrado o valor ideal ou até que um número pré-determinado de gerações seja atingido.

O Recozimento Simulado (RC) é um método de otimização baseado no processo físico de têmpera dos materiais. O método parte de uma solução inicial S_o que representa uma amostra de partículas de um dado material. O parâmetro de controle temperatura T é inicializado com um valor alto. A partir daí a temperatura é reduzida lentamente a cada etapa até que se atinja o equilíbrio térmico, a um baixo valor de T. Ao longo do processo, procura-se atingir o equilíbrio térmico através de novas soluções candidatas, vizinhas, geradas por um tipo de perturbação sobre as partículas atuais. A perturbação provoca o movimento das partículas pelo espaço de busca. A aceitação ou não da nova posição provocada pela perturbação, está condicionada ao custo da solução corrente, ao custo da nova solução e ao valor do parâmetro T.

A técnica de otimização por enxame de partículas (OEP) é baseada no comportamento e movimento de um grupo de animais. A busca é realizada pela troca de informações entre componentes de um grupo que busca a região ótima dentro do espaço de soluções.

Por ser um método comprovadamente robusto, eficiente e de simples implementação, o mesmo foi escolhido para o estudo e associação com outras técnicas de IA, conforme proposta deste trabalho de tese. Maiores detalhes do método, assim como a hibridização proposta são apresentados a seguir.

6.2 Otimização por enxame de partículas

A otimização por enxame de partículas, introduzida por Kennedy e Heberhart [9], é uma técnica de busca heurística, baseada no movimento de um determinado grupo (enxame) de animais (partículas) de mesma espécie ao longo de uma região. O movimento é orientado pela interação entre os membros de um grupo.

O embasamento biológico parte de observações de bandos de pássaros e cardumes em busca de alimento. A partir destas, é possível constatar que o comportamento de todo grupo é resultado da experiência individual acumulada de cada membro associada à experiência acumulada pelo grupo.

Assim como em outros métodos que compõe a CE, cada indivíduo ou partícula representa uma solução candidata e esta corresponde a um ponto no espaço de busca.

No algoritmo de OEP, cada partícula tem um valor associado de forma a indicar sua "proximidade" com relação à solução, e uma velocidade que define como a mesma irá se movimentar.

Ao longo do processo, cada partícula modifica sua velocidade de acordo com a melhor posição encontrada por si mesma e por todo o grupo. Estes dois mecanismos de exploração do

espaço estão associados ao componente cognitivo, que representa a experiência individual das partículas e ao componente social, que representa a experiência de todo o grupo.

Ao longo do tempo, cada uma modifica sua velocidade, levando em conta a informação das melhores posições individuais e do grupo e desta forma acabam encontrando o alimento.

6.3 Descrição do Método

Conforme dito anteriormente, o OEP é um método de otimização capaz de minimizar funções contínuas através de um conjunto de soluções candidatas que interagem entre si e utilizam procedimentos determinísticos para a exploração do espaço de busca atrás da solução ótima.

Assim como os outros métodos de CE que realizam esta função, o OEP inicia seu processo de busca através de um conjunto de partículas (enxame) com posições e velocidades atribuídas aleatoriamente.

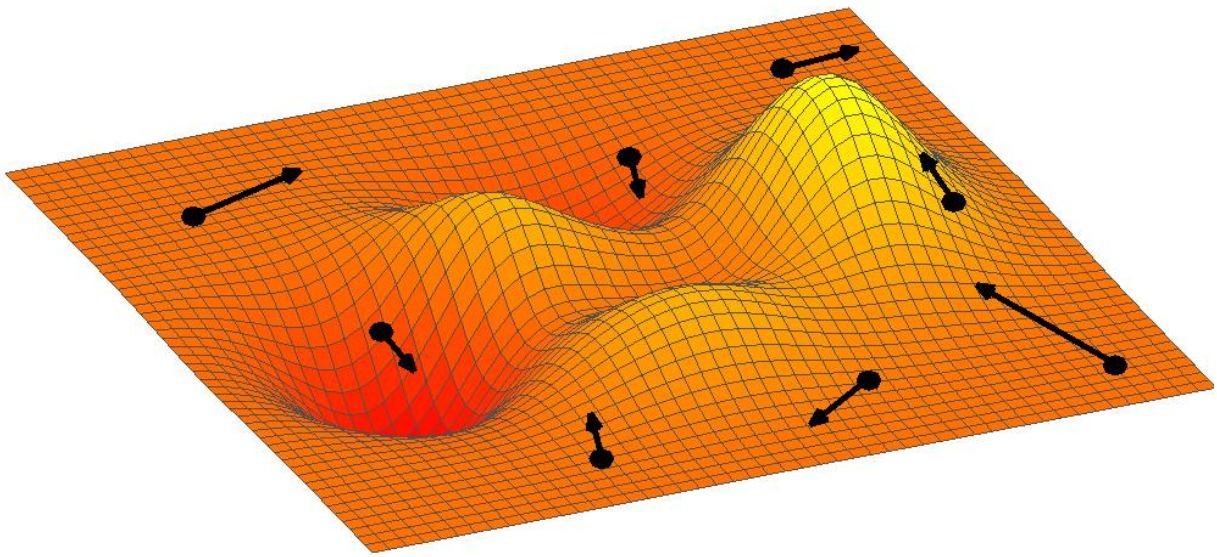


Figura 6.3.1: Localização e orientação das partículas no espaço de busca

As partículas ilustradas acima se deslocam de uma posição para outra, em um espaço multidimensional, através das seguintes equações, na forma canônica:

$$\vec{v}^{k+1} = \vec{v}_k + \vec{b}_1 \otimes \vec{r}_1 \otimes (\vec{p}_1 - \vec{x}_k) + \vec{b}_2 \otimes \vec{r}_2 \otimes (\vec{p}_2 - \vec{x}_k) \qquad (1)$$

$$\vec{x}^{k+1} = \vec{c} \otimes \vec{x}_k + \vec{d} \otimes \vec{v}_{k+1} \qquad (2)$$

Onde:

- $\vec{x}, \vec{v}$ são vetores que representam, respectivamente, a posição e velocidade de todas as partículas;

- $\vec{p}_1, \vec{p}_2$ representam as melhores posições já encontradas por cada indivíduo e por todo o grupo. Estas variáveis guardam informações sobre comportamento individual e coletivo;

- $\vec{b}_1, \vec{b}_2$ são coeficientes que representam a força de atração entre as partículas. Também chamados de taxas de aprendizado, os mesmos representam os parâmetros cognitivo e social;

- $\vec{r}_1, \vec{r}_2$ são números aleatórios uniformemente distribuídos no intervalo [0,1];

- $\vec{a}$ é o chamado fator de momento, responsável por ponderar a inércia da partícula;

- $\vec{c}, \vec{d}$ são vetores coeficientes.

Pela análise das equações é possível perceber que a partícula se movimenta em diferentes dimensões e que seu movimento em uma determinada dimensão é atualizado independente das outras. Desta forma, pode-se considerar um espaço unidimensional como caso geral.

Isto permite que sejam reescritas da seguinte forma:

$$v^{k+1} = v_k + b_1 r_1 (p_1 - x_k) + b_2 r_2 (p_2 - x_k) \qquad (3)$$

$$x^{k+1} = c x_k + d v_{k+1} \qquad (4)$$

As equações acima determinam como as velocidades e as posições das partículas serão atualizadas a cada iteração *k*.

Como *p1* representa a melhor posição encontrada pela partícula, pode-se perceber claramente pela equação (3) que o termo $(p_1 - x_k)$ representa o componente cognitivo e que o termo $(p_2 - x_k)$ o componente social, já que p_2 representa a melhor posição do grupo.

A ilustração a seguir mostra o deslocamento de uma partícula através da interação entre os componentes citados:

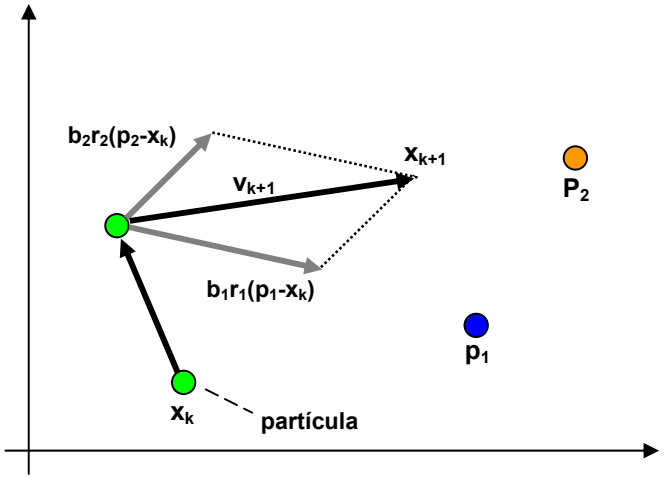


Figura 6.3.2: Interpretação geométrica do deslocamento das partículas

Podemos perceber que quando associamos o comportamento individual (local) ao social (global) temos um conjunto de partículas se deslocando para a posição ótima no espaço de busca. A figura a seguir ilustra o posicionamento das partículas após várias iterações.

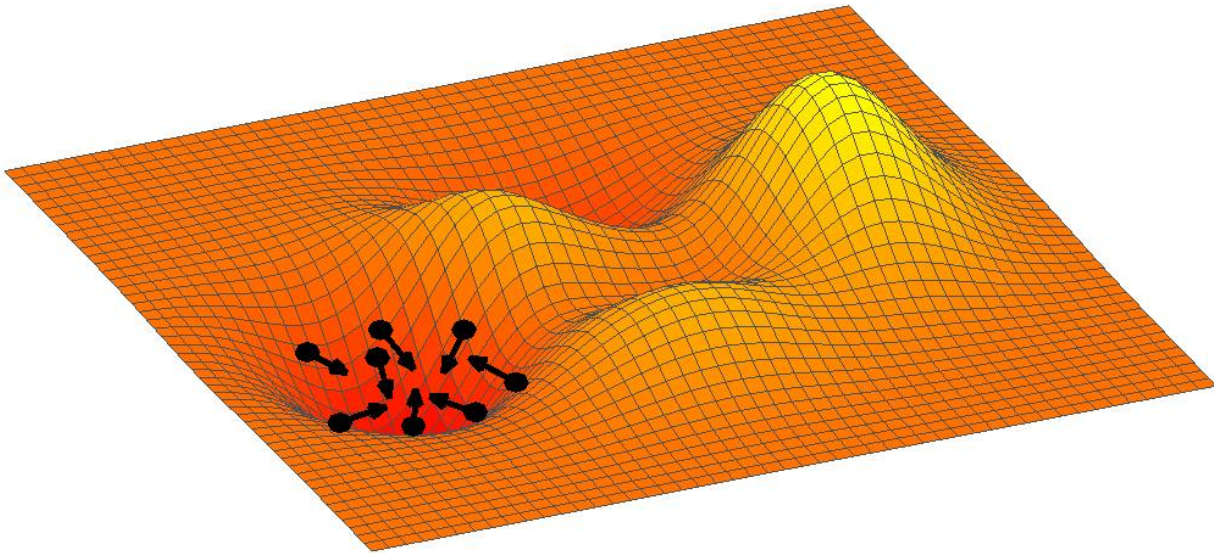


Figura 6.3.3: Localização das partículas após o processo de otimização

O algoritmo que descreve o processo de OEP [8] é apresentado a seguir:

- *Passo 1*: Inicializar as posições e velocidades das partículas aleatoriamente no espaço *n*-dimensional;

- *Passo 2:* Para cada partícula:

- Avaliar sua posição atual de acordo com a função objetivo

  - Se o valor obtido for melhor que *p1*, atualizar p1 com o novo valor;

  - Se o valor obtido for melhor que *p2*, atualizar p2 com o novo valor;

- Alterar a velocidade e posição da partícula;

- *Passo 3:* Repetir os passos anteriores até que o critério de parada seja satisfeito;

A cada execução do algoritmo tem-se o comportamento das partículas conforme ilustrado a seguir:

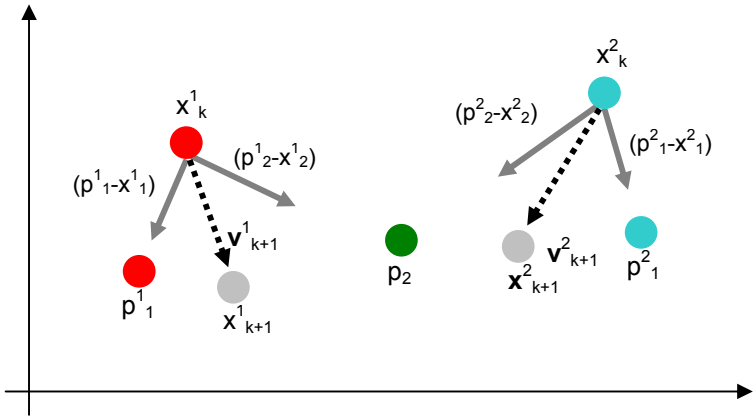


Figura 6.3.4: Movimento resultante em direção a melhor posição

Onde:

$r_1b_1 = r_2b_2 = 1$;

x^1_k é a posição atual da partícula 1;

x^1_{k+1} é a próxima provável posição da partícula 1;

x^2_k é a posição atual da partícula 2;

x^2_{k+1} é a próxima provável posição da partícula 2;

p^1_1 é a melhor posição da partícula 1;

p^2_1 é a melhor posição da partícula 2;

p_2 é a melhor posição de todas as partículas;

6.4 Variantes do OEP

Após primeira versão do método de OEP, vários pesquisadores ao aplicarem-no em situações práticas, perceberam que algumas modificações seriam pertinentes para o aprimoramento da técnica.

A primeira modificação proposta foi a adição de um fator, denominado *limitador de velocidade* (v_m), capaz de impor limites de velocidade, já que valores muito altos podem fazer

com que a partícula ultrapasse uma região promissora e muito baixos podem impedi-la de atingir um ponto desejado.

O limitador de velocidade é formado por um intervalo [$v_{mín}$,$v_{máx}$] que define limites máximos e mínimos. Desta forma, se o novo valor atualizado de velocidade for maior que o limite superior, a este deverá ser atribuído $v_{máx}$ e quando a velocidade atingir valores inferiores ao limite mínimo, a ela atribui-se o valor de $v_{mín}$.

Apesar de ser uma tentativa de solucionar o problema de controle de velocidade, este fator provocou efeitos indesejados, comprometendo a eficiência do método e conseqüentemente o resultado final em comparação a outros métodos de CE [34].

A solução proposta por [35] sugere a adição de dois novos parâmetros denominados *fator de constrição* e *componente inercial.*

6.4.1 Componente Inercial

A componente inercial foi proposta com intuito de controlar a velocidade das partículas de forma ponderada. Assim, tem-se uma nova proposta de atualização da velocidade da seguinte forma:

$$v^{k+1} = w_k v_k + b_1 r_1 (p_1 - x_k) + b_2 r_2 (p_2 - x_k) \qquad (5)$$

$$w^{k+1} = w_{máx} - k\left(\frac{w_{máx} - w_{mín}}{k_{máx}}\right) \qquad (6)$$

Onde $w_{mín}$ e $w_{máx}$ são, respectivamente, os limites impostos à componente inercial e k_{max} o número máximo de iterações.

Ao analisar as equações acima, pode-se perceber que a componente de ponderação inercial tem uma variação linear. Para Shi e Heberhart [35]**,** é ideal que *w* seja iniciado com um valor alto e vá se decrementando, já que um alto valor de *w* desloca a partículas para posições

mais distantes da atual, promovendo assim a exploração global e um baixo valor de *w* a desloca para regiões próximas, o que equivale à exploração local. Os mesmos relatam que valores de $w_{máx} = 0{,}9$ e $w_{mín} = 0{,}4$ foram determinantes para a obtenção de bons resultados.

Como alternativa Chatterjee e Siarry [36] apresentam uma proposta de variação não-linear da componente *w*. Esta tem o objetivo de impor um controle mais eficiente sobre a velocidade da partícula nas iterações finais de forma a promover a convergência.

A proposta é válida nos casos em que a variação linear faz com que a partícula chegue ao final do processo com velocidade tão reduzida a ponto de não alcançar a posição ideal.

A equação seguinte representa a variação não-linear da componente inercial :

$$w^{k+1} = w_{máx} - \left\{ \frac{(R-k)^{nl}}{R^{nl}} \right\} * \left(w_{máx} - w_{mín} \right) \qquad (7)$$

Onde *nl* representa o fator de não linearidade.

6.4.2 Coeficiente de Constrição

O coeficiente de constrição foi proposto por Clerc e Kennedy [37], que demonstraram que o mesmo pode ser usado para prevenir a explosão do sistema através da perda de controle sobre as velocidades das partículas. A equação a seguir mostra como o coeficiente é calculado:

$$\chi = \frac{2}{\left| 2 - \phi - \sqrt{\phi^2 4\phi} \right|} \qquad (8)$$

Onde:

- $\phi = \phi_1 + \phi_2$
- $\phi_1 = b_1 r_2$
- $\phi_2 = b_2 r_2$

Assim, a equação de atualização da velocidade passa a ser definida da seguinte forma:

$$v^{k+1} = \chi\left[v_k + b_1 r_1 (p_1 - x_k) + b_2 r_2 (p_2 - x_k)\right] \qquad (9)$$

Foi mostrado também que para valores de $\phi_2 > 4$ as partículas tendem a convergir rapidamente para a solução global, enquanto que para $\phi_2 < 4$ haveria uma forte tendência de conversão para um mínimo local. Este comportamento é muito similar ao provocado pela inserção do fator de inércia.

6.5 Redes neuro-fuzzy-polinomiais otimizadas via OEP

As dificuldades encontradas na otimização de redes neuro-fuzzy-polinomiais (RNFP) via algoritmos genéticos conduzem à avaliação de novas técnicas que possibilitem a geração de uma estrutura de RNFP compacta, ágil e capaz de convergir rapidamente, sem a necessidade de um excessivo número de ajustes de parâmetros a elevado tempo de processamento.

Por este motivo a proposta de utilização do OEP no auxílio à determinação das definições de projeto, assim como no ajuste de parâmetros inerentes à RNFP, é extremamente válida no que diz respeito às vantagens do OEP em relação a outros métodos de otimização como os algoritmos genéticos e recozimento simulado.

Como exemplos, podem-se citar aplicações como [38] onde o OEP foi aplicado no processo de treinamento de redes neurais do tipo feedfoward e em [39] em que o OEP foi avaliado através de aplicações envolvendo operação de sistemas elétricos de potência.

Para a proposta do presente trabalho é necessária uma metodologia de projeto que provoque a cooperação entre todas as técnicas envolvidas.

Este de trabalho apresenta, nesta linha de raciocínio, uma metodologia capaz de fazer com que sejam otimizados todos os parâmetros envolvidos no processo de projeto quase que automático, restando apenas poucos ajustes referentes ao método de otimização utilizado.

No intuito de buscar resultados ainda melhores do que os obtidos anteriormente, optou-se por envolver não só os parâmetros mais "superficiais", mas também aqueles mais internos, como o centro e largura das funções de pertinência, além da "sintonia fina" dos coeficientes dos polinômios de cada neurônio polinomial.

Para que isto seja possível, deve-se trabalhar não só com a versão contínua do OEP, mas também com uma versão discreta [40].

Apesar da versão tradicional do OEP ser contínua, atualmente diversas aplicações são desenvolvidas com base na versão discreta como, por exemplo, em [41, 42, 43, 44].

Nesta, as componentes que representam as posições das partículas assumem somente valores iguais a zero ou um. Quando a velocidade é utilizada para o ajuste de posições, deve ser colocada no intervalo [0,1], o que pode ser feito através da seguinte função sigmoidal:

$$sig(x) = \frac{1}{1 + \exp(-x)} \qquad (10)$$

Na equação (5), de ajuste da velocidade, o termo x^{k+1} é determinado, de forma probabilística, da seguinte maneira:

$$x^{k+1} = \begin{cases} 0, & se\ \xi \geq sig\left(v^{k+1}\right) \\ 1, & se\ \xi < sig\left(v^{k+1}\right) \end{cases}$$

Onde, ξ é um número aleatório gerado a partir de uma distribuição uniforme no intervalo [0,1].

A aplicação das versões discretas e contínuas pode ser realizada em conjunto, porém optou-se por fazê-la em duas etapas: na primeira é utilizada a versão discreta para o projeto da estrutura como um todo (topologia, tipos de funções de pertinência, etc) e na segunda é aplicada a versão contínua para que seja possível verificar a possibilidade de melhoria nos coeficientes estimados pelo método dos mínimos quadrados.

A geração das partículas, referente à primeira etapa, é feita com base na estrutura apresentada a seguir, de forma similar a utilizada na otimização via AG.

N1	N1	N1	N2	N2	FP	FP	NC	NC	GP1	GP2

Os parâmetros otimizados são:

N1 = número de neurônios selecionados para a camada 1;

N2 = número de neurônios selecionados para a camada 2;

FP = tipo de função de pertinência;

NC = número de conjuntos fuzzy por variável;

GP1 = grau do polinômio na camada polinomial 1;

GP2= grau do polinômio na camada polinomial 2.

Neste caso todas as posições do vetor (partícula) são valores binários.

Na segunda etapa, tem-se um modelo de partícula como mostrado abaixo:

D1	D2	D3	D4	A	B	C

D1kij = primeira coordenada do conjunto difuso *k* do neurônio *i* na camada *j*;

D2kij = segunda coordenada do conjunto difuso *k* do neurônio *i* na camada *j*;

D3kij = terceira coordenada do conjunto difuso *k* do neurônio *i* na camada *j*;

D4kij = quarta coordenada do conjunto difuso *k* do neurônio *i* na camada *j*;

A = primeiro coeficiente estimado para o neurônio polinomial *p*.

B = segundo coeficiente estimado para o neurônio polinomial *p*.

C = terceiro coeficiente estimado para o neurônio polinomial *p*.

Onde *k* corresponde ao conjunto difuso que compõe a função de pertinência do neurônio *i* na camada *j*. A, B e C são os coeficientes dos polinômios dos neurônios polinomiais.

As coordenadas apresentadas no exemplo de partícula acima baseiam-se numa função de pertinência do tipo trapezoidal.

Para que seja possível o ajuste nos parâmetros adotados na segunda etapa, tem-se que saber exatamente a topologia da rede, por este motivo o processo deve ser realizado em duas etapas. Primeiro se estabelece a topologia com os números de neurônios, graus de polinômios e tipos de função de pertinência adequados e só depois, na segunda etapa, realiza-se os ajustes necessários.

Para melhor entendimento é apresentada, em seguida, a seqüência de passos necessária à implementação do processo de otimização de RNFP via OEP no software Matlab versão 7.4.0, tomado como referência para as simulações e testes.

Passo 1 - Inicialização dos parâmetros referentes à OEP

São inicializados os seguintes parâmetros:

- S = Tamanho da população (enxame)

- P = Tamanho da partícula

- N = Número de iterações

- c_1, c_2 = acelerações (cognitiva e social)

- V_i = velocidade inicial

- V_f = velocidade final

- Vm = máximo incremento de velocidade

- C = fator de constrição

Passo 2 – Inicialização das partículas (enxame)

O enxame é criado com o número de partículas definido pela constante *S* e estas são inicializadas com tamanhos definidos pela constante *P*.

Passo 3 – Avaliação das partículas

Para a avaliação de cada partícula deve-se calcular a saída de cada NFP. Para isto deve-se utilizar dos recursos, através de linhas de comando, que a toolbox de redes neuro-fuzzy (ANFIS) oferecem. Após o cálculo das saídas dos neurônios que irão compor a primeira camada, deve-se estimar o erro médio quadrático (MSE) ou o somatório dos erros quadráticos (SSE) entre as saídas calculadas para cada NFP e a saída desejada da RNFP. Aqueles que apresentarem menor valor de erro permanecerão na estrutura e os outros não. As saídas destes melhores serão as entradas para a próxima camada. Cada neurônio polinomial na próxima camada tem como entrada uma combinação das saídas, tomadas duas a duas, assim como na filosofia de projeto das RNP. O processamento de cada neurônio agora é realizado através de um polinômio cujos coeficientes são ajustados às entradas, através do método dos mínimos quadrados. Os dois neurônios que obtiverem melhores aproximações e gerarem como saída, valores mais próximos aos desejados, serão preservados e os outros descartados. Estes se recombinam para formar o neurônio de saída que terá como base de processamento um polinômio estimado da mesma forma.

Como cada partícula prevê uma topologia com funções de pertinência e graus de polinômios pré-definidos, é possível avaliar quais são as mais promissoras. Assim tem-se condição de verificar quais partículas estão mais bem posicionadas no espaço de busca.

Passo 4 – Atualização das posições das partículas

De acordo com as definições do OEP as partículas têm suas posições atualizadas de acordo com a componente social e cognitiva.

Em seguida o processo é retornado ao passo 3. Isto ocorre até que o número de iterações pré-estabelecido seja atingido.

O mecanismo de hibridização é similar para as duas etapas, diferenciando-se apenas na forma em que as partículas são geradas e no processo de atualização das posições, visto que na primeira etapa os parâmetros são discretos e no segundo contínuos.

A partir deste ponto a avaliação da proposta é verificada através da realização de testes com os conjuntos de dados utilizados anteriormente e apresentados a seguir.

6.5.1 Testes Práticos

A partir deste ponto serão realizados testes práticos, envolvendo otimização de RNFP via OEP, com todos os conjuntos de dados utilizados anteriormente. Estes são fundamentais para avaliar o desempenho do método proposto e compará-lo com os resultados obtidos até então. Para que seja coerente a comparação, a estrutura de RNFP e os parâmetros envolvidos a serem otimizados são idênticos aos apresentados nos testes com AG. Além disto, optou-se por um modelo de partícula, na fase de aplicação do OEP discreto, exatamente igual ao cromossomo no AG.

As principais configurações adotadas para o OEP são as seguintes:

- número de iterações = 100;
- número de partículas = 5;
- c1=c2=2;
- velocidade inicial = 0.9;

- velocidade final = 0.2;

- velocidade máxima = 1.5;

- fator de constrição = 1;

6.5.2 Resultados Obtidos

A seguir são apresentados os resultados obtidos nos processos de treinamento e validação em todos os casos.

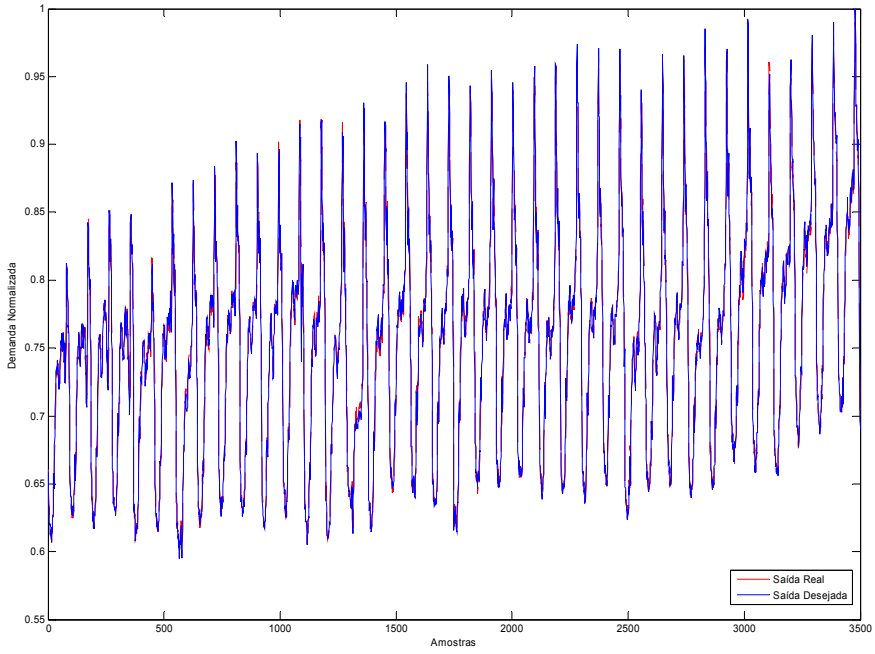


Figura 6.5.2.1: Resultado do processo de treinamento para o primeiro conjunto de dados

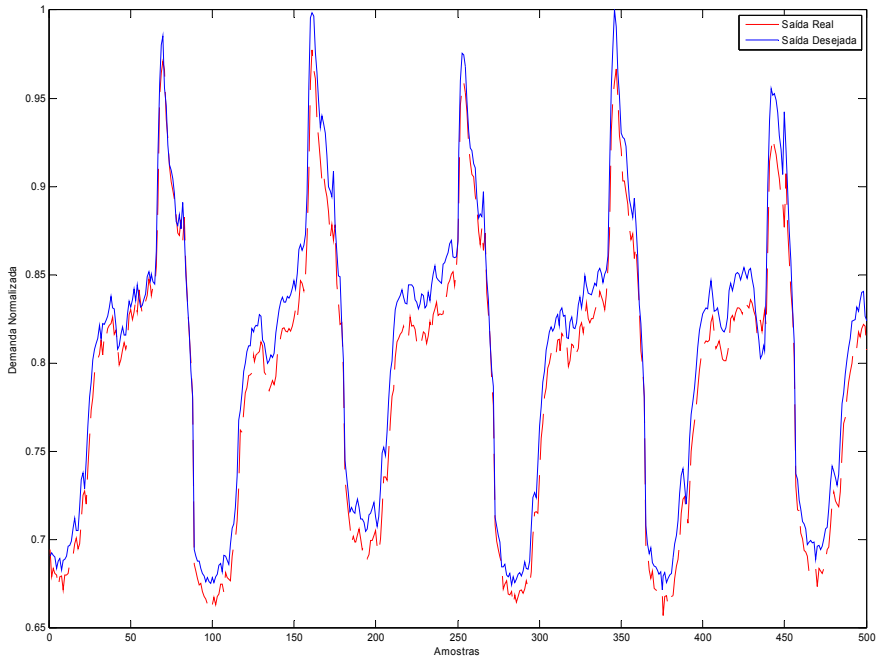


Figura 6.5.2.2: Resultado do processo de validação para o primeiro conjunto de dados

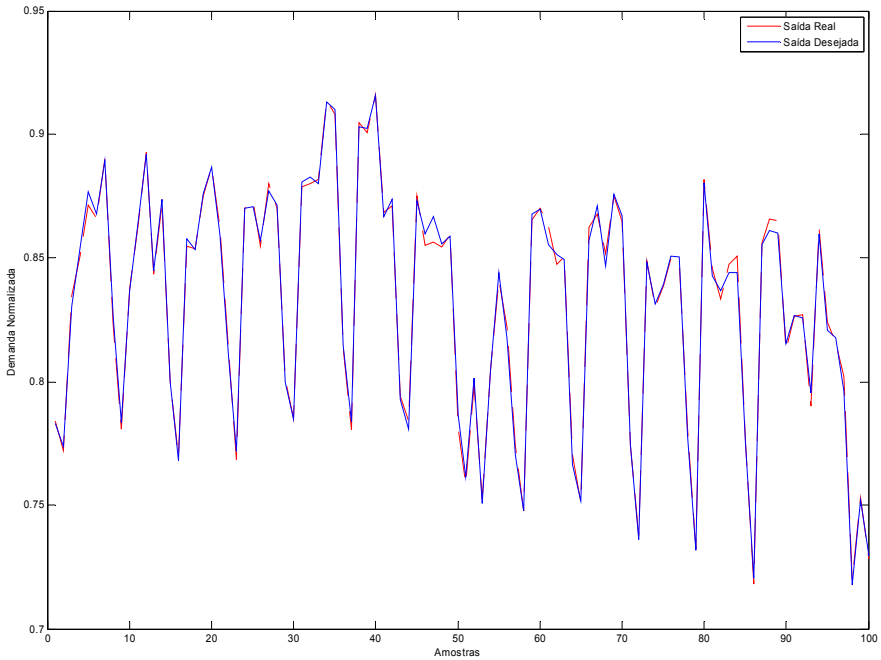


Figura 6.5.2.3: Resultado do processo de treinamento para o segundo conjunto de dados

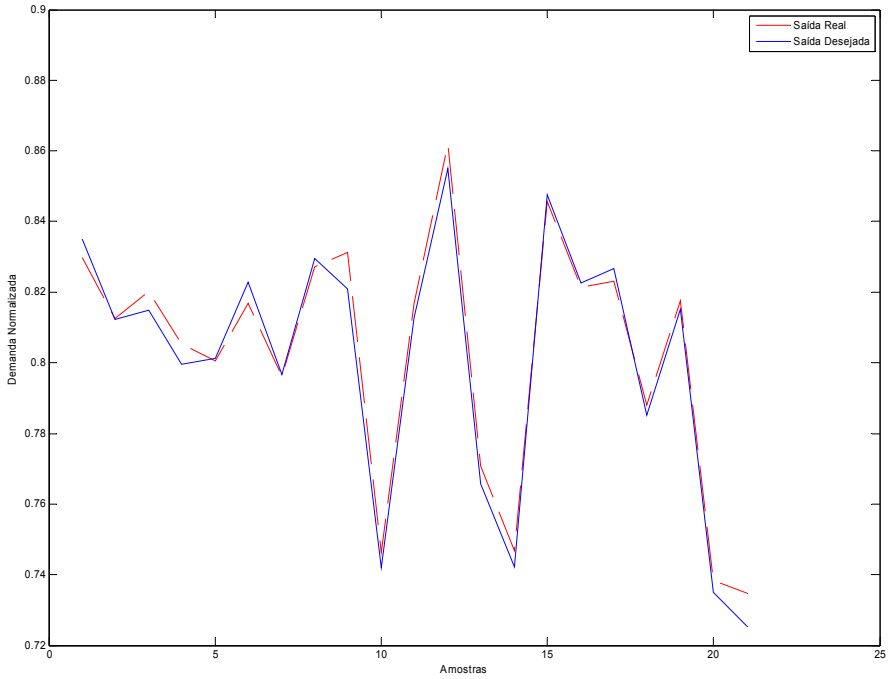


Figura 6.5.2.4: Resultado do processo de validação para o segundo conjunto de dados

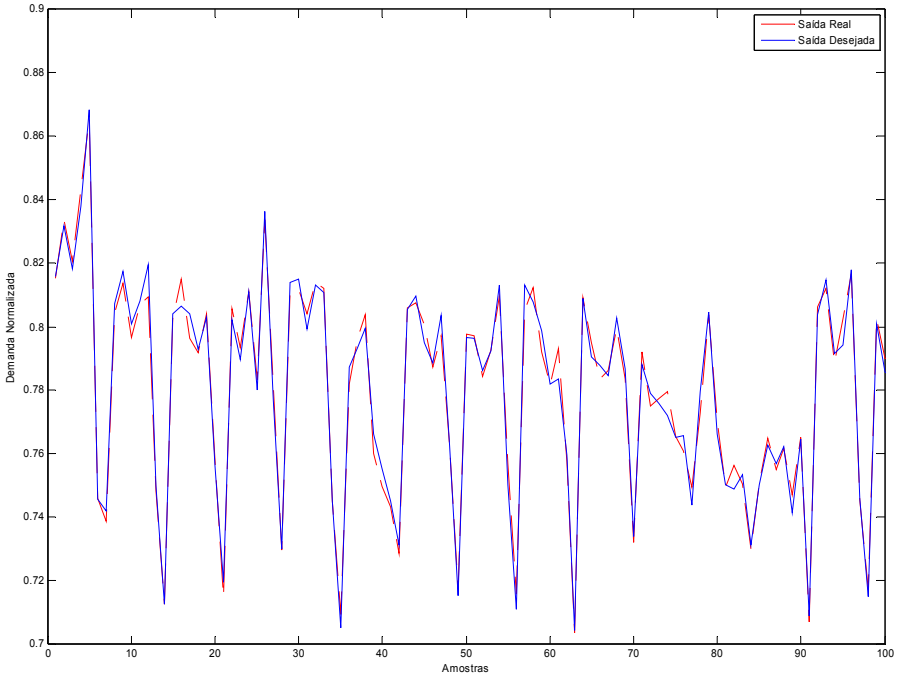


Figura 6.5.2.5: Resultado do processo de treinamento para o terceiro conjunto de dados

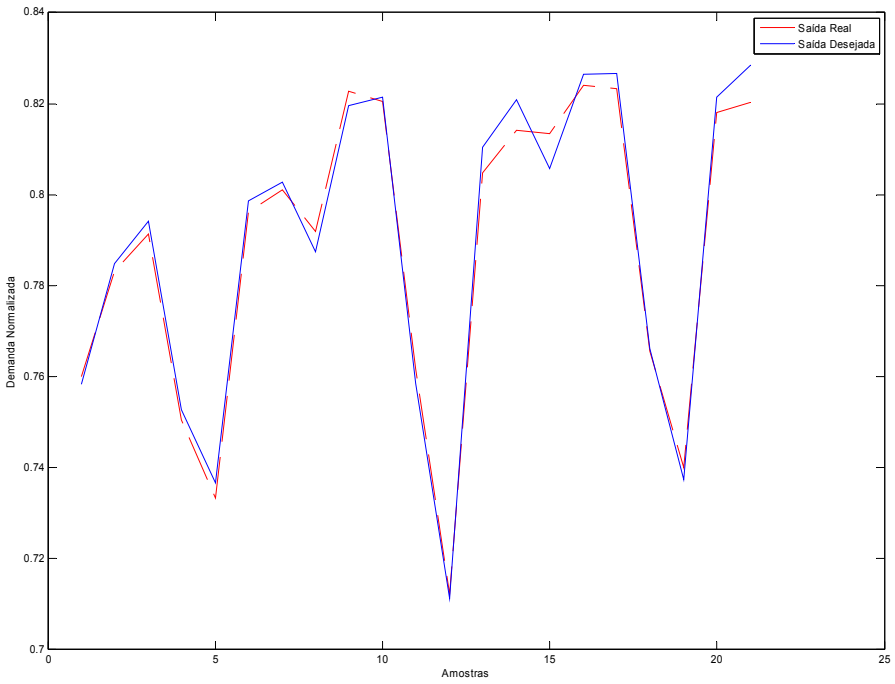


Figura 6.5.2.6: Resultado do processo de validação para o terceiro conjunto de dados

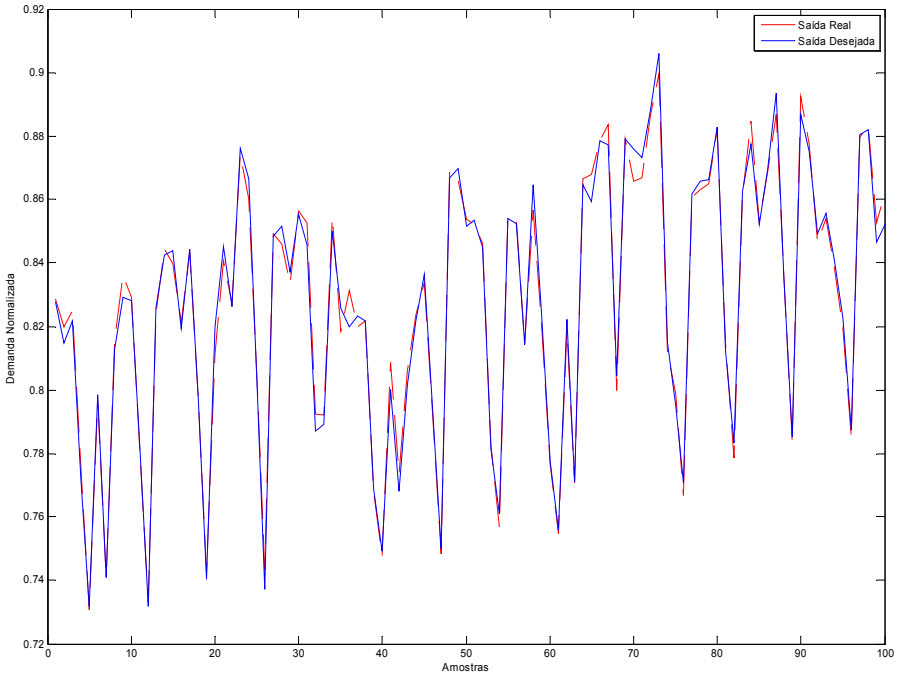


Figura 6.5.2.7: Resultado do processo de treinamento para o quarto conjunto de dados

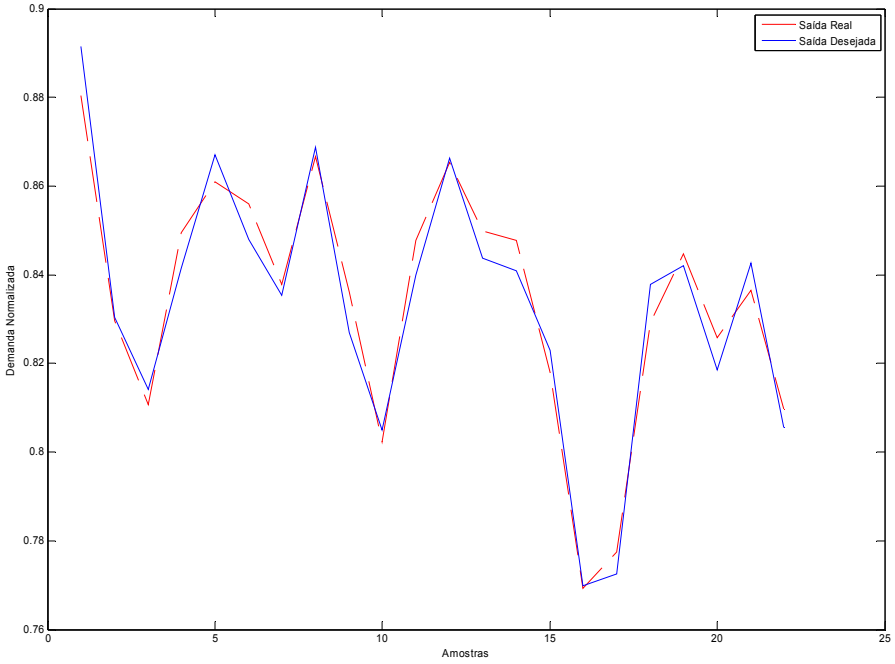


Figura 6.5.2.8: Resultado do processo de treinamento para o quarto conjunto de dados

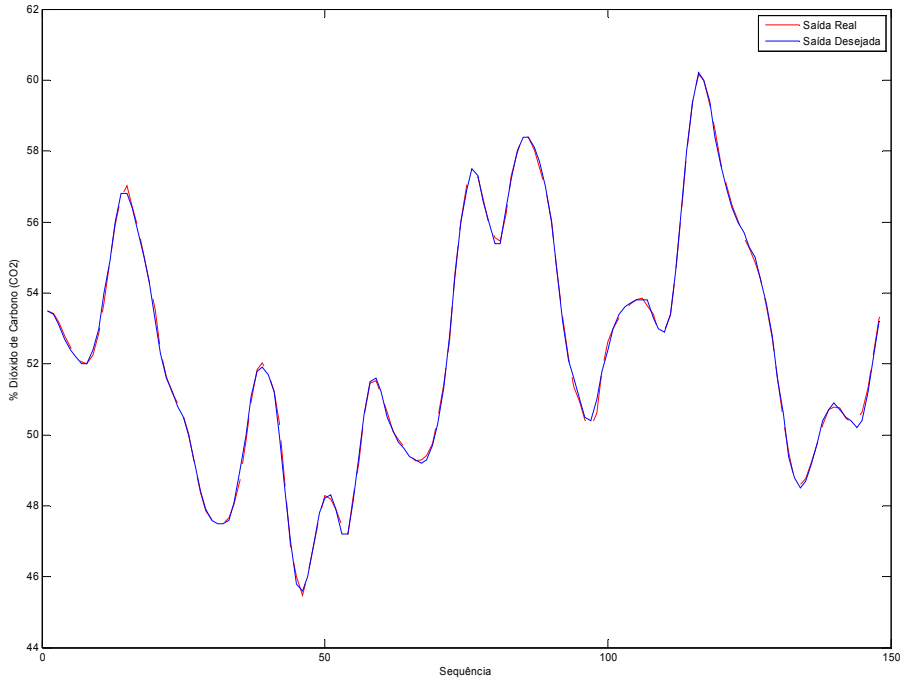


Figura 6.5.2.9: Resultado do processo de treinamento para a série de dados Gas Furnace

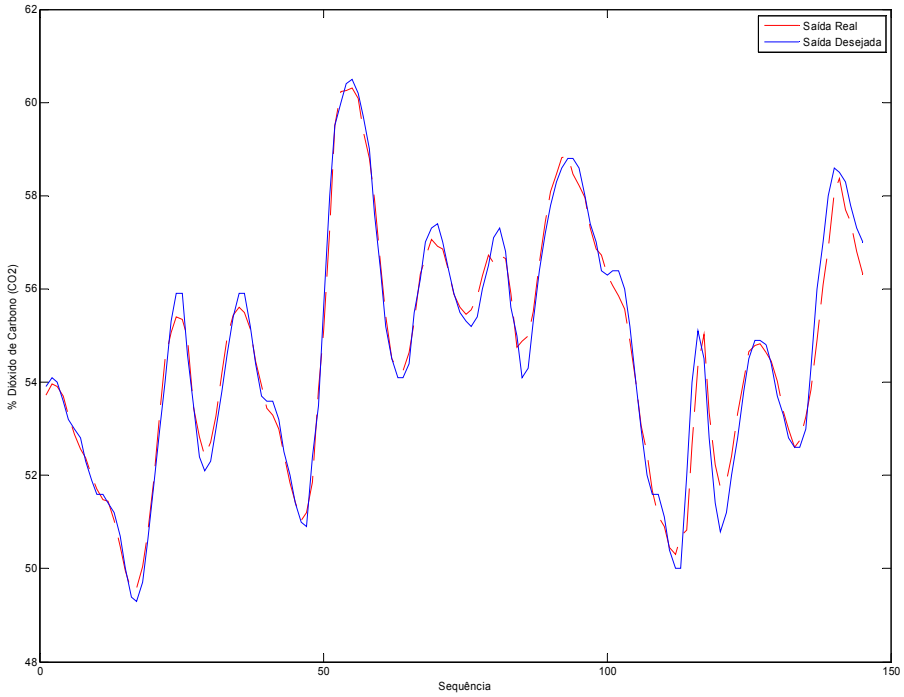


Figura 6.5.2.10: Resultado do processo de validação para a série de dados Gas Furnace

A tabela seguinte mostra os resultados obtidos nos testes realizados com redes neuro-fuzzy-polinomiais, otimizadas por OEP:

CONJUNTO DE DADOS	$SSE_{TREINAMENTO}$	$SSE_{VALIDAÇÃO}$
1	0.2611	3.1669E-004
2	8.5068e-004	4.9362e-004
3	0.0015	3.1971e-004
4	0.0017	7.8513e-004
Gas Furnace	2.0108	22.1656

Tabela 6.5.3.1: Resultados obtidos com redes neuro-fuzzy polinomiais, em relação a todos os conjuntos de dados

6.5.3 Avaliação dos Resultados

Os resultados obtidos demonstram a capacidade de exploração do espaço de busca quando utilizada a OEP. Porém, assim como aconteceu nos testes realizados com algoritmos genéticos, o método de otimização por enxame de partículas apresentou problemas de convergência

prematura. Isto ficou evidente no caso dos AG quando os bits referentes à topologia da RNFP assumiam valores iguais a “1”.

Em todos os casos, conduziu a uma estrutura final com muitos neurônios, o que contradiz a proposta de uma rede compacta e ágil.

No caso do OEP verificou-se, em alguns casos, que quando uma determinada partícula assumiu uma região promissora no espaço de busca, atraiu todas as outras para suas proximidades e o processo de busca se limitou às vizinhanças desta, o que também pode ser traduzido como convergência prematura, uma vez que a região encontrada não necessariamente foi a ótima.

O uso de artifícios como o fator de constrição e componente inercial, auxiliou no processo de busca, mas não de maneira a melhorar significativamente os resultados.

Nos casos onde tal fenômeno ocorreu, foi utilizada a versão do OEP proposta por Esmin [45], chamada de HPSOM. Esta tem como base a introdução de um mecanismo de mutação, exatamente como nos algoritmos genéticos, sobre uma determinada porção do enxame.

No caso específico deste trabalho, onde foi adotado um número de cinco partículas, sempre uma delas, escolhida aleatoriamente passou pelo processo de mutação.

A substituição do OEP pelo HPSOM trouxe melhorias significativas em dois casos: no primeiro e quarto conjunto de dados. Nos outros casos a utilização do HPSOM apresentou melhorias, porém pouco significativas se comparadas aos citados anteriormente.

6.5.4 Comparativo Final

Após a realização de todos os testes é importante um comparativo final que permita a comparação entre todos os resultados apresentados em todos os casos, conforme mostrado no quadro a seguir.

		CONJUNTO DE DADOS				
		1	**2**	**3**	**4**	**5**
TÉCNICA	**RNA**	0,1668	7,21e-004	3,92e-004	6,83e-004	33,6402
	RNP	0,1628	6,4551e-004	3,5085e-004	**6,3026e-004**	29,9934
	RNF	0.1914	0.0017	0.0015	0.0019	45.0636
	RNFP	0,3898	5,7918e-004	0,0013	8.8465e-004	26,4209
	RNFP-AG	0.1584	5.7492e-004	5.4575e-004	0.0010	29.8329
	RNFP-OEP	**3.1669e-004**	**4.9362e-004**	**3.1971e-004**	7.8513e-004	**22.1656**

Tabela 6.5.4.1: Resultados obtidos com todas as técnicas apresentadas

Pelo quadro comparativo é possível perceber que o método proposto apresentou resultados melhores em quase todos os testes, com exceção do quarto caso e mesmo neste, obteve boa performance, o que só reforça a eficiência do método.

Além disto, devem ser destacados aspectos como a simplicidade na implementação e ajustes iniciais do OEP além do tempo de processamento muito menor do que o necessário ao processo de otimização via AG, como mostrado a seguir.

		Tempo médio de processamento	Iterações
TÉCNICA	**RNA**	10 a 30 segundos	100
	RNP	4 a 10 segundos	100
	RNF	1 a 5 minutos	10
	RNFP	1 a 5 minutos	10
	RNFP-AG	10 a 30 minutos	10
	RNFP-OEP	1 a 5 minutos	100

Tabela 6.5.4.2: Comparação entre os tempos de processamento necessários ao processo de treinamento

A quantidade de iterações apresentadas se refere ao mínimo necessário para a obtenção dos resultados obtidos. A partir destas não houve melhorias significativas.

CONCLUSÕES

O processo de previsão é útil em diversas atividades humanas. É nele que se baseiam os desenvolvimentos futuros, as etapas de um planejamento e as verificações de disponibilidade dos sistemas. É sabido que quanto maior o horizonte de previsão mais difícil acertar os valores previstos face aos que realmente ocorrem. E mais, que a qualidade do processo de previsão está intimamente ligada à qualidade da base histórica dos dados disponível e da repetibilidade deste conjunto.

Uma base histórica com dados ruins ou com muitos ruídos pode gerar valores previstos muito fora da realidade, enquanto que a falta de repetição de padrões leva a construção de sistemas de previsão que não oferecem resultados homogêneos ao longo do tempo.

Este trabalho teve como proposta inicial a investigação das principais técnicas de inteligência artificial necessárias à solução de problemas de identificação de sistemas complexos e não lineares, além da união entre elas para a formação de sistemas híbridos mais robustos e eficientes.

Para verificar a aplicabilidade destas, foi escolhido o problema de previsão em séries temporais. Os diversos testes realizados buscaram apresentar de forma clara e objetiva, uma forma de compará-las através da utilização de conjuntos de dados em comum.

No primeiro caso, avaliou-se a performance das redes neurais artificiais do tipo perceptron multicamadas (MLP) com treinamento realizado pelo algoritmo de retropropagação dos erros. Normalmente, este tipo de estrutura apresenta bons resultados quando aplicadas em problemas de média à baixa complexidade. Caso contrário passam a sofrer com problemas de super e sub-especialização causados por questões associadas ao cálculo de derivadas. Isto faz com que o

processo de treinamento seja interrompido de forma prematura devido ao estacionamento em um ponto de mínimo local.

Em seguida verificou-se a capacidade de generalização e aprendizado, apresentadas pelas redes neurais polinomiais, RNP. Com topologias finais compactas, as mesmas se mostraram ágeis, simples de serem treinadas e capazes de produzir bons resultados. As dificuldades se referem às definições de projeto, como o número ótimo de camadas, de neurônios selecionados por camada e também os graus dos polinômios em cada neurônio.

Pelas afirmações anteriores, pode-se perceber que a maior dificuldade de projetos envolvendo de redes neurais artificiais consiste na definição de uma topologia ótima.

Diante disto, passou-se à análise dos sistemas híbridos, onde a cooperação entre duas ou mais técnicas passa a ser ponto chave para a geração de sistemas mais robustos e capazes de lidar com características de não-linearidade, típicas de sistemas complexos.

O primeiro modelo analisado foi o neuro-fuzzy. Neste ficou evidente que a necessidade de se ajustar muitos parâmetros faz com que o processo de treinamento seja lento e nem sempre apresente bons resultados. Apesar de gerar automaticamente a base de regras necessária ao sistema fuzzy, apresenta a limitação de trabalhar com poucas entradas. Os resultados apresentados deixaram a desejar e conduziram ao estudo de novas metodologias de projeto como as redes neuro-fuzzy-polinomiais (RNFP).

As RNFP foram propostas com o intuito de criar o mapeamento dos pares entrada-saída, típico das redes neuro-fuzzy, com a capacidade de generalização das redes neurais polinomiais. Através dos testes realizados com este tipo de associação também foi possível comprovar sua viabilidade através dos resultados obtidos. Por outro lado tem-se como inconveniente a dificuldade nas definições de projeto inerentes a cada estrutura operando em separado. Por este motivo verificou-se a necessidade de associação de um novo método capaz de encontrar o

conjunto de parâmetros otimizados para automatização do projeto das RNFP. A primeira investigação abordou o uso dos algoritmos genéticos como método de busca. Apesar de suas características de fácil implementação e geração de bons resultados, o mesmo apresentou-se lento e consumiu tempo de processamento mais elevado que a maioria dos casos anteriores para o processo de otimização.

A proposta inicial, tida como objetivo primário deste trabalho foi, além de investigar os principais métodos e técnicas aplicados ao problema de identificação de sistemas, propor um novo modelo baseado em RNFP e otimizados por algoritmos de enxame, ou otimização por enxame de partículas – OEP.

Através da aplicação desenvolvida pode-se constar que além de viável em termos de resultados, a mesma necessitou de tempos de processamento relativamente baixos para a geração dos resultados apresentados. Vale lembrar que a utilização do HPSOM foi de extrema valia e só através de sua aplicação é que foram possíveis melhorar os resultados obtidos em dois casos específicos.

Muito além dos resultados, buscou-se uma análise em função do tempo necessário à aplicação de cada método, da estrutura final otimizada e à quantidade de ajustes necessários.

As discussões, análises e conclusões, apresentadas ao final de cada caso foram de fundamental importância para fortalecer a necessidade de se propor uma nova metodologia capaz de atender aos requisitos de desempenho, generalização e agilidade.

A opção do OEP no processo de otimização de RNFP se justificou por suas características intrínsecas de agilidade, simplicidade e desempenho. A escolha das RNFP se deu pelo fato de serem auto-organizáveis e juntaram em uma só estrutura as características de agrupamento de dados das redes neuro-fuzzy com a capacidade de expansão do espaço representativo das redes neurais polinomiais. Pelos resultados apresentados pode-se considerar como promissora a

proposta já que a mesma contempla não só os resultados, mas também os requisitos citados anteriormente.

Como trabalhos futuros podem-se mencionar aplicação do sistema proposto em cargas elétricas diversas, optando-se por um sistema de agrupamento de dias típicos ou de áreas típicas durante a semana ou períodos do ano. Outro trabalho seria a confecção de um sistema híbrido capaz de fazer com que o sistema proposto pudesse aumentar seu horizonte de previsão gerando um sistema com maior número de aplicações. Além disto, é claro, aplicações envolvendo previsão de carga em tempo real.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Zadeh, L. A., “Fuzzy Sets,” in Information and Control, vol. 8. New York: Academic Press, 1965, pp. 338-353.

[2] Jang, J.-S. R., “ANFIS: Adaptive-Network-Based Fuzzy Inference System”, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, Nº 3, May/June – 1993.

[3] Rumelhart, D. E., Hinton,G. E., Williams,R. J., “Learning internal representations by error propagation”, Eds: Parallel Distributed Processing, Vol 1. MIT Press, Cambridge,1986.

[4] Lopes, M.L.M., Minussi, C.R., “Treinamento de Redes Neurais via Back-propagation com Controlador Nebuloso,” Proceedings do Congresso Brasileiro de Automática, 2000.

[5] Hines, J.W., Wrest, D.J., Uhrig, R.E., “Signal Validation using an Adaptive Neural Fuzzy Inference System,” Nuclear Technology, Vol. 119, August 1997.

[6] Calado, J. M. F., Mendes, M. J. G. C., Costa, J. M. G., Korbiez, J., “Neuro and neurofuzzy hierarchical structures comparison in FDI: case study”. IFAC, 15th Triennial World Congress, Barcelona, Spain, 2002.

[7] Wang X.Y. and Hu, W.Y., “The self-organizing polynomial network algorithm based on the self-organizing theory”. System Engineering - Theory & Practice, 19(4), 51-56. 1999.

[8] Eberhart, R. C. and Kennedy, J. “A new optimizer using particle swarm theory”, Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995.

[9] Kennedy, J. and Eberhart, R. C. “Particle swarm optimization”, Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995.

[10] McCulloch , Warren and Pitts, Walter, “A logical calculus of the ideas immanent in nervous activity”, Bulletin of Mathematical Biophysics, volume 5, 1934.

[11] Hebb, D.O. ”The Organization of Behavior: A Neuropsychological Theory”. John Wiley & Sons. New York, EUA, 1949.

[12] Rosenblatt, F., “Principles of Neurodynamics”, Spartan, New York, EUA, 1962.

[13] Rumelhart, D., McClelland, “Parallel Distributed Processing: Explorations in the Microstructure of Cognition”, Cambridge, EUA, MIT Press, 1986.

[14] Boole, G., “The Mathematical Analysis of Logic”, Cambridge, 1957.

[15] Ivakhnenko, A.G. “Polynomial Theory of Complex Systems”, IEEE Trans, on Systems, Man, and Cybernetics, vol.1, N:4, 364-378. 1971.

[16] Plackett, R.L., “The discovery of the method of least squares”, Biometrika, **59**, 239–251. 1972.

[17] S. J. Farlow, “Self-Organizing Methods in Modeling. GMDH Type Algorithm.” Marcel Dekker. New York, Basel. 1984.

[18] Goldberg, David E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Publishing Company, Inc., 1989.

[19] Holland J.H., "Adaptation in Natural and Artificial Systems", MIT Press, Cambridge, Massachusetts, 1975.

[20] S.-K. Oh, W. Pedrycz, H.-S. Park, “Self-organizing neurofuzzy networks in modeling software data”, Fuzzy Sets and Systems 145 (2004) 165–181, Elsevier B.V.

[21] Takagi,T., Sugeno, M., “Fuzzy Identification of Systems and its Applications to Modeling and Control,” IEEE Transactions on Systems Man, and Cybernetics, Vol. 15, No. 1, pp. 116-132, 1985.

[22] Lambert-Torres, G.; Borges da Silva, L.E. ; Masselli, Y.M.C.. Fuzzy Systems. In: Advanced Techniques and Technologies Facts and AI – Vol. III - Part2, por Chen-Ching Liu (Ed.), 33 p., 2009 (no prelo).

[23] S.-K. Oh, W. Pedrycz, H.-S. Park, “Hybrid identification in fuzzy-neural networks”, Fuzzy Sets and Systems 138 (2003) 399–426, Elsevier B.V.

[24] S.-K. Oh, W. Pedrycz, H.-S. Park, “Multi-layer hybrid fuzzy polynomial neural networks: a design in the framework of computational intelligence”, Neurocomputing 64 (2005) 397–431, Elsevier B.V.

[25] S.-K. Oh, W. Pedrycz, H.-S. Park, “Self-organizing neurofuzzy networks in modeling software data”, Fuzzy Sets and Systems 145 (2004) 165–181, Elsevier B.V.

[26] H.-S. Park, W. Pedrycz, S.-K. Oh, “Evolutionary design of hybrid self-organizing fuzzy polynomial neural networks with the aid of information granulation”, Expert Systems with Applications 33 (2007) 830–846, Elsevier B.V.

[27] S.-K. Oh, S.-B. Roh, W. Pedrycz, T.-C. Ahn, “IG-based genetically optimized fuzzy polynomial neural networks with fuzzy set-based polynomial neurons”, Neurocomputing 70 (2007) 2783–2798, Elsevier B.V.

[28] H.S.Hwang, “Fuzzy GMDH-type neural network model and its application to forecasting of mobile communication”, Computers & Industrial Engineering 50 (2006) 450–457, Elsevier B.V.

[29] S.-K. Oh, W. Pedrycz, “Multi-layer self-organizing polynomial neural networks and their development with the use of genetic algorithms”, Journal of the Franklin Institute 343 (2006) 125–136, Elsevier B.V.

[30] S.-K. Oh, W. Pedrycz, “A new approach to self-organizing multi-layer fuzzy polynomial neural networks based on genetic optimization”, Advanced Engineering Informatics 18 (2004) 29–39, Elsevier B.V.

[31] S.-K. Oh, W. Pedrycz, T.-C. Ahn, “Self-organizing neural networks with fuzzy polynomial neurons”, Applied Soft Computing 2 (2002) 1–10, Elsevier B.V.

[32] S.-K. Oh, W. Pedrycz, S.-B. Roh, “Genetically optimized fuzzy polynomial neural networks with fuzzy set-based polynomial neurons”, Information Sciences 176 (2006) 3490–3519, Elsevier B.V.

[33] Box, D. E., Jenkins, G. M. “Time Series Analysis, Forcasting and Control”, California: Holden Day. 1976

[34] Angeline, P., “Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Differences”. The Seventh Annual Conference on Evolutionary Programming, March 1998.

[35] Shi,Y.H. Eberthart , R.C., “A modified particle swarm optimizer”. Proc IEEE International Conference on Evolutionary Computation. Anchorage, 1998.

[36] Amitava Chatterjee, Patrick Siarry. “Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization”. Computers & OR 33: 859-871, 2006.

[37] Clerc, M.; Kennedy, J.. "The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space". IEEE Transactions on Evolutionary Computation, vol. 6, nº 1, 2002.

[38] Mendes, R.; Cortez, P.; Rocha, M.; Neves, J.. "Particle Swarms for Feedforward Neural Network Training". IEEE Computer Society, vol.2 , 2002.

[39] Alrashidi, M.; El-Hawary, M. "A Survey of Particle Swarm Optimization Applications in Power System Operations", Taylor and Francis Ltd, Electric Power Components and Systems, vol 34, 2006.

[40] Kennedy, J., Eberhart, R. C., "A Discrete Binary Version of the Particle Swarm Algorithm". In Proceedings of the Conference on Systems, Man, and Cybernetics (SMC97), pp. 4104- 4109, 1997.

[41] Guner, A.R., Sevkli, M., "A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem", Journal of Artificial Evolution and Applications. Volume 2008 (2008).

[42] Kashan, A.H., Karimi, B., **"**A discrete particle swarm optimization algorithm for scheduling parallel machines", Pergamon Press, Inc. Tarrytown, NY, USA. 2009.

[43] Mesdker, Larry L., "Hybrid Intelligent Systems", Kluwer Academic Publishers, 1995.

[44] Haykin, S., "Neural Networks: a Comprehensive Foundation", Prentice Hall, 1999.

[45] Esmin, A. A. A., "Estudo de Aplicação do Algoritmo de Otimização por Enxame de Partícula na Resolução de Problemas de Otimização Ligados ao SEP", Tese de Doutorado, Escola Federal de Engenharia de Itajubá –EFEI, Minas Gerais, Brasil, 2005.

Anexo

Referência [22]

Lambert-Torres, G.; Borges da Silva, L.E. ; Masselli, Y.M.C.. Fuzzy Systems. In: Advanced Techniques and Technologies Facts and AI – Vol. III - Part2, por Chen-Ching Liu (Ed.), 33 p., 2009 (no prelo).

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo