UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Procedimento de Ajuste de Parâmetros de Redes RBF via PSO

Felipe Andery Reis

Itajubá, dezembro de 2014

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Felipe Andery Reis

Procedimento de Ajuste de Parâmetros de Redes RBF via PSO

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação

Área de Concentração: Matemática da Computação

Orientador: Prof. Dr. Carlos Alberto M. Pinheiro
Co-Orientador: Prof. Dr. Otávio Augusto S. Carpinteiro

Dezembro de 2014
Itajubá - MG

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Felipe Andery Reis

Procedimento de Ajuste de Parâmetros de Redes RBF via PSO

Dissertação aprovada por banca examinadora em 10 de dezembro de 2014, conferindo ao autor o título de **Mestre em Ciência e Tecnologia da Computação**

Banca Examinadora:
Prof. Dr. Carlos Alberto M. Pinheiro (Orientador)
Prof. Dr. Otávio Augusto S. Carpinteiro (Co-Orientador)
Prof. Dr. Carlos Bernardes Rosa Júnior
Prof. Dr. Carlos Henrique Valério de Moraes

Dezembro de 2014
Itajubá - MG

Agradecimentos

Primeiramente agradeço a Deus pois sem Ele não teria chegado até aqui.

Agradeço aos meus pais, Neifi e Taofena, que desde o início me incentivaram a continuar estudando e a fazer o mestrado. Ao meu irmão Lucas e à minha noiva Ana Cláudia que sempre me apoiram durante meu trabalho.

Ao professor e orientador Dr. Carlos Alberto M. Pinheiro que me conduziu até aqui e sempre estava disposto a ajudar e tirar minhas dúvidas. Ao professor e co-orientador Prof. Dr. Otávio Augusto S. Carpinteiro pelo apoio e ajuda em sua época como coordenador do curso.

Aos meus colegas e amigos, em especial ao Lucas Palhão, que de alguma forma me ajudaram durante a realização deste trabalho.

Feliz aquele que transfere o que sabe e aprende o que ensina.

Cora Coralina

Resumo

As redes neurais de funções de base radial (RBF - *Radial Basis Function*) têm sido utilizadas para a resolução de vários problemas em diversos contextos. Os parâmetros de uma rede de base radial (valores de centros, larguras e pesos) têm grande influência na sua capacidade de mapear relações entre seus dados de entrada e saída. Algumas abordagens apresentam procedimentos diversificados para determinar e otimizar estes parâmetros.

Este trabalho aborda a combinação de métodos não supervisionados com o algoritmo de enxame de partículas (PSO – *Particle Swarm Optimization*) para a determinação de parâmetros em redes RBF. O algoritmo de otimização realiza um refinamento nos valores das larguras das funções de base radial a partir de um procedimento prévio de seleção de parâmetros. Utilizando valores pré-ajustados, o algoritmo converge em um menor número de passos em relação aos parâmetros inicializados aleatoriamente. O uso da abordagem proposta proporciona uma boa melhoria na exatidão de modelos de redes RBF em aplicações de aproximação de funções, previsão de série temporal e classificação de padrões.

Abstract

Radial Basis Function (RBF) neural networks have been used to solve several problems in diverse contexts. The parameters of a radial basis network (values of centers, widths and weights) have great influence on its capacity to map the relations of their input and output data. Some approaches use different alternatives to determine and optimize these parameters.

This paper addresses the combination of unsupervised methods with the Particle Swarm Optimization (PSO) algorithm for the determination of parameters in RBF networks. The optimization algorithm performs a refinement in the width values in radial basis function from a previous procedure of parameter selection. By using pre-adjusted values, the algorithm converges with a smaller number of steps in relation to the randomly initialized parameters. The use of the proposed approach brings a good improvement to the accuracy of the RBF networks models in applications of function approximation, time series prediction, and pattern classification.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Glossário

AC	*Ant Colony*
CPN	*Counter Propagation Network*
EMQ	*Erro Médio Quadrático*
EPAM	*Erro Percentual Absoluto Médio*
FCM	*Fuzzy C-Means*
GA	*Genetic Algorithm*
GK	*Gustafson–Kessel*
LMS	*Least Mean Squares*
OLS	*Orthogonal Least Squares*
PSO	*Particle Swarm Optimization*
RAN	*Resource Allocation Network*
RBF	*Radial Basis Function*
SEQ	*Somatória dos Erros Quadráticos*
SOM	*Self Organized Maps*

1 Introdução

Este capítulo apresenta as considerações iniciais deste trabalho e menciona as estratégias de treinamento utilizadas para o ajuste de parâmetros em uma rede neural de base radial. Em seguida, apresenta a proposta e os objetivos principais do trabalho.

1.1 Considerações iniciais

As redes neurais artificiais são modelos computacionais inspirados em neurônios biológicos com aplicações em várias áreas. Uma das dificuldades encontradas ao se estabelecer um modelo neural, é garantir que o mesmo apresente uma boa capacidade de generalização independente do contexto empregado (UROLAGIN; PREMA; REDDY, 2012).

Entre as arquiteturas de redes neurais existentes, as redes de funções de base radial (RBF - *Radial Basis Function*) têm sido amplamente utilizadas na resolução de problemas envolvendo aproximação de funções (SCHILLING; JR; AL-AJLOUNI, 2001) (JIN-YUE; BAO-LING, 2013), classificação de padrões (ZHANG et al., 2007) (NOMAN; SHAMSUDDIN; HASSANIEN, 2009) (LI et al., 2013), previsões de valores em séries temporais (SUN et al., 2005) (YAN et al., 2005) (CAIQING; PEIYU, 2010) e outras aplicações. Os parâmetros livres deste tipo de rede (os valores dos centros e das larguras das funções radiais e os dados dos pesos da camada de saída) têm grande influência na sua capacidade de mapear relações entre seus dados de entrada e saída, e consequentemente devem ser ajustados apropriadamente. O ajuste destes parâmetros pode ser realizado através de procedimentos distintos. Algumas estratégias utilizadas são classificadas como empíricas, auto-organizadas e supervisionadas (HAYKIN, 2001). Também podem ser utilizados processos de aprendizagem híbridos, onde o primeiro estágio é responsável por ajustar os parâmetros que constituem as funções de base radial (valores de centros e larguras), onde são empregados métodos não supervisionados, e no segundo estágio de treinamento podem-se aplicar métodos supervisionados para o ajuste dos pesos da camada de saída da rede (BARALDI et al., 2000).

Para tratar questões relacionadas com problemas de mínimos locais e de taxas baixas na convergência nos procedimentos de treinamento, estes processos de aprendizagem são combinados com métodos de otimização baseados na computação evolucionária, tais como os algoritmos genéticos (DING et al., 2012), enxame de partículas (FATHI; MONTAZER, 2013), colônia de formigas (JUN; ZUHUA, 2014) e outros, com a finalidade de encontrar uma solução ótima no ajuste de parâmetros e garantir uma rápida convergência no treinamento da rede em questão.

1.2 Objetivo

Este trabalho tem como proposta a utilização de um algoritmo de otimização por enxame de partículas (PSO – *Particle Swarm Optimization*), para realizar o ajuste fino de parâmetros em redes RBF, cujos parâmetros iniciais são pré-selecionados através de uma abordagem não supervisionada. Diferentemente de alguns estudos realizados como em Chen, Qin e Jia (2008) e Noman, Shamsuddin e Hassanien (2009), os valores das larguras das funções de base radial serão otimizados por um algoritmo de PSO. Aplicações em aproximações de funções, previsão de séries temporais e classificação de padrões serão utilizadas para mostrar o potencial da abordagem proposta. Os resultados obtidos serão comparados com técnicas utilizadas em outros trabalhos.

1.3 Organização

Os próximos capítulos da dissertação estão organizados conforme descrito a seguir:

O Capítulo 2 traz uma revisão bibliográfica referente aos principais artigos consultados no desenvolvimento deste trabalho.

O Capítulo 3 trata dos conceitos básicos sobre redes neurais artificiais, destacando-se as redes neurais do tipo RBF. Apresenta também conceitos sobre os métodos de treinamento utilizados para o ajuste dos parâmetros livres da rede, abordando métodos de ajuste não supervisionados, como o algoritmo de agrupamento *Fuzzy C-Means* e os métodos de ajuste supervisionados, como o algoritmo de otimização por enxame de partículas.

O Capítulo 4 aborda os métodos utilizados neste trabalho para o ajuste dos parâmetros de uma rede RBF. A partir de um procedimento prévio de ajuste de parâmetros, será proposta a utilização do algoritmo de otimização por enxame de partículas para realizar um ajuste fino nas larguras das funções de ativação de uma rede RBF.

O Capítulo 5 apresenta alguns experimentos para ilustrar a aplicação da metodologia proposta.

O Capítulo 6 conclui a dissertação e propõe sugestões para trabalhos futuros.

2 Revisão Bibliográfica

Este capítulo contém uma resenha bibliográfica referente aos principais artigos consultados no desenvolvimento deste trabalho. Foram pesquisados artigos relacionados com procedimentos supervisionados no ajuste de parâmetros de redes neurais artificiais com funções de base radial, principalmente em abordagens que utilizam algoritmos genéticos (GA – Genetic Algorithm) ou exame de partículas (PSO – Particle Swarm Optimization) na otimização dos parâmetros correspondentes.

As principais estratégias utilizadas para o ajuste dos parâmetros (centros, larguras e pesos) de redes neurais de função de base radial (RBF – *Radial Basis Function*) são classificadas como empíricas, auto-organizadas e supervisionadas (HAYKIN, 2001). Algumas abordagens utilizam procedimentos que combinam métodos supervisionados e auto-organizados para o treinamento dos parâmetros da rede, e são conhecidas na literatura como estratégias híbridas. Outras estratégias utilizam combinações de procedimentos de aprendizagem supervisionada com algoritmos de otimização para a determinação ótima dos parâmetros de redes RBF.

Em Schwenker, Kestler e Palm (2001) foram discutidos alguns métodos para o treinamento de uma rede RBF, onde as estratégias de treinamento foram divididas em três fases de aprendizado. Na primeira fase os valores dos pesos da camada de saída da rede são calculados através de uma estratégia supervisionada, os dados dos centros das funções de base radial são restritos a um subconjunto de amostras de treinamento, e os valores das larguras são pré-determinados. Na segunda fase de aprendizado é sugerido que os valores dos parâmetros sejam ajustados separadamente com técnicas de treinamento. Os valores dos centros podem ser determinados através de algoritmos de agrupamento, vetor de quantização ou classificação em árvore, e os pesos da camada de saída através de algum método supervisionado, como o gradiente descendente. Na terceira fase de aprendizado é proposto que todos os parâmetros da rede RBF pré-determinados pela segunda fase sejam ajustados através de algum procedimento de otimização.

Em Guerra e Coelho (2005) sugeriu-se que os valores dos centros e das larguras das funções de base radial de uma rede RBF fossem determinados através dos algoritmos de agrupamento *K-Means* e *Fuzzy C-Means* (FCM) para fins de comparação, e os pesos da camada de saída calculados pelo método da pseudo-inversa. Em seguida, foi utilizado um algoritmo PSO para a otimização destes valores pré-determinados pelos algoritmos de agrupamento. O trabalho de Zhu e He (2006) propõe a utilização do algoritmo de agrupamento *Fuzzy C-Means* para a determinação dos valores dos centros e larguras das funções de base radial, e o método do gradiente descendente para determinar os pesos da camada de saída.

Em Chen, Qin e Jia (2008) os dados dos centros das redes de base radial foram determinados através de um método de agrupamento subtrativo e otimizados por um algoritmo de PSO. As larguras das funções de base radial foram determinadas através da distância média entre os centros calculados, e os pesos da camada de saída através do método dos mínimos quadrados (LMS - *Least Mean Squares*). Em Aik e Zainuddin (2008) foi utilizada para o treinamento de uma rede de função de base radial uma versão modificada do algoritmo de agrupamento *Fuzzy C-Means* que é baseada na medida de distância de simetria. O procedimento resultou em vantagens tal como um tempo de treinamento mais rápido, melhores exatidões nos resultados obtidos e uma arquitetura de rede reduzida. Em Ziyang et al. (2008) foi proposto um novo método de aprendizagem para as redes RBF baseado no agrupamento *Fuzzy C-Means* e no algoritmo de otimização por colônias de formigas (AC - *Ant Colony*). Os centros foram determinados pelo algoritmo FCM e os pesos da camada de saída foram determinados pelo método dos mínimos quadrados. O AC foi introduzido para otimizar dois parâmetros do algoritmo FCM, como o expoente de ponderação (*weigthing expoent*) que impacta no desempenho do algoritmo, e o parâmetro relacionado ao número de clusters associados com a capacidade de interpolação da rede RBF resultante.

O trabalho de Noman, Shamsuddin e Hassanien (2009) sugeriu que os valores dos centros e das larguras das funções de base radial fossem determinados através do algoritmo de agrupamento *K-Means*, e dados dos pesos estabelecidos via método dos mínimos quadrados, sendo estes parâmetros depois otimizados por um algoritmo de PSO. Em Li, Dong e Zhang (2009) foi utilizado o método de agrupamento FCM para determinar os centros das funções de base radial, e o método da pseudo-inversa utilizado para a determinação dos pesos da camada de saída da rede. O método proposto demonstrou rápida convergência e um erro relativo médio menor em comparação a utilização do algoritmo de agrupamento *K-Means*. Em Liu, Chen e Song (2009) uma nova estrutura de rede RBF foi

proposta a partir da utilização de um algoritmo de agrupamento *Fuzzy C-Means* baseado em algoritmo genético, cuja aplicação estava relacionada com o diagnóstico de falhas em giroscópios e acelerômetros. O algoritmo *Fuzzy C-Means* combinado com o GA garantiu a obtenção de valores otimizados para os centros dos clusters. Em Wang, Xie e Sun (2009) foi proposta a utilização de um algoritmo de PSO para calcular e otimizar os pesos da camada de saída de redes RBF. Os valores dos centros e das larguras das funções de base radial foram determinados via algoritmo de agrupamento *K-Means*. Em Chun-tao, Kun e Li-yong (2009) foi proposto um procedimento para treinamento de uma rede RBF com a utilização de um algoritmo de PSO e um método de alocação de recursos (RAN - *Resource Allocation Network*). O número de funções de base radial da camada escondida da rede foi determinado pelo RAN, e todos os parâmetros livres da rede foram otimizados via PSO. Em Esmaeili e Mozayani (2009) os valores dos centros, números de centros e valores das larguras das funções de base radial de uma rede RBF foram determinados via algoritmo de PSO, e os dados dos pesos da camada de saída determinados por método de decomposição em valores singulares.

Em Chen e Qian (2009), Su-wei (2010) e Lianghai e Yiming (2010) uma rede neural RBF foi projetada e treinada através de um algoritmo de PSO para determinar todos os parâmetros livres da rede (centros, larguras e pesos). Em Changbing e Wei (2010) os valores dos centros e das larguras das funções de base radial de uma rede RBF foram determinados por um algoritmo genético. Nos trabalhos de Ming, Bin e Zhong (2010), Qing-wei, Zhi-Hai e Jian (2010) e Pan et al. (2011) também foi utilizado o algoritmo genético para determinar todos os parâmetros livres (centros, larguras e pesos) da rede RBF. Em Caiqing e Peiyu (2010) os dados dos centros das funções de base radial de redes RBF foram determinados pelo método de agrupamento *K-Means*, os valores das larguras foram fixadas em um valor e os pesos da camada de saída da rede foram determinados via algoritmo PSO. Em Sun e Li (2010) foi utilizado um modelo de rede RBF para previsão e identificação de mercado de ações. O algoritmo de aprendizado utilizado para o treinamento da rede RBF foi dividido em dois estágios. No primeiro estágio, foi utilizada uma estratégia não supervisionada onde o algoritmo de agrupamento *K-Means* foi escolhido para a determinação do número e posições iniciais dos centros e larguras das funções de base radial. Os pesos iniciais foram calculados pelo método dos mínimos quadrados. No segundo estágio, utilizou-se de uma estratégia supervisionada onde um algoritmo de treinamento baseado no método do gradiente foi utilizado para o ajuste dos parâmetros pré-determinados no estágio anterior. Em Lin e Wu (2011) foi proposto um algoritmo aprendizagem para melhorar o desempenho de uma rede RBF através da combinação de

um algoritmo supervisionado e de um não supervisionado. Foi desenvolvido um algoritmo de aprendizagem de dois passos. No primeiro passo, os valores dos centros das funções de base radial são selecionados a partir de uma análise dos dados de entrada da rede através um mapa auto-organizável (SOM - *Self Organized Maps*). Deste processamento foram extraídos também os possíveis valores das larguras que irão compor as funções de base radial da rede RBF em questão. No segundo passo, os valores dos centros determinados via SOM foram avaliados através de um algoritmo supervisionado, e os dados dos pesos calculados pelo método dos mínimos quadrados. Em Golbabai e Safdari-Vaighani (2011) as informações dos centros das funções de base radial foram determinadas a partir de um esquema de quantização vetorial, e os pesos da camada de saída da rede determinados a partir dos mínimos quadrados. Os valores das larguras foram determinados a priori a partir do método dos p-vizinhos mais próximos, e escalados através de um fator, sendo este otimizado por um algoritmo genético. Em Alzohairy (2011) foi proposta uma estratégia para o controle adaptativo de sistemas não lineares utilizando uma rede de função de base radial. O processo de treinamento para o ajuste dos parâmetros da rede RBF foi dividido em duas fases. Na primeira fase, utilizou-se uma estratégia não supervisionada utilizando o algoritmo de agrupamento *K-Means* para a determinação dos valores iniciais dos dados dos centros, e o método dos p-vizinhos mais próximos para o cálculo das larguras das funções de base radial. Na segunda fase, utilizou uma estratégia supervisionada através do método do gradiente descendente para otimizar os valores dos centros e das larguras pré-determinados, e depois estimar os valores dos pesos da camada de saída da rede. Em Tao, Jianping e Bingxin (2011) foi utilizada uma combinação de dois algoritmos de aprendizagem supervisionada na estimação dos dados dos centros e das larguras das funções de base radial, e os pesos da camada de saída foram ajustados através do método dos mínimos quadrados. O procedimento proposto foi aplicado no diagnóstico de câncer de pulmão através de redes RBF. Delican, Ozyilmaz e Yildirim (2011) apresentou diferentes métodos baseados nos algoritmos evolucionários para a determinação dos parâmetros livres de uma rede RBF propostos para o diagnóstico da doença de Parkison. Algoritmos de otimização por enxame de partículas, GA e o algoritmo baseado em colônia de abelhas foram utilizados para o treinamento da rede. Através dos experimentos foi demonstrado que o algoritmo colônia de abelhas obteve o melhor resultado em termos de tempo de processamento e exatidão para o diagnóstico da doença.

Em Niros e Tsekouras (2012) foi proposto um método de agrupamento híbrido para estimar os valores dos centros e das larguras das funções de base radial. O procedimento baseou-se na combinação de técnicas de agrupamento via conjuntos *fuzzy* e conjuntos

convencionais (*crisp*), onde os pesos da camada de saída da rede foram determinados através do método dos mínimos quadrados. Em Wu e Liu (2012) uma rede neural RBF foi utilizada em um sistema de previsão de consumo de combustível de veículos. Os dados dos centros das funções de base radial foram determinados pelo algoritmo de agrupamento *K-Means*. Em seguida, os valores das larguras foram determinados pelo método do p-vizinhos mais próximos e os pesos da camada de saída determinados pelo algoritmo LMS. Ao final, a rede RBF proposta foi comparada com uma rede multicamadas, com um melhor desempenho resultante na rede de base radial. Em Xue-qin e Tong-di (2012) foi utilizada uma rede RBF em um sistema de avaliação da qualidade de água. Os dados dos centros e dos desvios das funções de base radial foram inicialmente calculados pelo algoritmo de agrupamento *K-Means*, e os pesos da camada de saída foram determinados diretamente pelo método dos mínimos quadrados. Em seguida todos os parâmetros da rede foram otimizados via algoritmo PSO. Em Cui, Wang e Yang (2012) uma rede RBF foi utilizada em uma aplicação para o diagnóstico de falhas. Os parâmetros da rede como os valores dos centros, larguras e o número de funções de base radial foram determinados por um algoritmo de PSO, e os pesos obtidos diretamente pelo método dos mínimos quadrados. Em Yu-liang et al. (2012) foi utilizada uma rede RBF para o diagnóstico de falhas na geração de uma unidade geradora. Os dados dos centros, larguras e pesos da rede RBF foram determinados e otimizados por um modelo PSO-GA, onde o algoritmo genético foi utilizado com o objetivo de melhorar o desempenho do algoritmo de otimização por enxame de partículas, aumentando a velocidade de treinamento e a precisão de convergência da rede. Em Ding et al. (2012) foi proposta a utilização de um procedimento híbrido via GA para otimização dos parâmetros (centros, larguras e pesos) de redes RBF. Maruzuki, Ishak e Setumin (2012) utilizaram uma rede RBF otimizada via PSO para o reconhecimento de placas de carros na Malásia, Ji, Zhou e Yu (2012) para aplicação em um determinado sistema de controle, Shuzhi, Jie e Xianwen (2012) para auxiliar nas estimativas de pureza em um processo de produção de cloreto de vinil, e Song et al. (2012) para previsão de fluxo de tráfico de dados em sistemas de comunicação.

Em Jin-Yue e Bao-Ling (2013) os dados dos centros foram determinados pelo algoritmo de agrupamento *K-Means* e os valores dos desvios determinados pela distância máxima entre os centros selecionados. Os pesos da camada de saída foram determinados pelo método dos mínimos quadrados. Em Kayhan, Ozdemir e Eminoglu (2013) foram revisados alguns métodos para inicialização de uma rede RBF e apresentados procedimentos para construção de redes através de um algoritmo híbrido. No primeiro passo do algoritmo proposto, o número e os valores das posições dos centros de uma rede RBF foram de-

terminados utilizando um algoritmo como o CPN (*Counter Propagation Network*). No segundo passo os parâmetros relacionados com as larguras foram determinados através de algoritmos de agrupamentos como o GK (*Gustafson–Kessel*). No terceiro passo os valores dos pesos da rede foram otimizados através do método do gradiente descendente. Em Ozdemir e Eminoglu (2013) também foi proposto um algoritmo de treinamento híbrido para construção de redes RBF. No primeiro estágio do treinamento foi determinado o número e a localização dos centros das funções de base radial através de um algoritmo OLS (*Orthogonal Least Squares*). Em seguida, todos os parâmetros da rede foram ajustados pelo algoritmo GK. O segundo estágio foi dividido em dois passos. O primeiro passo realizou um ajuste nos pesos da camada de saída da rede utilizando o algoritmo de *Levenberg-Marquardt* e os outros parâmetros (centros e larguras) foram fixados em valores pré-determinados. Posteriormente, os dados dos centros e das larguras das funções de base radial foram ajustados pelo algoritmo do gradiente descendente. Em Chang (2013) foi proposto um método híbrido que combina a utilização do algoritmo dos mínimos quadrados ortogonais com algoritmo genético na construção de modelos via redes RBF para a previsão de geração de energia de origem eólica. O algoritmo OLS foi utilizado para determinar o número ótimo de nós da rede. Em seguida, os valores dos centros, das larguras das funções de base radial e dos pesos da camada de saída da rede foram ajustados através de um algoritmo genético. Li et al. (2013) utilizaram uma rede RBF otimizada via PSO para avaliação de crédito pessoal em sistemas bancários. Em Li, Cong e Zhang (2013) um modelo de rede otimizado por um algoritmo genético foi construído para a utilização no processo de tratamento de esgoto.

Em Jun e Zuhua (2014) um algoritmo de otimização por colônia de formigas foi utilizado para otimizar os parâmetros livres de uma rede RBF para a previsão de tráfico de dados em redes de comunicação. O algoritmo garantiu uma boa convergência dos parâmetros da rede RBF e melhorou sua capacidade de generalização.

3 Revisão de Conceitos

Este capítulo aborda os conceitos básicos sobre redes neurais artificiais destacando as redes neurais de função de base radial (RBF – Radial Basis Function). Apresenta também algumas técnicas de treinamento utilizadas para o ajuste dos parâmetros livres da rede, destacando os métodos de ajuste não supervisionados, como o algoritmo de agrupamento Fuzzy C-Means e os métodos de ajuste supervisionados, como o algoritmo de otimização por exame de partículas (PSO – Particle Swarm Optimization).

3.1 Redes Neurais Artificiais

3.1.1 Introdução

As redes neurais artificiais são modelos computacionais inspirados em neurônios biológicos. Elas possuem a capacidade de armazenar informações através de interligações conhecidas como sinapses (HAYKIN, 2001). O processo de aprendizagem é realizado através de algoritmos de treinamento que ajustam os parâmetros livres da rede de forma a alcançar uma determinada resposta desejada. O primeiro modelo de neurônio artificial foi introduzido por McCulloch e Pitts (1943), e o diagrama da Figura 1 ilustra a estrutura deste modelo.

O modelo do neurônio artificial é constituído dos seguintes elementos básicos (HAYKIN, 2001):

1. Sinais de entrada, referenciados pelo vetor de entrada $x = [x_0, x_1, x_2, \ldots, x_n]$, responsáveis por trazer informações do ambiente externo;

2. Pesos sinápticos, referenciados pelo vetor de peso $w = [w_0, w_1, w_2, \ldots, w_n]$, responsáveis pela ponderação das variáveis de entrada;

3. Somador ou combinador linear $\sum$, responsável por gerar um potencial de ativação

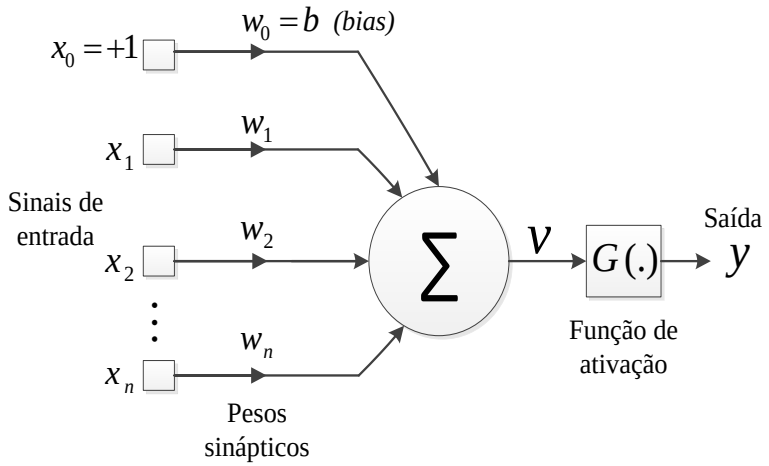

Figura 1: Modelo de um neurônio artificial.

na entrada da função de ativação correspondente, através do somatório dos sinais de entrada que são ponderados pelos respectivos pesos sinápticos;

4. Função de ativação $G(.)$, responsável por limitar o sinal de saída produzido pelo combinador linear;

5. Sinal de saída y, corresponde ao valor final produzido pelo modelo em relação a um determinado padrão de entrada aplicado.

Um valor de *bias*, representado por "*b*", pode ser aplicado externamente, sendo responsável por alterar o valor da entrada da função de ativação. O sinal de entrada x_0, cujo valor é fixado em 1, e o peso sináptico w_0 que se refere ao próprio valor de b, são adicionados ao modelo. Assim, o modelo de neurônio pode ser representado matematicamente através das equações (3.1) e (3.2). As funções de ativação $G(.)$, que definem a saída do neurônio, são classificadas em vários tipos e estão detalhadas nas referências (HAYKIN, 2001) e (SILVA; SPATTI; FLAUZINO, 2010).

$$v = \sum_{i=0}^{n} w_i x_i \tag{3.1}$$

$$y = G(v) \tag{3.2}$$

Desde o primeiro modelo matemático inspirado em neurônios biológicos, vários pes-

quisadores realizaram estudos no desenvolvimento de outros modelos computacionais, resultando em uma série de arquiteturas de redes e algoritmos de aprendizagem. As arquiteturas das redes definem como os neurônios estão estruturados e como se relacionam com as estratégias de aprendizagem utilizadas nos treinamentos das redes.

3.1.2 Tipos de Arquitetura

Determinados tipos de arquiteturas de redes neurais podem ser divididos em três camadas: a camada de entrada, que conecta a rede ao meio externo responsável por trazer informações (padrões ou exemplos); a camada escondida ou oculta, responsável pelo processamento da rede; e a camada de saída, responsável pela apresentação dos resultados. Estas arquiteturas podem ser divididas em classes que se diferenciam pela maneira como os neurônios são interligados e pela quantidade de camadas existentes. Entre elas estão as redes alimentadas adiante (*feedforward*) com camada única, redes alimentadas adiante com múltiplas camadas e redes recorrentes (SILVA; SPATTI; FLAUZINO, 2010).

As redes *feedforward* de camada única são constituídas de uma camada de entrada e uma camada de saída composta por modelos de neurônios. Os sinais são propagados da entrada para a saída e nunca vice-versa. Os principais tipos de rede que se enquadram nesta arquitetura são o *Perceptron* e o *Adaline*. Já as redes *feedforward* de múltiplas camadas são constituídas por uma camada de entrada, uma ou mais camadas ocultas, e uma camada de saída. São empregadas em diversos tipos de problemas, como aproximação de funções, classificação de padrões e outros. Os principais tipos de rede que se enquadram nesta arquitetura são o *Perceptron* de múltiplas camadas (redes MLP) e as redes de função de base radial (redes RBF).

As redes recorrentes podem ser constituídas de camadas de neurônios, cujas saídas são sinais de entrada para outros neurônios, ou seja, existe um sistema de realimentação de informações. Podem ser empregadas em problemas de previsão de séries temporais, identificação de sistemas e outros. Os principais tipos de rede que se enquadram nesta arquitetura são as redes de *Hopfield* e as redes de *Perceptron* de múltiplas camadas com realimentação.

3.1.3 Processos de Aprendizagem

Uma questão importante para uma rede neural é sua habilidade em aprender (HAYKIN, 2001). O processo de treinamento consiste em ajustar os parâmetros livres da rede através

de algum algoritmo de aprendizagem, de modo que, para qualquer padrão ou exemplo aplicado à entrada da rede, ela seja capaz de produzir uma saída próxima daquela desejada. Segundo Haykin (2001), a aprendizagem de uma rede neural pode ser dividida em dois paradigmas: a aprendizagem supervisionada e a não supervisionada.

Na aprendizagem supervisionada a rede neural trabalha com a supervisão de um "instrutor" que conhece o ambiente computacional e a relação existente entre a entrada e a saída da rede. Durante o processo de treinamento, para uma determinada amostra aplicada à entrada da rede, o instrutor é capaz de informar qual será sua saída desejada. Os parâmetros livres da rede devem ser ajustados por um algoritmo de aprendizado até que o erro gerado pela rede seja o menor possível, ou seja, deve-se minimizar a diferença entre a resposta fornecida pela rede e a resposta desejada.

Na aprendizagem sem instrutor, ou não supervisionada ou auto-organizada, a relação entre a entrada e saída da rede não é conhecida devido a ausência de um "instrutor". Nesse caso, a rede deve ser capaz de encontrar padrões e relações existentes entre os dados de entrada e, através de um algoritmo de aprendizado, ajustar os parâmetros livres. Estas relações entre os dados podem ser identificadas através de técnicas como algoritmos de agrupamentos, sendo assim possível descobrir as similaridades ou diferenças entre os padrões existentes.

3.2 Redes Neurais de Base Radial

3.2.1 Introdução

As redes neurais de funções de base radial têm sido utilizadas para a resolução de vários problemas em diversos contextos, sendo originariamente aplicadas em problemas de interpolação de funções (BISHOP, 1995). As RBF são consideradas redes do tipo *feed-forward* multicamada, onde o fluxo de informações é sempre no sentido da entrada para a saída da rede. Diferentemente das redes neurais do tipo MLP, as redes do tipo RBF possuem uma única camada escondida, cujos neurônios são constituídos de funções de base radial (POWELL, 1987). Segundo Haykin (2001), o projeto de uma rede neural do tipo RBF é visto como um problema aproximação de uma superfície em um espaço n-dimensional. O aprendizado neste tipo de rede é equivalente a encontrar uma superfície no espaço que forneça o melhor ajuste para os dados de treinamento. A generalização é equivalente a usar esta superfície para a interpolação de dados que não foram utilizados na etapa de treinamento. É interessante ressaltar algumas diferenças entre as redes RBF

e as redes de múltiplas camadas. Segundo Haykin (2001) e Bishop (1995), os dois tipos de rede diferem em alguns aspectos:

- Uma rede de base radial possui uma única camada escondida, enquanto uma rede de múltiplas camadas pode ter uma ou mais camadas escondidas;
- Nas redes RBF e nas redes MLP utilizadas em problemas de regressão, cada unidade da camada oculta desempenha uma transformação não linear do espaço de entrada enquanto a camada de saída da rede é linear. Nas redes MLP utilizadas em classificação de padrões, normalmente as unidades das duas camadas desempenham transformações não lineares.
- As redes MLP constroem aproximações globais de um mapeamento de entrada-saída não linear, enquanto nas redes RBF as aproximações são locais.
- Em uma rede RBF, o treinamento normalmente é realizado em dois estágios, utilizando-se de métodos não supervisionados na primeira camada, e de métodos supervisionados na segunda camada. Os parâmetros de uma rede MLP são geralmente determinados ao mesmo tempo utilizando-se de uma estratégia de treinamento supervisionada.

3.2.2 Estrutura típica de uma rede neural RBF

A estrutura típica de uma rede de função de base radial possui uma camada de entrada que está associada diretamente às informações de entrada da rede, uma única camada escondida constituída por funções de ativação de base radial que realizam uma transformação não linear do espaço de entrada, e uma camada de saída linear que fornece a resposta ao padrão aplicado nas entradas da rede (HAYKIN, 2001). A Figura 2 ilustra a estrutura básica de uma rede RBF.

Diferentemente das redes neurais de múltiplas camadas onde as informações das unidades escondidas são determinadas pelo produto interno entre o vetor de entrada e o vetor de peso daquela unidade, nas redes neurais do tipo RBF as informações das funções da camada escondida são determinada pela distância, normalmente euclidiana, entre o vetor de entrada e o centro daquela unidade. Ou seja, as variáveis de entrada constituem os dados utilizados pelas funções de ativação da rede, onde os pesos que conectam a camadas de entrada e escondida podem ser considerados unitários. O processamento da camada escondida é baseado no teorema de Cover (HAYKIN, 2001), referente a discriminação de

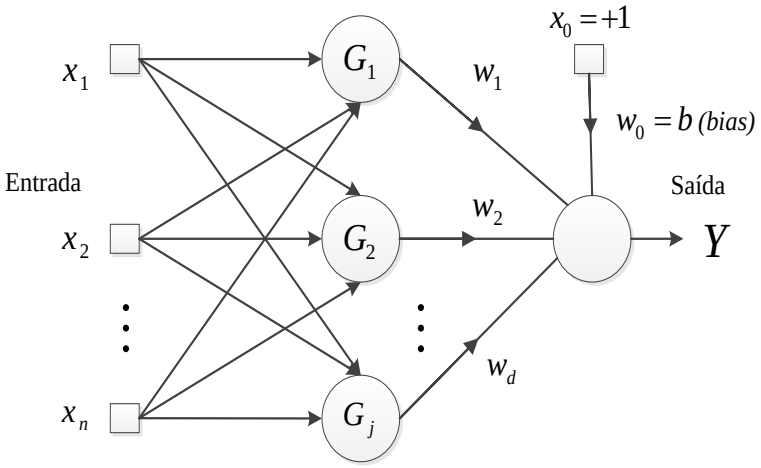

Figura 2: Estrutura de uma rede RBF.

padrões. Neste teorema um problema não linearmente separável pode ser transformado em um problema linearmente separável através de uma transformação não linear que mapeia um espaço para outro de maior dimensão.

A camada escondida de uma rede RBF geralmente é constituída por funções de ativação Gaussianas (BISHOP, 1995) representadas pela equação (3.3). Outras funções de ativação de base radial, que também podem ser consideradas para redes RBF, são aquelas representadas por (3.4) e (3.5), denominadas multiquadrática e multiquadrática inversa, respectivamente.

$$y_j = G_j(x_n) = e^{-\left(\frac{\|x_n - c_j\|}{2\sigma_j^2}\right)^2} \tag{3.3}$$

$$y_j = G_j(x_n) = \sqrt{(\|x_n - c_j\|)^2 + \sigma_j^2} \tag{3.4}$$

$$y_j = G_j(x_n) = \frac{1}{\sqrt{(\|x_n - c_j\|)^2 + \sigma_j^2}} \tag{3.5}$$

Nas funções (3.3), (3.4) e (3.5), x_n representa o vetor de entrada da rede, c_j denota o j-ésimo valor central da função associada, σ_j simboliza o valor da largura (desvio padrão) ou do campo receptivo da função, e a norma $\|x_n - c_j\|$ representa a distância euclidiana entre a n-ésima entrada e o j-ésimo valor central da função de base radial correspondente.

Uma representação expandida da função gaussiana é ilustrada por (3.6), onde G_j representa j-ésima função de base radial, $x \in \Re^n$ representa o vetor das variáveis de entrada $x = [x_1, x_2, x_3, \ldots, x_n]$, $c \in \Re^n$ denota os valores dos centros $c = [c_1, c_2, c_3, \ldots, c_n]$ da j-ésima função de base radial e $\sigma \in \Re^n$ representa os valores das larguras $\sigma = [\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_n]$ da j-ésima função de base radial. Esta representação será utilizada neste trabalho.

$$y_j = G_j(x) = e^{-\left(\frac{(x_1 - c_{1j})}{2\sigma_{1j}^2} + \cdots + \frac{(x_n - c_{nj})}{2\sigma_{nj}^2}\right)} \tag{3.6}$$

A informação de saída Y de uma rede RBF é representada por (3.7), podendo existir mais de uma saída conforme a aplicação em questão. A camada de saída não contém funções de ativação, ela é considerada uma combinação linear que é dada pela soma ponderada dos valores gerados pelas funções de ativação das unidades escondidas, que são multiplicados pelos correspondentes pesos $(w_1, w_2, w_3, \ldots, w_d)$ da camada de saída da rede. Um valor de *bias* unitário multiplicado por um peso "w_0" é adicionado à ponderação resultante para compensar eventuais diferenças entre o valor médio do conjunto de dados das funções de ativação e o valor médio sobre a saída resultante (BISHOP, 1995).

$$Y = w_0 + \sum_{j=1}^{d} w_j y_j \tag{3.7}$$

3.2.3 Procedimentos de Treinamento de Redes RBF

O processo básico de treinamento de uma rede de base radial consiste em determinar todos os parâmetros livres da mesma (valores dos centros, larguras e pesos). Alguns procedimentos adotados para o treinamento de redes RBF podem tornar o processo de aprendizado mais rápido em relação aos métodos para redes MLP (HAYKIN, 2001).

Uma das estratégias citadas consiste em escolher os valores dos centros das funções radiais aleatoriamente a partir dos dados do conjunto de treinamento. As informações dos valores das larguras das funções podem ser estimadas pela relação entre a distância máxima dos centros escolhidos pela raiz quadrada do número de funções de ativação adotadas. A ideia desta abordagem consiste em distribuir uniformemente as funções de base radial no espaço de dados do problema, cujo objetivo é mapear adequadamente os dados de entrada e obter boa capacidade de generalização da rede. Os pesos da camada de saída são então calculados diretamente através do método da pseudo-inversa ou dos

mínimos quadrados.

Outra estratégia usual consiste em utilizar um processo de treinamento, onde na primeira etapa se utiliza um algoritmo de agrupamento de dados (*K-Means*, por exemplo), cujo objetivo consiste em agrupar os valores dos centros das funções de base radial de forma organizada, e estimar a partir destes agrupamentos resultantes os valores referentes às larguras das funções associadas. Na segunda etapa de treinamento, os pesos da camada de saída da rede são ajustados pelo método dos mínimos quadrados ou outro correspondente. Outras estratégias não supervisionadas são citadas em Bishop (1995), como a utilização do algoritmo dos mínimos quadrados ortogonais (OLS – *Orthogonal Least Square*) e de mapas auto-organizáveis de *Kohonen*.

Em estratégias supervisionadas o método do gradiente descendente é um método usualmente empregado para o ajuste de parâmetros (centros, desvios padrões e pesos) de redes RBF. A informação do erro entre a informação de saída da rede em questão e o padrão desejado para a mesma é utilizada neste processo de treinamento. No capítulo de revisão bibliográfica foram citadas as principais abordagens utilizadas para o ajuste de parâmetros de redes RBF, a Tabela 1 resume alguns procedimentos referenciados.

Tabela 1: Procedimentos não supervisionados e supervisionados.

Aprendizagem não supervisionada	**Aprendizagem supervisionada**
Métodos Empíricos	
Agrupamentos (*K-Means, C-Means*, etc)	Gradiente Descendente
Alocação de Recursos	Algoritmos Genéticos
Mapas Auto-Organizáveis	Otimização por Enxame de Partículas
Mínimos Quadrados Ortogonais	

3.2.4 Exemplo de Modelagem de uma Função Ou Exclusiva

Com a finalidade de ilustrar o processamento de informações em uma rede RBF, utilizou-se um contexto de classificação de padrões simples como a modelagem de uma função ou exclusiva, muito citada em várias referências (HAYKIN, 2001). Os padrões de dados deste exemplo são (1,1), (1,0), (0,1) e (0,0), que representam o espaço de entrada bidimensional dos dados binários da função em questão. O objetivo do exemplo é construir um classificador que produza a saída "0" para os padrões (1,1) ou (0,0) e a saída "1" para os padrões (1,0) ou (0,1).

A rede RBF que serve de modelo deste exemplo é constituída de duas entradas x_1 e x_2, duas unidades escondidas G_1 e G_2 (cujas funções são do tipo Gaussianas), e uma

unidade de saída linear. A estrutura da rede RBF utilizada neste exemplo está ilustrada na Figura 3. A relação entre a entrada e saída da rede é definida por (3.8).

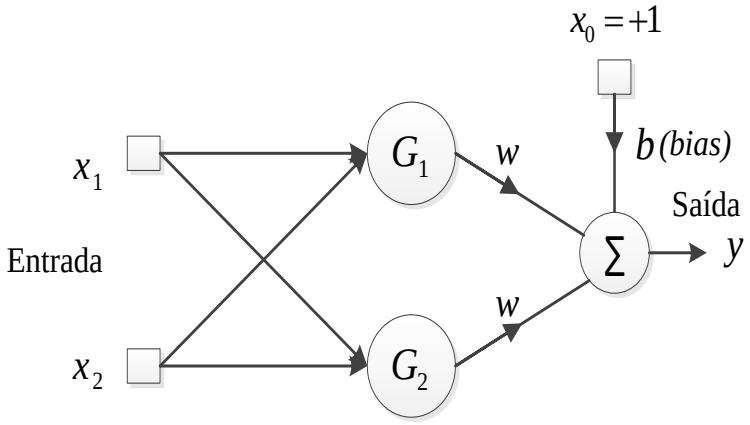

Figura 3: Estrutura de rede RBF com o objetivo de representar uma função XOR.

$$y(x) = \sum_{i=1}^{2} wG_i\|x - c_i\| + b \tag{3.8}$$

Neste exemplo, para as funções de base radial serão atribuídos valores de larguras unitários, cujos valores dos centros em 0,25 e 0,75 foram escolhidos aleatoriamente. Para os vetores de entrada $x_1 = [0; 0; 1; 1]$ e $x_2 = [0; 1; 0; 1]$ da rede, têm-se os seguintes valores de saída das funções gaussianas associadas: $G = [0,88251 \quad 0,3247; 0,5352 \quad 0,5353; 0,5352 \quad 0,5353; 0,3247 \quad 0,8825]$. Com estes valores e com os dados do vetor de saída $Y = [0; 1; 1; 0]$, calcula-se os coeficientes (W) dos pesos da rede em questão através do método dos mínimos quadrados de acordo com a equação (3.9), onde o símbolo "^{T}" denota a matriz transposta e "$^{-1}$" a matriz inversa:

$$G = \begin{bmatrix} 0,8825 & 0,3247 & 1 \\ 0,5353 & 0,5353 & 1 \\ 0,5353 & 0,5353 & 1 \\ 0,3247 & 0,8825 & 1 \end{bmatrix}; Y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix};$$

$$W = [G^T G]^{-1} Y = \begin{bmatrix} -7,3206 \\ -7,3206 \\ +8,8375 \end{bmatrix} \tag{3.9}$$

Exemplificando numericamente o processamento dos dados da rede RBF obtida, tem-se para $x_1 = 1$ e $x_2 = 1$, $y = 8,8375 - 7,3206 * 0,3247 - 7,3206 * 0,8825 = 0,0004$ (que corresponde ao nível lógico "0"). De forma similar para $x_1 = 0$ e $x_2 = 0$, tem-se $y = 0,0004$. Para $x_1 = 0$ e $x_2 = 1$, obtém-se $y = 1,0006$ (que corresponde ao nível lógico "1"). Idem para $x_1 = 1$ e $x_2 = 0$, com $y = 1,0006$. Assim, a rede RBF resultante modela adequadamente a função ou exclusiva tomada como exemplo.

3.3 Algoritmo de Otimização por Enxame de Partículas

3.3.1 Introdução

O algoritmo de otimização por exame de partículas (PSO) foi introduzido por Kennedy e Eberhart (1995) como um método supervisionado a ser utilizado para a otimização de funções não lineares contínuas, onde é possível encontrar soluções para um determinado problema através de um processo iterativo. O PSO tem sido empregado para resolução de problemas em várias áreas do conhecimento, e uma relação detalhada das aplicações usuais pode ser encontrada em Poli (2007).

Algumas técnicas de computação evolucionária tais como os algoritmos genéticos, imunológicos, entre outros, são inspirados em aspectos da evolução na natureza. Nos algoritmos genéticos, por exemplo, a população de indivíduo, que representam as soluções para determinado problema, é modificada de acordo com uma regra de sobrevivência pelo mais apto, através de operadores genéticos como mutação e reprodução. Nos algoritmos de enxame de partículas a inspiração está baseada no comportamento social de determinadas espécies como bando de pássaros, colônia de insetos, cardumes de peixes, etc. Em um algoritmo de PSO os indivíduos, conhecidos como partículas, apresentam um comportamento social através da interação entre eles próprios, e através da troca de experiências com o grupo (SHI; EBERHART, 1998). Fazendo uma analogia através do comportamento social de bando de pássaros, em um algoritmo de PSO o enxame de partículas (bando de pássaros) é colocado em um espaço de busca (área percorrida) de um problema, em busca de uma solução ótima (por exemplo, a analogia por busca de alimentos), e cada partícula (pássaro) em sua posição atual avalia sua aptidão, ou seja, qual é sua proximidade em relação a melhor solução (encontrar alimento), através de uma função objetivo que modela o problema em questão. Para determinar seus deslocamentos no espaço de busca, as partículas trocam experiências entre si e entre o grupo, dividindo alguns aspectos como sua melhor posição local (melhor aptidão obtida pela partícula) e a melhor posição encontrada

pelo enxame (melhor aptidão obtida pelo grupo). A Figura 4 ilustra o deslocamento de algumas partículas em um espaço bidimensional em busca da solução ótima.

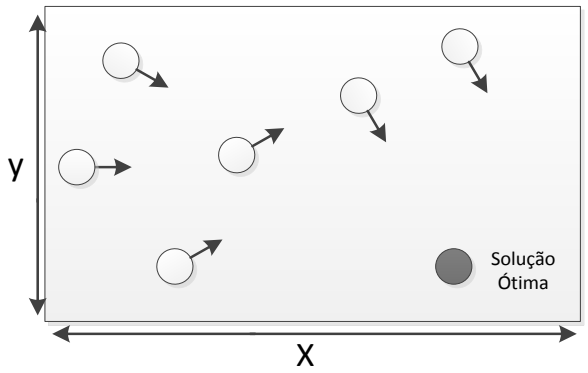

Figura 4: Exame de partículas em busca da solução ótima.

3.3.2 Funcionamento de um Algoritmo de PSO

Em um algoritmo de PSO todas as partículas que representam um enxame são candidatas à solução de um problema de otimização em um espaço de busca n-dimensional, sendo devidamente avaliadas por uma função objetivo, conhecida como *fitness*. A ideia do algoritmo é que cada partícula percorra o espaço de busca com a sua informação de posição e velocidade atualizada a cada passo, utilizando tanto os resultados individuais obtidos a cada avaliação, quanto os resultados coletivos do enxame. O algoritmo é interrompido ao encontrar uma solução ótima determinada pelo melhor *fitness* que atenda a especificação do problema, ou até se atingir de um número máximo de passos de execução do algoritmo.

Considerando o espaço de busca n-dimensional e o número de partículas do enxame igual a i têm-se que: $p_i = [p_{i1}, p_{i2}, p_{i3}, \ldots, p_{in}]$ é o vetor posição da i-ésima partícula no espaço de busca; $v_i = [v_{i1}, v_{i2}, v_{i3}, \ldots, v_{in}]$ é a velocidade da i-ésima partícula; a posição individual da partícula i que possui o melhor valor de *fitness* é conhecida como $pBest_i$; a posição global que possui o melhor valor de *fitness* do enxame é denotada como *gBest*. As equações (3.10) e (3.11) definem a atualização da velocidade e da posição de cada partícula no passo t do algoritmo, cujo comportamento pode ser representado através da Figura 5.

$$v_i(t+1) = v_i(t) + a_1 r_1(pBest_i - p_i(t)) + a_2 r_2(gBest - p_i(t)) \tag{3.10}$$

$$p_i(t+1) = p_i(t) + v_i(t+1) \tag{3.11}$$

Onde:

- a_1 e a_2 são fatores de aceleração, também conhecidos como parâmetros cognitivos e sociais, que representam a aglutinação que existe entre as partículas;
- r_1 e r_2 são números gerados aleatoriamente dentro do intervalo $[0, 1]$;
- v_i(t+1) é a nova velocidade da partícula no passo t+1;
- p_i(t+1) é a nova posição da partícula no passo t+1;
- v_i(t) é a velocidade atual da partícula no passo t;
- p_i(t) é a posição atual da partícula no passo t.

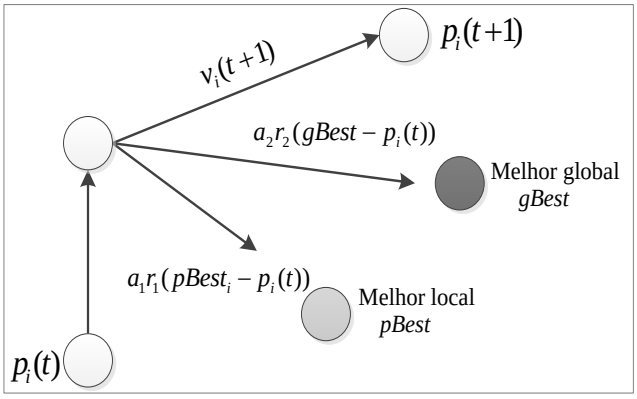

Figura 5: Ilustração do deslocamento de uma partícula no espaço de busca.

Alguns parâmetros do PSO podem ser previamente estabelecidos a fim de melhorar o desempenho do algoritmo em questão. Por exemplo, a velocidade de uma partícula é limitada para prevenir que trajetórias fora do espaço de busca sejam percorridas (POLI; KENNEDY; BLACKWELL, 2007). Para uma dada dimensão N as velocidades das partículas podem ser limitadas na faixa $[vMin_i; vMax_i]^N$, onde se o valor de $vMax$ é alto as partículas

realizam uma busca global que pode ultrapassar eventuais soluções ótimas, e se o valor de *vMax* é baixo as partículas podem não ser capazes de mover o suficiente para alcançar boas soluções (EBERHART; SHI, 2001). Com a finalidade de obter um melhor controle na velocidade das partículas algumas alterações na equação de velocidade (3.10) foram propostas e serão discutidas nas próximas seções. De modo similar o espaço de busca da partícula é limitado por uma faixa $[pMin_i; pMax_i]^N$ em uma dimensão N.

Os coeficientes de aceleração a_1 e a_2 são responsáveis por ponderar a influência sobre o movimento da partícula em relação a sua melhor posição individual ($pBest_i$), e a melhor posição do enxame (*gBest)*. Valores altos para estes parâmetros fazem com que as partículas tenham movimentos abruptos em direção às regiões alvo, e valores baixos fazem com que elas se distanciem dessas regiões. O valor usualmente utilizado para estes coeficientes é 2 (EBERHART; SHI, 2001) (HU; SHI; EBERHART, 2004).

O tamanho do enxame define a quantidade de partículas utilizadas em busca de uma solução ótima. Ele geralmente é escolhido empiricamente dependendo da dimensão e da dificuldade do problema a ser determinado. Valores na faixa de 20 a 50 são mais comuns de serem escolhidos (EBERHART; SHI, 2001) (POLI; KENNEDY; BLACKWELL, 2007).

A topologia de um enxame define a maneira de como as partículas trocam informações e se comunicam. As principais topologias utilizadas e conhecidas como topologias estáticas são: a topologia global e a topologia local. Na topologia global ou topologia *gBest*, todas as partículas são conectadas entre si representando um grafo totalmente conectado. A trajetória das partículas é influenciada pela melhor solução encontrada (*gBest*) por alguma outra partícula do enxame. Esta topologia garante uma rápida convergência na solução de problemas, porém ela pode se convergir em mínimos locais. Na topologia local ou topologia *lBest*, o enxame está organizado em anel, onde cada partícula interage com seus dois vizinhos próximos. Dessa forma, a trajetória das partículas é influenciada pelas partículas adjacentes. Apesar da convergência nesta topologia ser mais lenta, ela é capaz de escapar de mínimos locais e explorar novas áreas em busca de soluções (KENNEDY; MENDES, 2002). As figuras a seguir ilustram as topologias citadas.

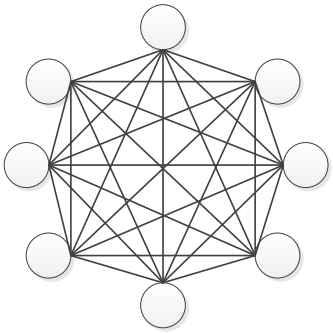

Figura 6: Topologia global.

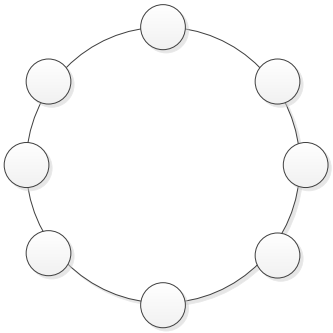

Figura 7: Topologia local.

3.3.3 Algoritmo básico de PSO

O algoritmo de otimização por enxame de partículas básico pode ser descrito da seguinte forma:

1. Inicializar as posições e velocidades das partículas com valores aleatórios;
2. Inicializar os valores de *pBest*, *gBest* e os outros parâmetros do algoritmo;
3. Para cada passo t do algoritmo fazer:
   (a) Avaliar a função de *fitness* para cada partícula;
   (b) Comparar o valor atual de *fitness* da partícula com seu *pBest*. Se o valor atual é melhor, atualizar *pBest*;
   (c) Atualizar o valor de *gBest* com o melhor valor encontrado pelo enxame;

(d) Atualizar as posições e velocidades das partículas;

4. Como critério de parada, se uma solução ótima for encontrada ou o número de passos $tMax$ for alcançado, o algoritmo é interrompido. Senão, voltar ao passo 3.

A Figura 8 ilustra em um fluxograma o algoritmo descrito.

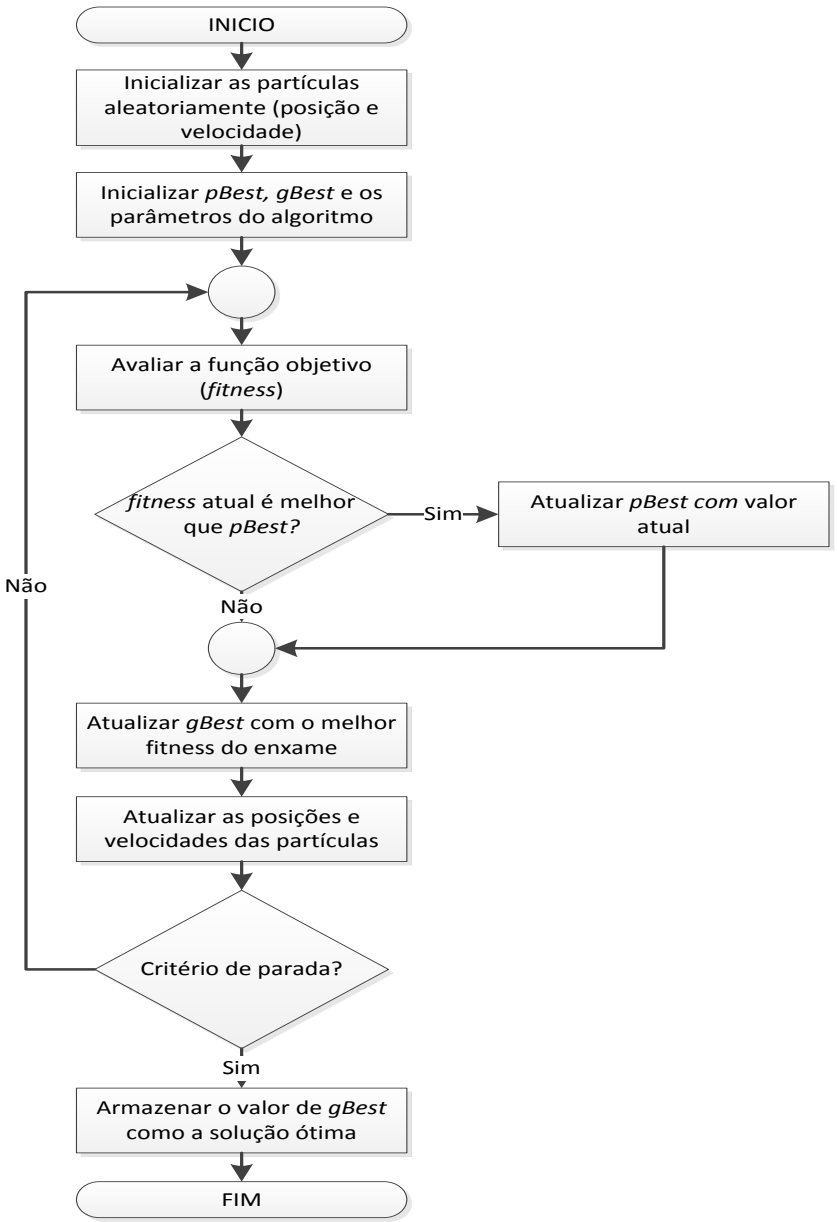

Figura 8: Fluxograma de um algoritmo básico de PSO.

3.3.4 Fator de Inércia

Para um melhor controle da velocidade da partícula no espaço de busca e com a finalidade de reduzir o efeito do limite de velocidade $vMax$ da partícula, um fator de inércia foi proposto por Shi e Eberhart (1998) de forma a influenciar na capacidade de busca local e global do algoritmo de PSO. A equação (3.12) mostra a inclusão do fator de inercia ω na velocidade da partícula.

$$v_i(t+1) = \omega v_i(t) + a_1 r_1(pBest_i - p_i(t)) + a_2 r_2(gBest - p_i(t)) \tag{3.12}$$

Para um valor de ω pequeno ($\omega < 0,8$), o algoritmo de PSO se comporta como um algoritmo de busca local, e se existe no espaço de busca inicial uma solução aceitável o algoritmo encontra um ótimo global rapidamente. Se o valor de ω é grande ($\omega > 1,2$), o PSO se comporta como um método de busca global sendo capaz de explorar uma área de busca ampla, porém isso faz com que o algoritmo leve mais passos para encontrar uma solução e também pode não encontra-la. Quando o valor de ω é médio ($0,8 < \omega < 1,2$), o algoritmo terá uma melhor chance de encontrar uma solução ótima, porém poderá gastar um número maior de passos. Alguns algoritmos de PSO iniciam com um valor de ω maior, e vão decrementando este valor linearmente até um valor menor, de acordo com o número de passos do algoritmo. Este procedimento faz com que o PSO tenha uma boa capacidade de busca global no início da execução do algoritmo, e uma exploração mais local no final da execução. A equação (3.13) mostra uma forma de atualização do fator de inércia.

$$w(t+1) = \omega_{max} - \frac{(\omega_{max} - \omega_{min})}{t_{max}}(t) \tag{3.13}$$

Onde:

- ω_{max} e ω_{min} correspondem aos valores máximos e mínimos do componente inercial ω;
- $\omega(t+1)$ é o novo valor de ω no passo t+1;
- t_{max} é o número máximo de passos alcançados pelo algoritmo;
- t é passo atual do algoritmo.

3.3.5 Fator de Constrição

O fator ou coeficiente de constrição foi introduzido por Clerc (1999) com a finalidade de garantir a convergência do algoritmo de otimização por enxame de partículas e eliminar a necessidade de limitar a velocidade das mesmas. A equação (3.14) mostra um modo de estimar este fator.

$$K = \frac{2}{\left|2 - \varphi - \sqrt{(\varphi^2 - 4\varphi)}\right|} \tag{3.14}$$

Onde φ é uma constante definida a partir dos valores dos parâmetros a_1 e a_2, sendo $\varphi = a_1 + a_2$ e $\varphi > 4$ (por exemplo, $\varphi = 4,1$). A equação da velocidade utilizando o método de constrição é definida pela equação (3.15).

$$v_i(t+1) = K[\omega v_i(t) + a_1 r_1(pBest_i - p_i(t)) + a_2 r_2(gBest - p_i(t))] \tag{3.15}$$

Nota-se que o algoritmo PSO com o coeficiente de constrição é equivalente algebricamente a utilizar o algoritmo de enxame com o fator de inércia (POLI; KENNEDY; BLACKWELL, 2007). As partículas do enxame utilizando o fator de constrição deveriam convergir para uma solução sem a necessidade de limitar a velocidade para *vMax*. Porém, através de alguns experimentos realizados foi concluído que a melhor forma para a utilização deste método de constrição seria limitar o valor de *vMax* em relação ao valor de *pMax* (EBERHART; SHI, 2000).

3.3.6 Exemplo de otimização com o algoritmo PSO

Para ilustrar o funcionamento de um algoritmo de PSO básico, utilizou-se neste exemplo um procedimento para resolver um problema de otimização de uma função não linear de duas dimensões. O objetivo consiste em determinar para quais valores de x_1 e x_2 a função dada por (3.16) é minimizada.

$$y = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{3.16}$$

Os parâmetros considerados para o algoritmo foram: o tamanho do enxame igual a 30, o valor dos coeficientes de aceleração (a_1 e a_2) igual a 2, o fator de inércia (ω) igual a 0,9. A velocidade das partículas [*vMin* e *vMax*] foi limitada na faixa [-1 e 1] e o espaço de

busca [$pMin$ e $pMax$] limitado em [-2 e 2]. A topologia do enxame escolhida foi a global. Como critério de parada foi utilizado um número de passos de execução igual a 100. Após a execução do algoritmo, o valor de $gBest$ que corresponde ao valor minimizado de y da função correspondente foi igual a 0 e os valores das variáveis associadas foram $x_1 = 1$ e $x_2 = 1$. A Figura 9 ilustra o valor de $gBest$ encontrado pelo algoritmo em cada passo de execução.

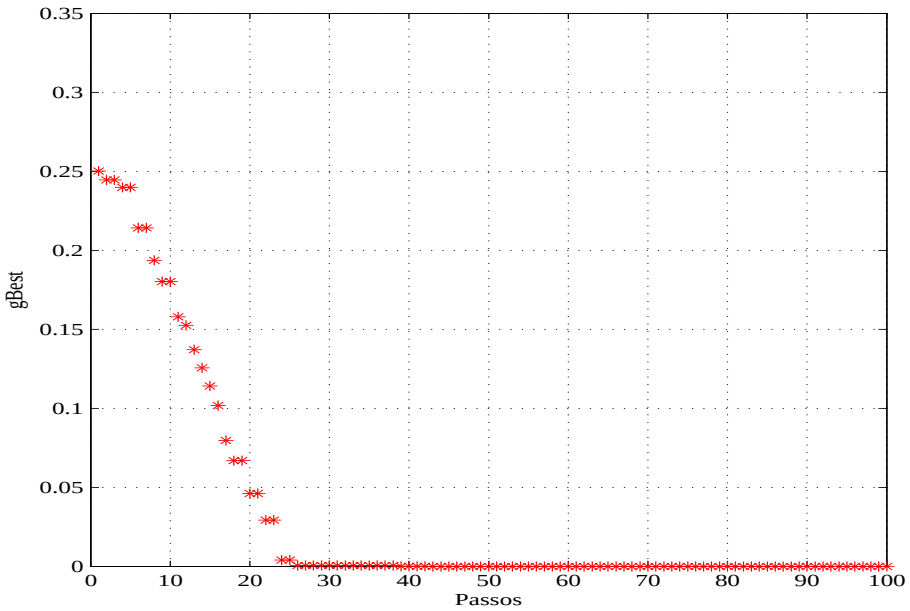

Figura 9: Passos de execução do algoritmo PSO.

Desta forma conclui-se que com um algoritmo de PSO básico é possível resolver problemas de otimização de funções não lineares.

3.4 Algoritmo de Agrupamento Fuzzy C-Means

Os algoritmos de agrupamento são considerados procedimentos não supervisionados que visam à partição de um determinado conjunto de dados em subgrupos baseando-se na relação de similaridade que existe entre os dados. O objetivo de um agrupamento é dividir o conjunto de dados de tal maneira que elementos pertencentes a um mesmo grupo ou cluster possuam grande similaridade entre si e grande dissimilaridade em relação aos outros grupos. Esta similaridade pode ser medida através da distância, normalmente euclidiana, entre dois vetores de dados (OLIVEIRA; PEDRYCZ et al., 2007).

Alguns algoritmos de agrupamento encontrados na literatura são baseados na teoria de conjuntos convencionais, conhecidos como *crisp*, ou nos conjuntos nebulosos, conhecidos como *fuzzy*. Os métodos de agrupamento que se baseiam nos conjuntos *crisp* exigem que

cada ponto de dado de um conjunto deve pertencer a exatamente um grupo. Entre os algoritmos de agrupamento convencionais mais conhecidos pode-se citar *K-Means* e *C-Means*. Os métodos de agrupamento *fuzzy* que se baseiam nos conjuntos *fuzzy* (ZADEH, 1965), permitem que cada dado de um determinado conjunto possa pertencer a mais de um grupo ou cluster de acordo com um grau de pertinência que pode variar entre 0 e 1. O algoritmo *Fuzzy C-Means*(FCM) é um exemplo de algoritmo de agrupamento *fuzzy* (JANG; SUN; MIZUTANI, 1997).

O algoritmo de agrupamento de dados *Fuzzy C-Means* introduzido por Dunn (1973) e melhorado por Bezdek (1981) é um método não supervisionado de agrupamento de dados e corresponde a uma generalização dos algoritmos que utilizam partições *crisp*. O objetivo do algoritmo é otimizar os centros dos agrupamentos através de um procedimento iterativo, minimizando uma função objetivo que é relacionada com a distancia entre os dados de um conjunto e os centros dos agrupamentos aos quais os dados pertencem com determinado grau de pertinência. Esta minimização pode produzir clusters mais adequados do que aqueles produzidos pelos métodos de agrupamento convencionais (ROCHA et al., 2012).

O FCM tem sido utilizado em diversas aplicações como em sistemas de classificação (NITANDA; HASEYAMA; KITAJIMA, 2004) (LIN; TSAI; CHEN, 2011), sistemas de previsão (MENG et al., 2007) (WEI; YANG, 2009) e outras. Também é utilizado em aplicações de redes RBF para estimar as posições iniciais dos valores dos centros das funções de base radial (JANG; SUN; MIZUTANI, 1997). Para ilustrar o processamento de informações de um algoritmo FCM, considere o seguinte conjunto de dados X representado por (3.17) com n elementos, onde cada elemento possui a dimensão p.

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ x_{31} & x_{32} & \dots & x_{3p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} \tag{3.17}$$

O algoritmo particiona o conjunto de dados X em c partições *fuzzy* e a cada iteração busca por um centro ótimo para cada partição. Os centros destes agrupamentos podem ser representados por uma matriz de protótipos ou centros M de dimensão c x p, como representada por (3.18).

$$M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1p} \\ m_{21} & m_{22} & \dots & m_{2p} \\ m_{31} & m_{32} & \dots & m_{3p} \\ \vdots & \vdots & \vdots & \vdots \\ m_{c1} & m_{c2} & \dots & m_{cp} \end{bmatrix} \tag{3.18}$$

Cada elemento do conjunto de dados pode pertencer a um ou mais clusters com um determinado grau de pertinência associado, sendo representado por uma matriz de partição U (3.19) cuja dimensão é c x n, onde $2 \leq c < n$.

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ u_{21} & u_{22} & \dots & u_{2n} \\ u_{31} & u_{32} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ u_{c1} & u_{c2} & \dots & u_{cn} \end{bmatrix} \tag{3.19}$$

Algumas condições devem ser consideradas ao particionar o conjunto de dados, onde u_{ij} representa o grau de pertinência do dado j no grupo i. As equações a seguir modelam este contexto.

$$u_{ij} \in [0, 1] \qquad 1 \leq i \leq c; \quad 1 \leq j \leq n; \tag{3.20}$$

$$\sum_{i=1}^{c} u_{ij} = 1 \qquad 1 \leq j \leq n; \tag{3.21}$$

$$0 < \sum_{j=1}^{n} u_{ij} < n \qquad 1 \leq \ i \leq c; \tag{3.22}$$

A condição (3.20) define que o grau de pertinência do dado i para o cluster j pertença a faixa de valores de 0 a 1, a condição (3.21) garante que a soma dos graus de pertinência para cada dado seja igual a 1, e a condição (3.22) garante que nenhum cluster é vazio. O objetivo do algoritmo é minimizar a distância que existe entre os centros de cada agrupamento e os dados do conjunto daquele agrupamento através da função objetivo J_m representada pela equação (3.23).

$$J_m(U, M) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m \left\| x_j - c_i \right\|^2 \qquad 1 \leq m < \infty; \tag{3.23}$$

Onde:

- m é conhecido como parâmetro de fuzzificação;
- $\|x_j - c_i\|$ é a distância euclidiana entre o j-ésimo dado do conjunto X e o i-ésimo centro do agrupamento.

O parâmetro de fuzzificação m influencia no grau de pertinência de determinado dado de um grupo, devendo ser escolhido adequamente. Os valores recomendados estão na faixa de $1,5 \leq m < 3$. O valor $m = 2$ é frequentemente escolhido (BEZDEK; EHRLICH; FULL, 1984). O algoritmo FCM realiza o particionamento do conjunto de dados atualizando os centros c_i de cada agrupamento, representado pela equação (3.24), e atualizando os graus de pertinência u_{ij} da matriz U, de acordo coma equação (3.25), de forma a minimizar o valor da função objetivo.

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m} \tag{3.24}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|x_j - c_i\|}{\|x_j - c_k\|} \right)^{\frac{2}{m-1}}} \tag{3.25}$$

A atualização pode ser interrompida se o número de iterações ultrapassar um número pré-definido, ou quando as mudanças nos protótipos dos agrupamentos forem menores do que um determinado valor de precisão estipulado.

Os passos básicos de um algoritmo *Fuzzy C-Means* estão descritos a seguir:

1. Determinar o número de agrupamentos c, o valor do expoente m, e os critérios de parada como o valor de precisão δ e o número de iterações máximo;
2. Inicializar o contador de iterações t com 0;
3. Inicializar a matriz de partição *fuzzy* U com valores aleatórios entre 0 e 1 de acordo com o critério estabelecido pela equação (3.20);
4. Calcular os centros c_i dos agrupamentos de acordo com a equação (3.24);

5. Computar a função custo J_m de acordo com a equação (3.23);

6. Atualizar a matriz de partição U com os valores de pertinência u_{ij} de acordo com a equação (3.25);

7. Incrementar o contador t;

8. Repetir até $\left|U^{(t+1)} - U^{(t)}\right| < \delta$ ou o número de iterações t for maior que o máximo permitido. Senão voltar para passo 4.

3.4.1 Exemplo de agrupamento com o algoritmo Fuzzy C-Means

Para ilustrar o funcionamento do algoritmo de agrupamento *Fuzzy C-Means*, utilizou-se neste exemplo uma função não linear com duas variáveis de entrada, dada por (3.26). Para representar a função, 100 dados foram gerados aleatoriamente. A Figura 10 ilustra os dados gerados.

$$y = x_1 x_2 + x_2^2 \qquad x_1, x_2 \in [-1, +1] \tag{3.26}$$

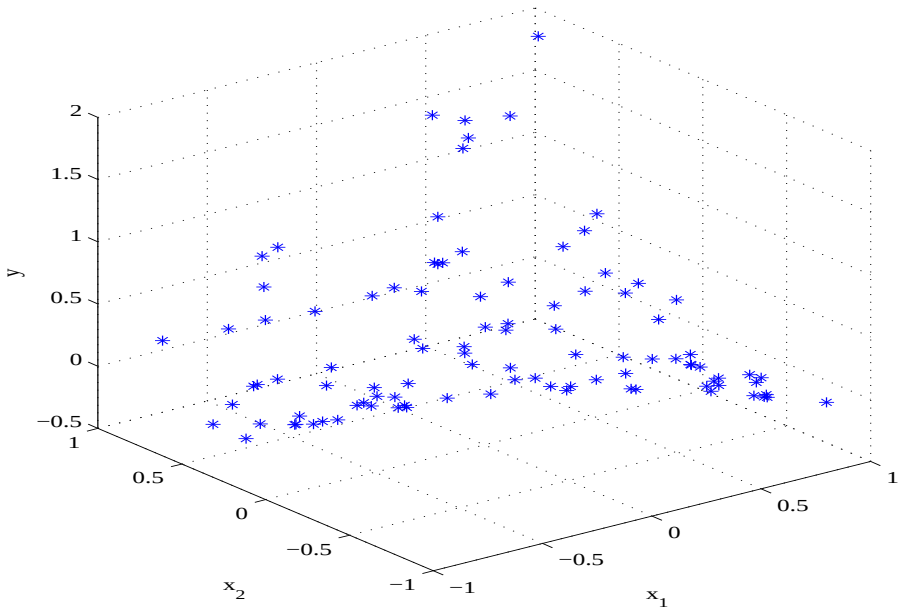

Figura 10: Dados da função não linear.

Os parâmetros considerados para o algoritmo de agrupamento foram: o valor do parâmetro de fuzificação m igual a 2, o número máximo de iterações igual a 100, o valor de precisão igual a 10^{-5} e o número de agrupamentos c igual a 6.

A atualização do algoritmo foi interrompida quando as mudanças nos protótipos dos agrupamentos foram menores do que o valor de precisão 10^{-5} estipulado. O algoritmo convergiu com 59 iterações e o valor da função objetivo J_m encontrada foi igual a 9,68. A Figura 11 ilustra os seis agrupamentos resultantes representados por cores diversas e os respectivos centros dos clusters indicados por círculos.

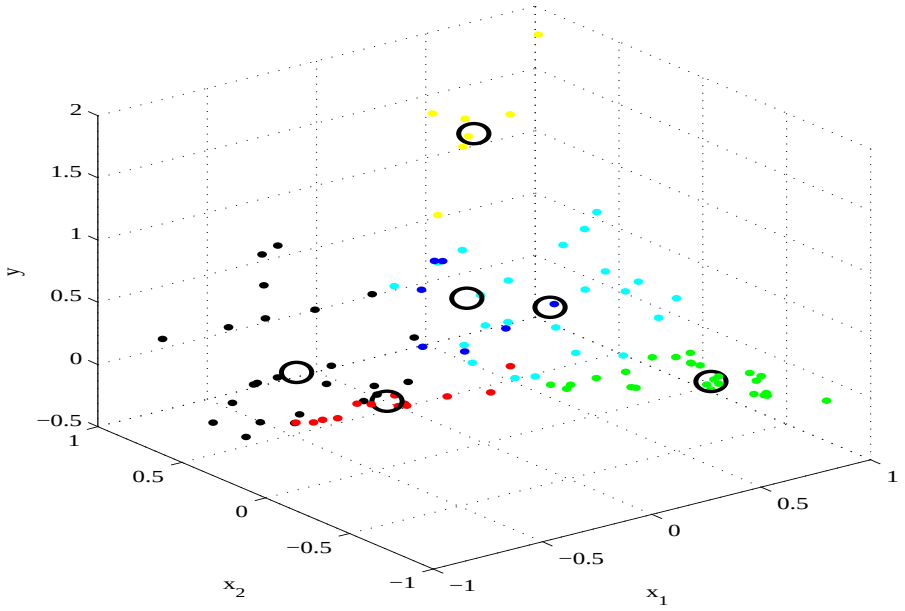

Figura 11: Agrupamentos resultantes.

Os valores dos respectivos centros dos agrupamentos são ilustrados na Tabela 2.

Tabela 2: Valores dos centros dos agrupamentos.

Centro	**Valores**
c_{11}	-0,585831
c_{12}	-0,183268
c_{21}	-0,760319
c_{22}	-0,886766
c_{31}	0,615630
c_{32}	-0,549407
c_{41}	-0,512356
c_{42}	0,451900
c_{51}	0,596887
c_{52}	0,840596
c_{61}	0,488788
c_{62}	0,244412

4 Metodologia

Neste capítulo serão abordados os métodos utilizados para o ajuste dos parâmetros livres de uma rede RBF. A partir de um procedimento prévio de ajuste de parâmetros, será proposta a utilização do algoritmo de otimização por exame de partículas para realizar um ajuste fino nas larguras das funções de entrada da rede RBF a fim de obter uma melhoria na exatidão dos modelos resultantes.

4.1 Procedimento de seleção de parâmetros de redes RBF

Como discutido nos capítulos anteriores, os métodos de treinamento de uma rede RBF usualmente combinam estratégias não supervisionadas e supervisionadas para o ajuste dos parâmetros da rede (dados relacionados com os centros, larguras e pesos). Este trabalho aborda a combinação de um método não supervisionado para pré-selecionar os parâmetros de redes RBF com a utilização de um algoritmo PSO para realizar o ajuste fino dos valores das larguras das funções de entrada da rede em questão. A função que determina a precisão ou exatidão de um determinado modelo pode ser dada pelo valor da somatória dos erros quadráticos (SEQ) sobre todos os padrões de treinamento, expressa por (4.1), onde L é o número de amostras de treinamento, d_n corresponde à saída desejada da n-ésima amostra e Y_n é o valor da saída atual correspondente a n-ésima amostra.

$$SEQ = \frac{1}{2}\sum_{i=1}^{L}(d_n - Y_n)^2 \tag{4.1}$$

O método não supervisionado utilizado é composto pela associação de diferentes abordagens descritas em Santos (2014). O resumo das etapas utilizadas para a seleção prévia dos parâmetros de uma rede RBF é mostrado a seguir:

1. Escolher o número de nós da camada escondida da rede RBF em questão, ou seja, a quantidade de funções de base radial empregada na rede neural. Determinar um

valor de precisão (SEQ) como critério de parada da rede.

2. Os dados de treinamento da rede são processados pelo algoritmo de agrupamento *Fuzzy C-Means* para determinar as faixas de dados e os valores dos centros das funções de base radial. As informações dos clusters serão utilizadas para selecionar os parâmetros das funções de ativação, e o número de agrupamentos definirá o número de nós da canada escondida da rede.

3. Os valores dos centros (c_j) das funções gaussianas da rede serão os valores centrais dos clusters determinados pelo algoritmo *Fuzzy C-Means*.

4. Os valores das larguras das funções são inicialmente estimadas a partir da equação (4.2) pelo cálculo do desvio padrão (σ_j^*) a partir das faixas dos dados (x_j) de cada variável relacionada com cluster identificado pelo algoritmo de agrupamento.

$$\sigma_j^* = \sqrt{\frac{\sum_{j=1}^{m} x_j^2 - \frac{1}{m}(\sum_{j=1}^{m} x_j)^2}{m-1}} \tag{4.2}$$

5. Realizar um escalonamento nos dados dos desvios padrões a partir da equação (4.3) objetivando obter valores das larguras das funções que possibilitem uma boa capacidade de interpolação da rede em questão.

$$\sigma_j^{'} = \frac{1}{2}\frac{|max(x_j) - min(x_j)|}{\sigma_j^*} \tag{4.3}$$

6. Promover um ajuste adicional nos valores das larguras ($\sigma_j^{'}$) por intermédio de um fator multiplicativo comum f_m, possibilitando uma melhor exatidão no modelo resultante. A faixa de valores entre $[1, 10]$, comum a todos as larguras, servirá como ajuste adicional dos parâmetros em questão.

$$\sigma_j = f_m * \sigma_j{'} \tag{4.4}$$

7. Calcular os pesos w_j da camada de saída da rede através do método dos mínimos quadrados dado pela equação (3.9). Calcular a informação de saída da rede através da equação (3.7).

8. Validar o modelo da rede RBF resultante e verificar se a exatidão obtida está dentro da especificação desejada. Em caso afirmativo o procedimento está finalizado. Armazenar as informações dos parâmetros determinados.

9. Senão, voltar ao passo (1) do procedimento e alterar o número de nós da rede ou variar o valor do fator multiplicativo do passo (6), repetindo os passos consecutivos.

A Figura 12 ilustra em um fluxograma o procedimento de seleção prévia de parâmetros de uma rede RBF.

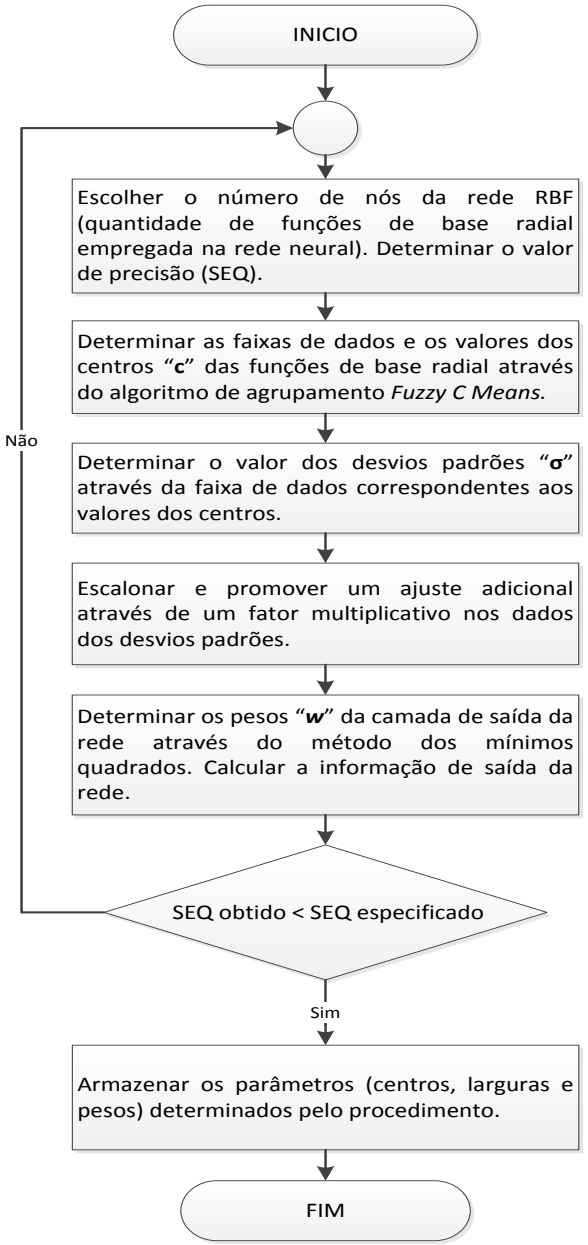

Figura 12: Fluxograma do procedimento de seleção prévia de parâmetros.

4.2 Procedimento proposto para o ajuste de parâmetros de redes RBF via PSO

O procedimento proposto nesta seção tem como objetivo refinar os valores das larguras das funções de base radial de uma rede, cujos valores foram previamente ajustados pela abordagem descrita na seção anterior. As partículas do PSO serão inicializadas conforme a representação dada por (4.5), com os valores dos desvios padrões pré-ajustados conforme a abordagem citada.

$$p_i = [\sigma_{1i}, \sigma_{2i}, \sigma_{3i}, \ldots, \sigma_{ni}] \tag{4.5}$$

A velocidade de cada partícula é inicializada aleatoriamente entre 0 e $vMax$ (4.6), onde $rand()$ é um número aleatório gerado no intervalo de 0 a 1.

$$v_i = vMax * rand() \tag{4.6}$$

O processo de refinamento dos parâmetros de redes RBF via algoritmo de PSO é descrito a seguir:

1. Definir os parâmetros do algoritmo: os limites de velocidade $vMax$ e $vMin$; o espaço de busca $pMax$ e $pMin$; os fatores de aceleração a_1 e a_2; o fator de inércia $wMax$ e $wMin$; o número máximo de passos $tMax$; e o tamanho do enxame. Inicializar os valores de $pBest$ e de $gBest$;

2. Inicializar as posições das partículas (4.5) do algoritmo com os valores correspondentes às larguras pré-ajustadas via procedimento descrito na seção anterior. Inicializar também as informações das velocidades (4.6);

3. Avaliar o *fitness* de cada partícula através da equação (4.1), onde os valores dos pesos da camada de saída da rede são ajustados pelo método dos mínimos quadrados;

4. Se o *fitness* atual da partícula i for menor que o valor de $pBest_i$, atualizar o algoritmo com o valor do $pBest$ atual. Se o *fitness* atual da partícula i for menor que o valor de $gBest$, atualizar o algoritmo com o valor atual;

5. Atualizar as velocidades (Equação 3.12) e posições (Equação 3.11) das partículas;

6. Se uma solução otimizada for encontrada ou o número de passos $tMax$ for alcançado,

o algoritmo é interrompido e os novos valores de larguras da RBF serão atribuídos por meio dos dados associados ao melhor $gBest$ obtido;

7. Senão, voltar ao passo (3) do procedimento;

8. Na finalização do algoritmo, a exatidão obtida com rede RBF ajustada via PSO, é validada através de um conjunto de dados de teste, calculando-se o valor da SEQ correspondente. Se o resultado obtido for melhor do que aquele resultante via parâmetros pré-ajustados, o modelo da RBF está refinado. Senão, se define outros parâmetros do PSO e realiza-se o procedimento outra vez;

A Figura 13 ilustra em um fluxograma o procedimento de refinamento de parâmetros de uma rede RBF. No próximo capítulo deste trabalho encontram-se alguns exemplos de aplicações que servirão para ilustrar a metodologia proposta neste capítulo.

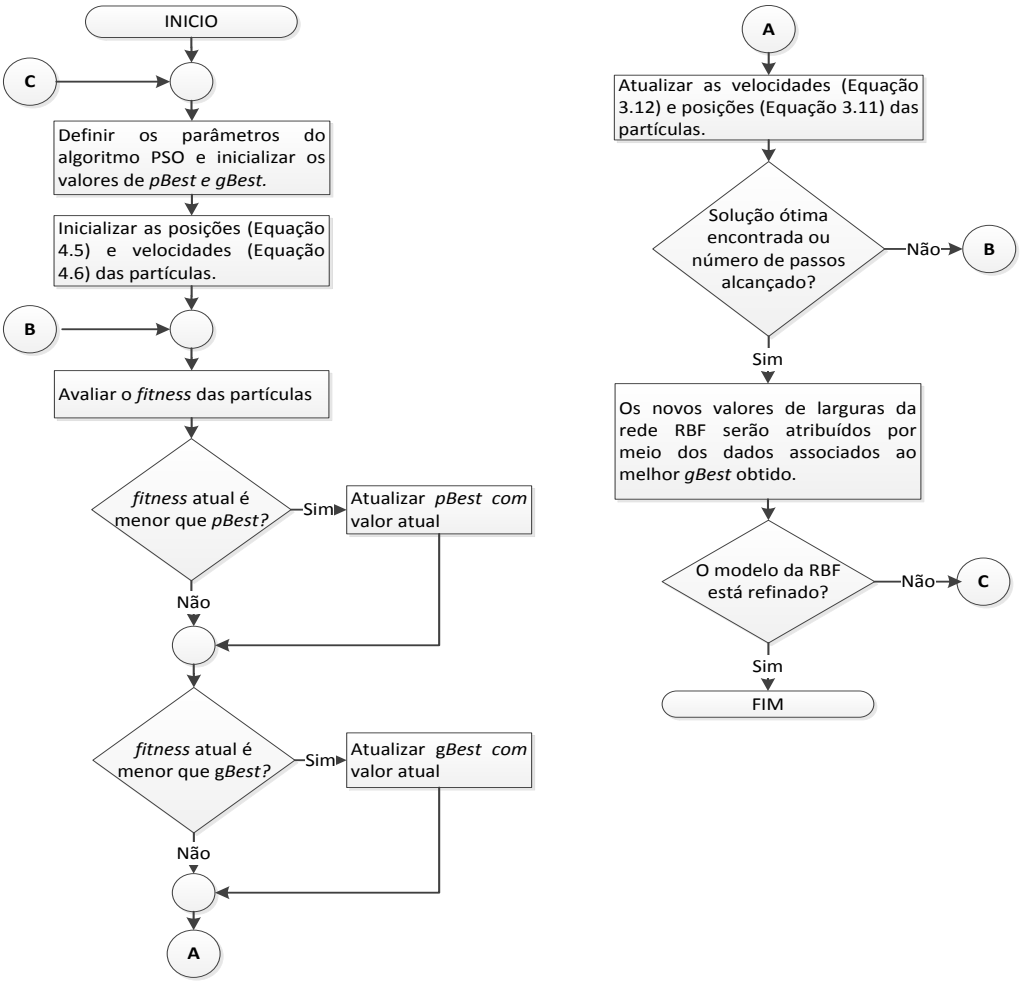

Figura 13: Fluxograma do procedimento de refinamento de parâmetros.

5 Resultados

Neste capítulo serão apresentados alguns exemplos para ilustrar a aplicação da metodologia proposta neste trabalho. Exemplos de aproximações de funções, previsão de série temporal e classificação de padrões serão utilizados com o objetivo de mostrar o potencial da abordagem proposta. Os resultados obtidos serão comparados com técnicas utilizadas em outros trabalhos.

5.1 Experimentos

Para ilustrar o procedimento proposto neste trabalho, dois exemplos de aplicações em aproximações de funções não lineares, uma aplicação em previsão de série temporal e um exemplo de classificação serão utilizados. Para os três primeiros exemplos citados, a estrutura da rede RBF utilizada nos experimentos é constituída de duas entradas, seis funções de base radial, e uma unidade na camada de saída. O valor do fator multiplicativo (equação (4.4)) utilizado no procedimento prévio para o ajuste adicional nos valores das larguras das funções de base radial foi igual a seis para todos os exemplos. Para o algoritmo de PSO utilizado, os parâmetros associados que apresentaram os melhores resultados nos experimentos estão ilustrados na Tabela 3. Alguns destes valores foram ajustados de acordo com a característica do experimento abordado.

Tabela 3: Configuração dos parâmetros utilizados pelo algoritmo de PSO.

Parâmetros	**Valores**
Número de partículas	50
Fator de Inércia [$wMin$; $wMax$]	[0,4; 0,9]
Constante de aceleração cognitiva (a_1)	2
Constante de aceleração social (a_2)	2
Velocidade [$vMin$; $vMax$]	[-10; 10]
Posição [$pMin$; $pMax$]	$[10^{-6}; 1000]$

O algoritmo de otimização foi executado dez vezes para cada experimento para testar o desempenho do procedimento, e os valores resultantes dos passos correspondentes em

cada experimento foram obtidos através da média dos valores associados a cada execução do algoritmo do PSO. Os experimentos foram executados em uma máquina Intel Core i3, com 4GB de memória RAM e sistema operacional Windows 7 64 bits. Os procedimentos foram desenvolvidos no software Matlab R2013a. O código implementado referente ao Exemplo 1 dos experimentos se encontra no Apêndice A da dissertação.

5.1.1 Exemplo 1 - Aproximação de Função

Este exemplo utiliza uma função não linear (5.1) com duas variáveis de entrada que representa uma função estática com duas não linearidades. Foram utilizados 200 dados gerados aleatoriamente dentro de um intervalo pré-definido para representar a função, sendo que 100 dados foram utilizados para treinamento e 100 dados utilizados para validação do modelo.

$$y = x_1 x_2 + x_2^2 \qquad x_1, x_2 \in [-1, +1] \tag{5.1}$$

A Figura 14 ilustra os dados originais da função e os aproximados por uma rede RBF, cujos parâmetros da mesma foram ajustados através da abordagem proposta por Santos (2014).

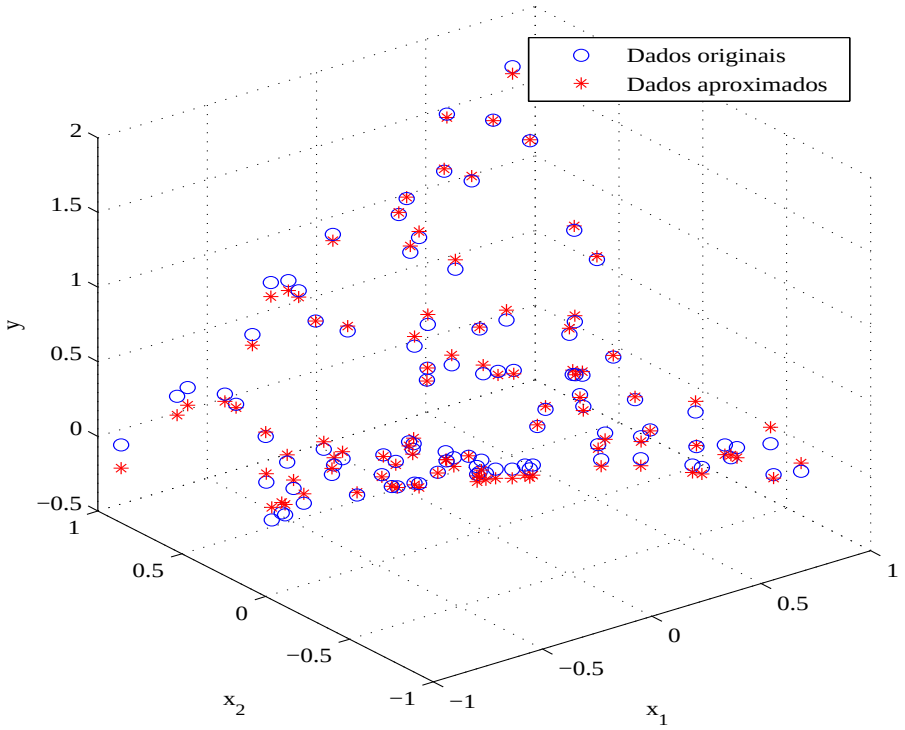

Figura 14: Dados originais e os aproximados pela rede RBF utilizando o ajuste prévio de parâmetros referente ao Exemplo 1.

O tempo de processamento do procedimento de pré-ajuste de parâmetros foi de 0,16 segundos. Utilizou-se o procedimento proposto nesta dissertação para o refinamento dos valores das larguras da rede RBF, onde se empregou como critério de parada do algoritmo PSO um valor de SEQ menor que 10^{-6} ou um número máximo de passos igual a 2000. A rede RBF resultante apresentou após dez execuções do algoritmo um valor médio de SEQ de $6,24 * 10^{-7}$, convergindo com uma média de 44 passos de execução do PSO. O menor valor encontrado foi de $6,26 * 10^{-8}$. O tempo de processamento médio foi de 0,45 segundos. A Tabela 4 ilustra os valores dos parâmetros pré-ajustados para rede RBF e os valores refinados via PSO, a partir do conjunto de dados de treinamento. Os valores dos centros das funções de base radial para os dois modelos de redes RBF foram: $c = [c_{11}, c_{12}, c_{21}, c_{22}, \ldots, c_{61}, c_{62}] = [-0,657425, 0,218492, 0,766503, 0,835955, 0,638923, 0,008197, -0,228962, -0,705242, -0,160886, 0,556787, 0,697378, -0,675270]$. A Figura 15 mostra os resultados da aproximação em questão utilizando o procedimento proposto neste trabalho.

Tabela 4: Valores dos parâmetros pré-ajustados e refinados da rede RBF referentes ao Exemplo 1.

Parâmetros	**Valores pré-ajustados**	**Valores refinados**
σ_{11}	10,701066	98,360791
σ_{12}	9,105471	94,308600
σ_{21}	7,104280	91,147963
σ_{22}	5,736190	92,410385
σ_{31}	9,175625	98,845443
σ_{32}	9,589272	90,796023
σ_{41}	9,838328	92,418208
σ_{42}	10,577233	92,491817
σ_{51}	9,822681	91,130667
σ_{52}	9,854824	91,933546
σ_{61}	9,120310	92,939390
σ_{62}	8,534768	92,254109
w_0	-0,085724*10^4	0,006863*10^7
w_1	1,026315*10^4	-1,085196*10^7
w_2	0,016475*10^4	4,322262*10^7
w_3	1,661979*10^4	1,300870*10^7
w_4	0,393652*10^4	6,504516*10^7
w_5	-1,911057*10^4	-5,083008*10^7
w_6	-1,104105*10^4	-5,966198*10^7

A Tabela 5 contêm os valores dos erros de aproximação com o conjunto de treinamento e de teste para as duas condições consideradas de parâmetros. Nota-se que a rede RBF com parâmetros refinados via PSO foi 10^8 vezes mais precisa em relação à rede com parâmetros apenas pré-ajustados.

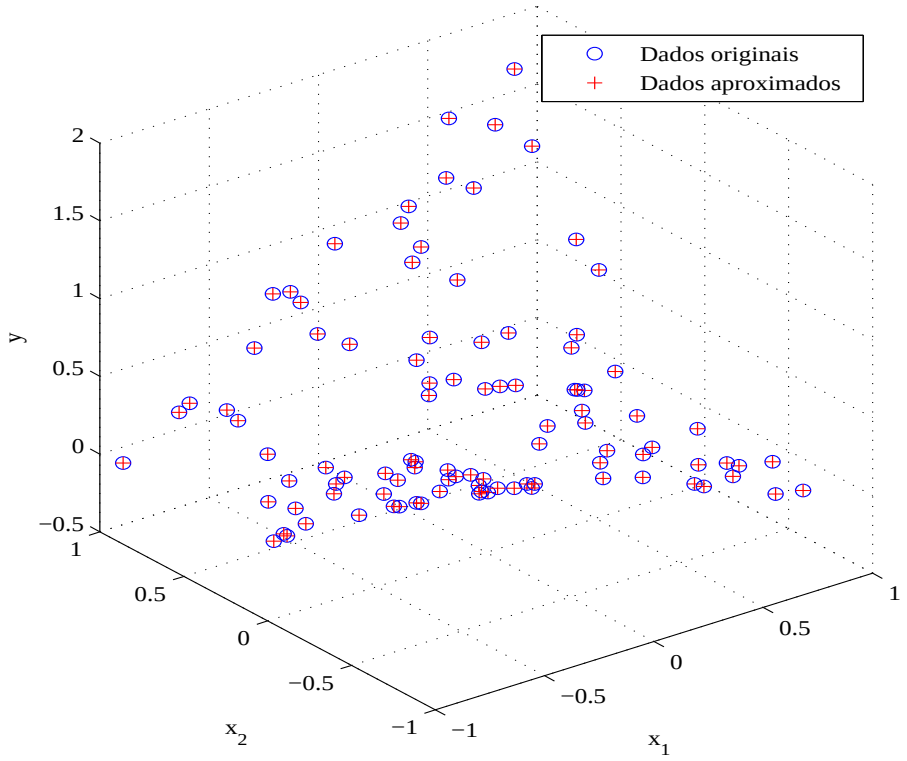

Figura 15: Dados originais e os aproximados pela rede RBF após o procedimento proposto referente ao Exemplo 1.

Tabela 5: Erro de aproximação da rede RBF para o Exemplo 1.

	Conjunto de Treinamento	**Conjunto de Teste**
CONTEXTO	**SEQ**	**SEQ**
Pré-Ajuste	0,1457	0,1240
Refinamento	$6{,}2629*10^{-8}$	$8{,}2349*10^{-8}$

5.1.2 Exemplo 2 - Aproximação da Função de Rosenbrock

Este exemplo utiliza a função não linear de Rosenbrock (ROSENBROCK, 1960)(5.2) que apresenta características não lineares acentuadas. Foram utilizados 200 dados gerados aleatoriamente dentro de um intervalo pré-definido para representar a função, sendo que 100 dados foram utilizados para treinamento e 100 dados utilizados para validação do modelo.

$$y = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \qquad x_1, x_2 \in [-1, +1] \tag{5.2}$$

A Figura 16 ilustra os dados originais da função e os aproximados pela rede através do procedimento de pré-ajuste de parâmetros. O procedimento resultou em um tempo de processamento de 0,17 segundos.

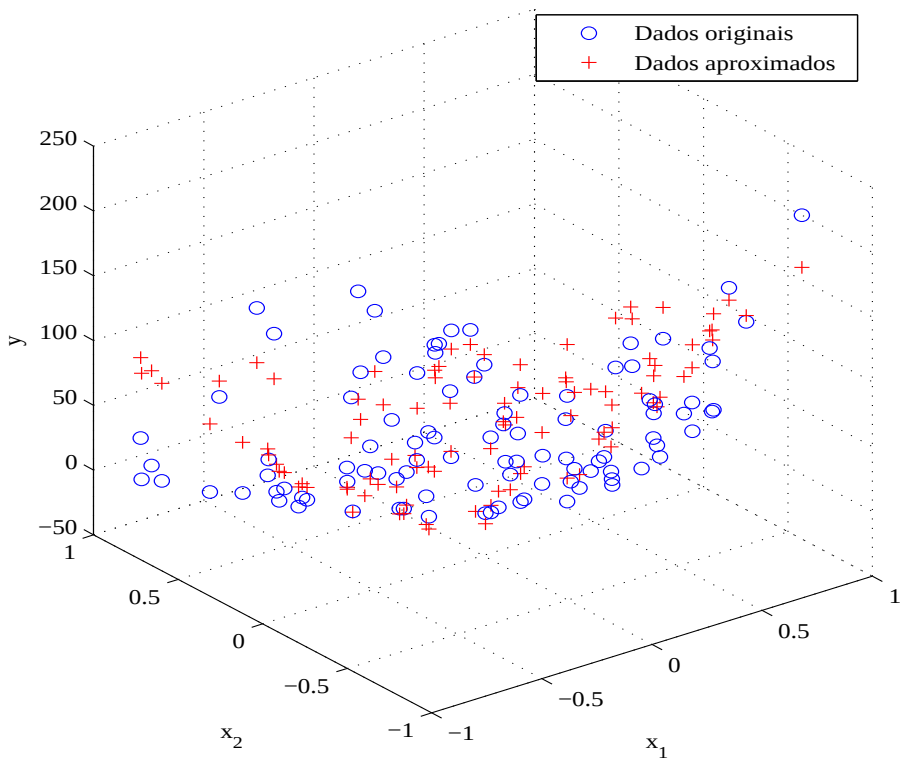

Figura 16: Dados originais e os aproximados pela rede RBF utilizando o ajuste prévio de parâmetros referentes ao Exemplo 2.

O procedimento proposto para refinamento de parâmetros de redes RBF foi aplicado, onde se utilizou como critério de parada do algoritmo um valor de SEQ menor que 10^{-1} ou um número máximo de passos igual a 2000. A rede RBF resultante apresentou após dez execuções do algoritmo um valor médio de SEQ de $8,96 * 10^{-2}$, convergindo com uma média de 512 passos de execução do PSO. O menor valor encontrado foi de $5,50 * 10^{-2}$. O tempo médio de processamento foi de 5,4 segundos. A Tabela 6 ilustra os valores dos parâmetros pré-ajustados para rede RBF e os valores refinados via PSO, a partir do conjunto de dados de treinamento. Os valores dos centros das funções de base radial para os dois modelos de redes RBF foram: $c = [c_{11}, c_{12}, c_{21}, c_{22}, \ldots, c_{61}, c_{62}] = [-0,404344, -0,796274, -0,154321, -0,030756, -0,311015, -0,692154, -0,207107, 0,126532, 0,004165, 0,281403, 0,502756, -0,449258]$.

A Figura 17 mostra os resultados da aproximação em questão utilizando o procedimento proposto neste trabalho.

A Tabela 7 contêm os valores dos erros de aproximação com o conjunto de treinamento e de teste para as duas condições consideradas de parâmetros. A precisão da rede RBF cujos parâmetros foram refinados via PSO foi 10^6 vezes melhor em relação à rede com parâmetros apenas pré-ajustados.

Tabela 6: Valores dos parâmetros pré-ajustados e refinados da rede RBF referente ao Exemplo 2.

Parâmetros	**Valores pré-ajustados**	**Valores refinados**
σ_{11}	6,298317	28,153401
σ_{12}	6,851702	18,009637
σ_{21}	12,896616	29,496220
σ_{22}	7,748794	16,692102
σ_{31}	7,124916	23,843233
σ_{32}	8,439169	18,452502
σ_{41}	8,977491	15,937635
σ_{42}	8,043480	27,452020
σ_{51}	10,799251	8,312757
σ_{52}	11,738003	12,264684
σ_{61}	9,021083	15,863757
σ_{62}	9,408646	9,102590
w_0	-1,633834*10^6	-0,035294*10^8
w_1	-1,073892*10^6	2,322673*10^8
w_2	1,874673*10^6	0,082226*10^8
w_3	2,084700*10^6	-2,711903*10^8
w_4	-3,176845*10^6	0,318537*10^8
w_5	2,671739*10^6	0,037616*10^8
w_6	-0,749340*10^6	-0,013466*10^8

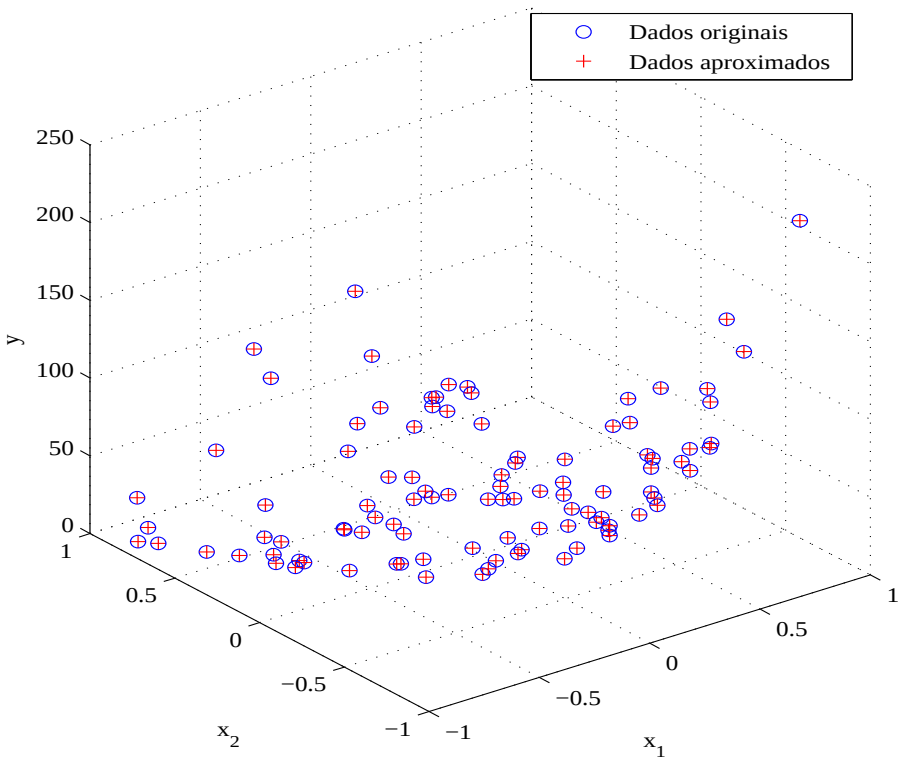

Figura 17: Dados originais e os aproximados pela rede RBF após o procedimento proposto referente ao Exemplo 2.

Tabela 7: Erro de aproximação da rede RBF para o Exemplo 2.

	Conjunto de Treinamento	**Conjunto de Teste**
CONTEXTO	**SEQ**	**SEQ**
Pré-ajuste	$5{,}5875*10^{4}$	$5{,}8192*10^{4}$
Refinamento	$5{,}5073*10^{-2}$	$3{,}9874*10^{-2}$

No procedimento via PSO para ajustar os valores das larguras das funções da camada escondida de redes RBF, os valores das partículas associadas ao procedimento poderiam ser inicializados aleatoriamente e o algoritmo correspondente poderia ajustá-los até obter-se a precisão desejada. Mas, com a utilização de valores pré-ajustados o algoritmo converge com um número de passos menor em relação a parâmetros inicializados aleatoriamente. Objetivando obter resultados mais exatos, os parâmetros do algoritmo de PSO, como os coeficientes cognitivo e social (a_1 e a_2), foram ajustados fazendo com que as partículas tenham deslocamentos menores no espaço de busca, e deste modo possibilitando atingir a melhor avaliação da função objetivo do algoritmo correspondente. O algoritmo foi executado vinte vezes para cada cenário considerado, sendo registrado o número de passos resultantes durante as execuções. Como critério de parada foi utilizado um valor de SEQ igual a 10^{-1} ou um número máximo de passos igual a 2000. O fator de inércia foi diminuído linearmente de 1,4 para 0,5. Os valores dos coeficientes de aceleração a_1 e a_2 foram ajustados em 1. As velocidades das partículas foram inicializadas aleatoriamente entre $vMin$ e $vMax$. A Tabela 8 compara o número de passos de convergência resultantes através da inicialização das partículas a partir dos valores prévios de larguras, e partindo-se de valores aleatórios. Observa-se que com a inicialização através de valores aleatórios, o algoritmo PSO necessita de mais passos para obter a mesma exatidão, ou então atinge o número máximo de passos estabelecido no algoritmo. Após as vinte execuções, o número médio de passos resultantes a partir da inicialização com valores pré-ajustados e a partir de valores aleatórios foi de 1226,2 e 1838,5, respectivamente.

Tabela 8: Número de passos resultantes.

Inicialização com valores prévios									
1228	1170	1261	1222	1311	1212	1269	1255	1111	1197
1035	855	1238	1115	2000	1083	1155	1201	1440	1166
Inicialização com valores aleatórios									
2000	2000	2000	1197	2000	2000	2000	2000	2000	2000
2000	2000	1209	2000	1217	2000	2000	2000	1147	2000

5.1.3 Exemplo 3 - Série de Preço de Arroba de Boi

Neste item, o procedimento proposto será aplicado a um exemplo de previsão de série temporal relacionada com o preço de arroba de boi, cujos dados estão ilustrados na Figura 18.

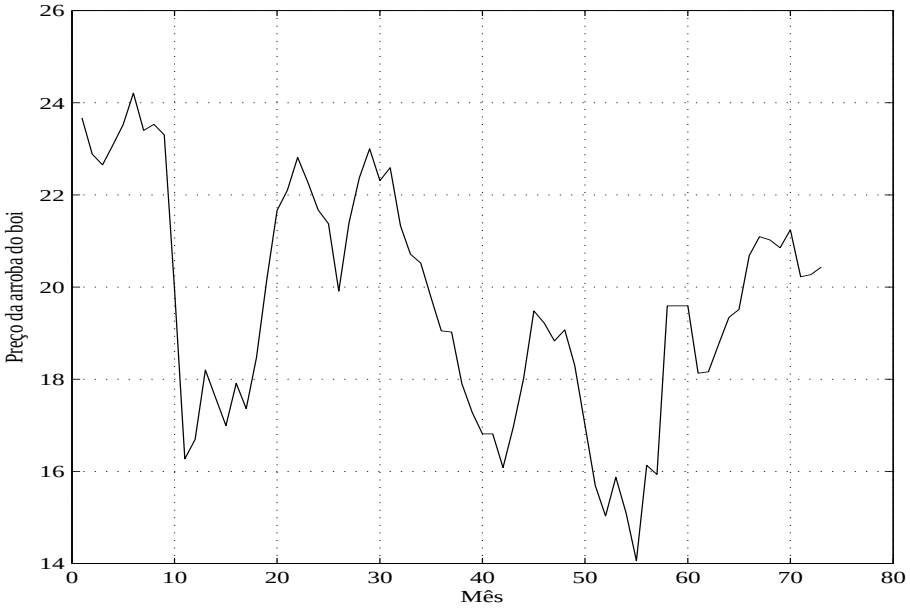

Figura 18: Dados originais da série temporal relativa ao Exemplo 3.

O conjunto de treinamento utilizado para pré-ajustar os parâmetros da rede RBF foi composto pelos dados referentes aos primeiros sessenta e dois meses da série. Para a previsão de um passo a frente, cada amostra de dados foi formada por duas observações de preço referentes aos dois meses anteriores ao mês que se deseja prever. Os valores dos coeficientes de aceleração a_1 e a_2 foram ajustados em 1 e as velocidades da partícula $vMax$ e $vMin$ foram limitadas em 5 e -5, respectivamente. O procedimento via pré-ajuste de parâmetros resultou em um tempo de processameto de 0,16 segundos. O procedimento proposto para refinamento de parâmetros de redes RBF foi aplicado, onde se utilizou como critério de parada do algoritmo um número máximo de passos igual a 2000. A rede RBF resultante apresentou um valor médio de SEQ de 28,14 após 2000 passos de treinamento. O tempo médio de processamento foi de 29,7 segundos. A Tabela 9 ilustra os valores dos parâmetros pré-ajustados para rede RBF e os valores refinados via PSO a partir do conjunto de dados de treinamento. Os valores dos centros das funções de base radial para os dois modelos de redes RBF consideradas foram: $c = [c_{11}, c_{12}, c_{21}, c_{22}, \ldots, c_{61}, c_{62}] = [22,183812, 22,303946, 17,332979, 17,254879, 15,611567, 15,440840, 23,349729, 23,392015, 18,989097, 18,944679, 21,003959, 20,616044]$. Os valores estimados pelos modelos foram comparados com os dados originais da série e estão ilustrados na Figura 19 e Figura 20.

Tabela 9: Valores dos parâmetros pré-ajustados e refinados da rede RBF referentes ao Exemplo 3.

Parâmetros	**Valores pré-ajustados**	**Valores refinados**
σ_{11}	11,245697	1,824484
σ_{12}	9,814081	11,859703
σ_{21}	11,360312	7,079993
σ_{22}	10,005365	4,601595
σ_{31}	8,932972	34,623346
σ_{32}	8,269446	6,191798
σ_{41}	10,073190	0,189253
σ_{42}	10,023970	34,262102
σ_{51}	10,325168	0,111568
σ_{52}	8,723710	0,807232
σ_{61}	10,311218	35,644022
σ_{62}	7,590768	27,412345
w_0	-76,359284	87,816476
w_1	345,775307	0,609819
w_2	184,920309	12,328390
w_3	-8,265429	-27,486378
w_4	-142,856323	-1,353230
w_5	-210,181730	-1,642026
w_6	-66,261684	-57,037667

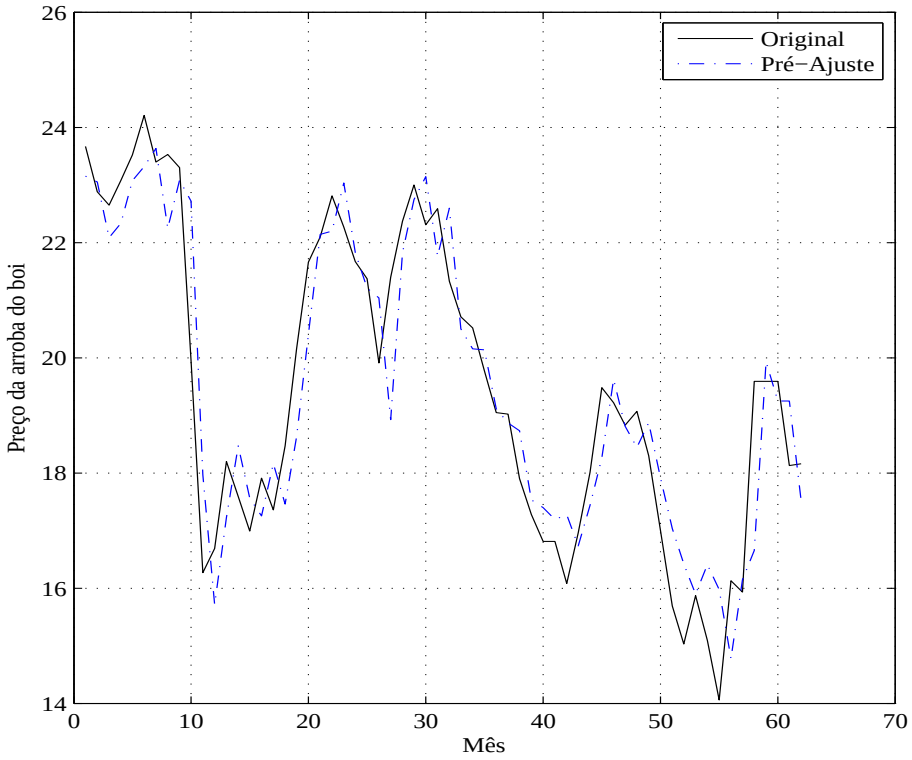

Figura 19: Dados originais da série e os aproximados pela rede RBF após o procedimento prévio de ajuste de parâmetros referente ao Exemplo 3.

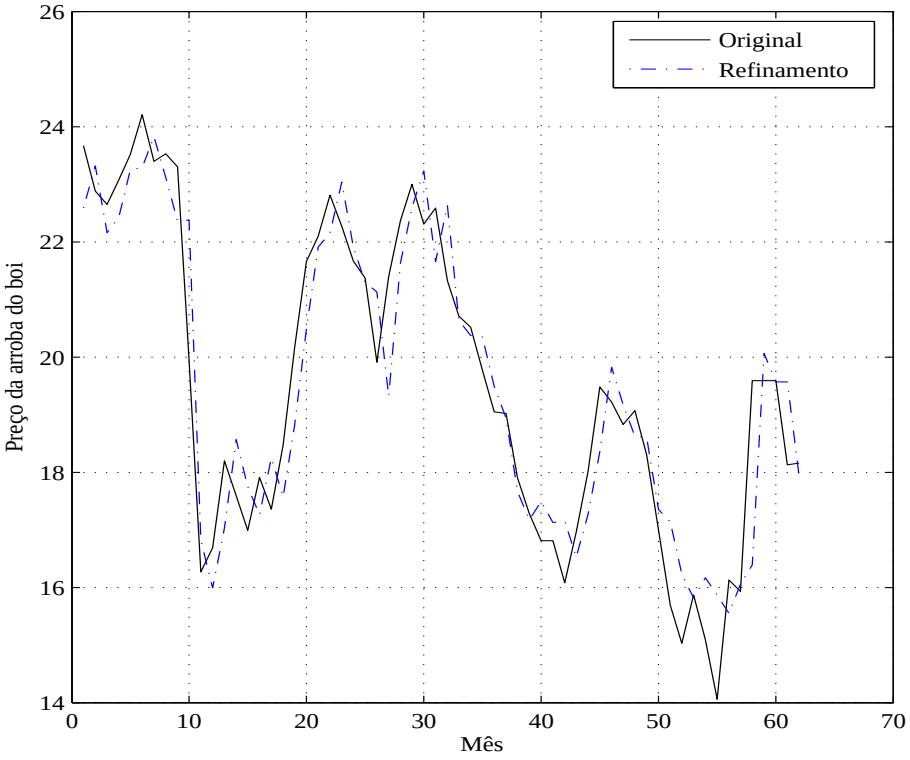

Figura 20: Dados originais da série e os aproximados pela rede RBF após o procedimento proposto referente ao Exemplo 3.

Para testar o modelo da rede RBF resultante foram utilizados os onze dados finais da série com a finalidade de verificar a capacidade de previsão da modelagem correspondente. A Figura 21 ilustra os resultados obtidos nas previsões, onde se têm os dados originais da série, as previsões da rede com os parâmetros previamente ajustados e com os parâmetros refinados via PSO.

A Tabela 10 contêm os valores dos erros considerados no experimento, onde foi adicionada a informação do erro percentual absoluto médio (EPAM). Observa-se que com a utilização do refinamento via PSO houve uma melhoria de 0,8% na capacidade de previsão da modelagem resultante.

Tabela 10: Erro da rede RBF para o Exemplo 3.

	Conjunto de Treinamento		**Conjunto de Teste**	
CONTEXTO	**SEQ**	**EPAM**	**SEQ**	**EPAM**
Pré-ajuste	31,6858	4,2375%	3,0207	3,1031%
Refinamento	28,0913	3,9452%	1,8707	2,2959%

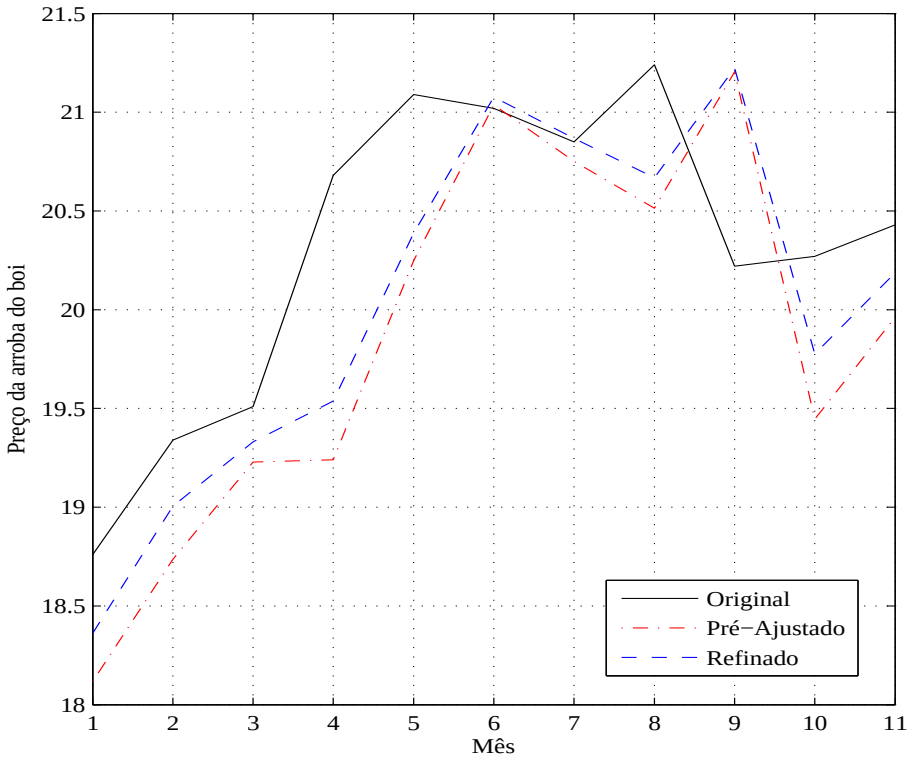

Figura 21: Dados originais da série e os da rede RBF com parâmetros pré-ajustados e refinados via PSO.

5.1.4 Exemplo 4 – Classificação de Padrões

Neste exemplo o procedimento proposto será aplicado no ajuste de parâmetros de uma rede RBF utilizada para a classificação de dados relacionados ao diagnostico de câncer de mama, cujas informações fazem parte do repositório UCI Machine Learning (BACHE; LICHMAN, 2013). As informações desta base de dados são compostas por nove atributos e 699 exemplos, onde 65,4% dos dados são classificações referentes a diagnósticos classificados como benignos e 34,5% como malignos. Alguns dados apresentaram inconsistências e foram removidos da base de dados, resultando em um conjunto com 683 amostras. As primeiras 349 amostras serão utilizadas para o procedimento de ajuste de parâmetros da rede RBF considerada, e as últimas 175 amostras para efeito de teste da rede resultante. Os dados de treinamento foram apresentados à rede de forma aleatória. Um valor de threshold igual a 0,5 foi utilizado na saída da rede para classificar a informação de saída da RBF como estado benigno ou maligno. Os resultados serão comparados com os obtidos pelo método proposto por Noman, Shamsuddin e Hassanien (2009), que utilizou a mesma proporção de amostras de treinamento e teste.

A estrutura da rede RBF utilizada neste experimento consiste de nove entradas, dois nós na camada escondida e uma unidade de saída. As informações de entrada da rede

são: espessura; uniformidade do tamanho da célula; uniformidade da forma da célula; aderência marginal; tamanho da célula epitelial; cromatina branda; núcleo descoberto; nucléolo normal; mitose. As velocidades da partícula $vMax$ e $vMin$ foram limitadas em 1 e -1 respectivamente.

Os procedimentos de pré-ajuste e refinamento de parâmetros da rede RBF foram aplicados. Utilizou-se como critério de parada do algoritmo de PSO um valor de erro médio quadrático (EMQ) menor que 0,005 ou um número máximo de passos igual a 2000. A rede RBF resultante apresentou após dez execuções do algoritmo um valor médio de EMQ de 0,0265 após o número máximo de passos de treinamento. O tempo de processamento via abordagem de pré-ajuste foi de 0,12 segundos, enquanto que o tempo médio de processamento via procedimento de refinamento foi de 50,4 segundos. Os valores na Tabela 11 indicam os parâmetros pré-ajustados para rede RBF e os valores refinados via PSO, a partir do conjunto de exemplos de treinamento. Os parâmetros referentes aos centros das funções de base radial para as duas modelagens foram: $c = [c_{11}, c_{12}, \ldots, c_{19}, c_{21}, c_{22}, \ldots, c_{29}] = [0,692646, 0,703056, 0,691083, 0,597203, 0,555467, 0,812875, 0,625149, 0,620446, 0,264020, 0,291770, 0,132338, 0,143656, 0,135286, 0,208987, 0,132398, 0,211677, 0,123858, 0,108299]$.

Tabela 11: Valores dos parâmetros pré-ajustados e refinados da rede RBF referentes ao Exemplo 4.

Parâmetros	**Valores pré-ajustados**	**Valores refinados**
σ_{11}	11,330792	49,845677
σ_{12}	8,976854	1,637906
σ_{13}	9,695303	0,854894
σ_{14}	8,572920	14,008969
σ_{15}	9,798474	1,433562
σ_{16}	8,525740	1,667283
σ_{17}	10,673202	1,196716
σ_{18}	8,079547	1,239226
σ_{19}	10,119156	0,512793
σ_{21}	12,408353	4,297305
σ_{22}	26,335127	10,636327
σ_{23}	21,505132	45,903744
σ_{24}	30,155745	9,669115
σ_{25}	25,840186	6,659443
σ_{26}	27,550582	6,643929
σ_{27}	16,211732	7,180254
σ_{28}	24,940500	7,597928
σ_{29}	41,705232	2,524060
w_0	82,479207	16,071959
w_1	58,583295	1,799035
w_2	-140,293121	-17,057478

A Tabela 12 contêm os valores referentes às etapas de treinamento e o percentual de acerto na classificação de diagnósticos. Os resultados obtidos pelos procedimentos de pré-ajuste e refinamento via PSO foram confrontados com os resultados encontrados por Noman, Shamsuddin e Hassanien (2009). Observa-se que utilizando o procedimento de refinamento houve uma melhoria de 24,8% no percentual de acertos em relação ao resultado encontrado por Noman, Shamsuddin e Hassanien (2009) e 1,15% em relação ao resultado encontrado utilizando o procedimento de pré-ajuste.

Tabela 12: Comparação dos resultados.

CONTEXTO	**Noman et al.**		**Pré-ajuste**		**Refinamento**	
	Treino	**Teste**	**Treino**	**Teste**	**Treino**	**Teste**
Passos	10000	-	1	-	2000	-
Classificação(%)	97,65	71,77	96,56	95,42	96,84	96,57

Uma segunda abordagem utilizada neste exemplo foi dividir os conjuntos de treinamento e teste em partes iguais, onde as primeiras 341 amostras foram utilizadas para o procedimento de ajuste de parâmetros da rede e as seguintes 341 amostras para efeito de teste da rede resultante. Os procedimentos de pré-ajuste e refinamento de parâmetros da rede RBF foram aplicados. Utilizando-se do mesmo critério de parada da abordagem anterior, a rede RBF resultante apresentou após dez execuções do algoritmo um valor médio de EMQ de 0,0266 após 2000 passos de treinamento. A Tabela 13 ilustra os valores dos parâmetros pré-ajustados para rede RBF e os valores refinados via PSO, a partir do conjunto de treinamento. Os valores referentes aos centros das funções de base radial para os dois modelos de rede foram: $c = [c_{11}, c_{12}, \ldots, c_{19}, c_{21}, c_{22}, \ldots, c_{29}] = [0.678187, 0, 693594, 0, 689836, 0, 584668, 0, 536870, 0, 778231, 0, 615053, 0, 595280, 0, 268146, 0, 298546, 0, 131776, 0, 141380, 0, 136420, 0, 211951, 0, 135695, 0, 209271, 0, 125844, 0, 111176]$.

A Tabela 14 traz o percentual de acerto na classificação de diagnósticos, confrontando os resultados obtidos pelo procedimento de pré-ajuste com o refinamento via PSO. Nota-se uma melhoria de 0,5% na classificação quando se utiliza o procedimento via PSO, e também um pequeno incremento de acertos (0,49%) em relação à tabela anterior.

Tabela 13: Valores dos parâmetros da segunda abordagem do Exemplo 4.

Parâmetros	**Valores pré-ajustados**	**Valores refinados**
σ_{11}	10,714678	19,851081
σ_{12}	10,055280	1,571837
σ_{13}	10,528070	1,252686
σ_{14}	8,134845	3,612527
σ_{15}	10,243011	2,316478
σ_{16}	8,446178	1,652485
σ_{17}	10,174372	1,665215
σ_{18}	8,027342	1,507054
σ_{19}	9,844451	1,018350
σ_{21}	12,937149	6,169153
σ_{22}	13,991633	10,702532
σ_{23}	15,930176	32,631041
σ_{24}	28,721439	11,628663
σ_{25}	23,413578	15,487031
σ_{26}	23,907565	13,113512
σ_{27}	17,850148	10,313565
σ_{28}	25,343032	27,663594
σ_{29}	35,015549	4,556311
w_0	25,392314	31,366074
w_1	71,508524	3,013624
w_2	-96,028134	-33,446354

Tabela 14: Comparação dos resultados da segunda abordagem.

CONTEXTO	**Pré-ajuste**		**Refinamento**	
	Treino	**Teste**	**Treino**	**Teste**
Passos	1	-	2000	-
Classificação(%)	97,65	96,56	98,53	97,06

6 Conclusão

Este capítulo conclui a dissertação e propõe sugestões para trabalhos futuros.

6.1 Considerações Finais e sugestões para Trabalhos Futuros

Como abordado no Capítulo 2, vários métodos de treinamento podem ser aplicados no ajuste dos parâmetros livres de uma rede RBF. Alguns destes métodos combinam estratégias não supervisionadas com supervisionadas durante o processo de aprendizagem.

O procedimento proposto nesta dissertação utiliza um algoritmo de PSO para realizar um refinamento nos valores das larguras das funções de entrada de redes RBF, cujos parâmetros iniciais são pré-selecionados pela abordagem descrita em Santos (2014). Alguns experimentos foram considerados para ilustrar a metodologia proposta.

Nas aplicações de aproximações de funções consideradas nos experimentos, observou-se através dos resultados encontrados que a exatidão das redes RBF com parâmetros refinados através do método proposto foi bem melhor em relação às redes com parâmetros apenas pré-ajustados. Para o primeiro exemplo considerado, uma precisão de 10^8 vezes na aproximação da função foi alcançada, enquanto que para o segundo exemplo, a exatidão foi de 10^6 vezes. Aplicações em aproximações de funções não lineares foram consideradas inicialmente. Na aplicação referente à previsão de valores de uma série temporal financeira, notou-se que com a utilização do refinamento via PSO houve uma melhoria de 0,8% na capacidade de previsão da modelagem resultante. Uma melhor exatidão nas estimativas das previsões pode ser obtida com a utilização um maior número de observações anteriores referentes às informações empregadas, onde as janelas temporais correspondentes podem ser determinadas via técnicas de auto-correlação, e também com a inclusão de variáveis exógenas que possam estar relacionadas com o processo de previsão considerado. Na aplicação de classificação de padrões notou-se uma pequena melhoria nos resultados das classificações quando se utiliza o procedimento via PSO. Com a utilização deste método, houve uma melhoria de 24,8% em relação ao método utilizado por Noman, Shamsuddin

e Hassanien (2009) e 1,15% em relação ao procedimento de pré-ajuste.

Para todos os experimentos considerados, o tempo de execução de cada procedimento abordado foi medido. Foi observado que o tempo de processamento utilizando o procedimento de refinamento de parâmetros é maior em relação à utilização do procedimento de pré-ajuste. Apesar desta diferença de tempo, a melhoria trazida pelo método de refinamento compensou o tempo de execução.

Assim, pode-se concluir dos experimentos exemplificados neste trabalho, que a abordagem adotada proporciona uma melhoria na exatidão dos modelos resultantes em comparação ao método de pré-ajuste de parâmetros, e alguns outros citados na revisão bibliográfica.

Para trabalhos futuros são propostas as seguintes sugestões:

- Considerar outras aplicações de modelagem, tais como de sistemas dinâmicos com características não lineares, outros exemplos de séries temporais e de classificação de padrões.
- A utilização de técnicas de agrupamento que possibilitem a determinação ótima dos números de clusters relacionados com o número de funções de base radial especificadas para uma determinada rede RBF. Desta forma espera-se obter estruturas mais reduzidas para as redes projetadas.
- Utilização do algoritmo de PSO para refinar também os valores dos centros das funções de base radial de redes RBF.
- Aplicação de outros algoritmos de otimização para o ajuste dos parâmetros de redes RBF, como aqueles baseados em colônias de formigas, enxame de abelhas, cardumes de peixes, e outros.

Referências

AIK, L. E.; ZAINUDDIN, Z. An improved fast training algorithm for rbf networks using symmetry-based fuzzy c-means clustering 1. Citeseer, 2008.

ALZOHAIRY, T. A. Direct adaptive control of unknown nonlinear systems using radial basis function networks with gradient descent and k-means. *system*, v. 2, p. 1, 2011.

BACHE, K.; LICHMAN, M. *UCI Machine Learning Repository*. 2013. Disponível em: <http://archive.ics.uci.edu/ml>.

BARALDI, A. et al. Rbf two-stage learning networks exploiting supervised data in the selection of hidden unit parameters: An application to sar data classification. In: IEEE. *Geoscience and Remote Sensing Symposium, 2000. Proceedings. IGARSS 2000. IEEE 2000 International*. [S.l.], 2000. v. 2, p. 672–674.

BEZDEK, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

BEZDEK, J. C.; EHRLICH, R.; FULL, W. Fcm: The fuzzy< i> c</i>-means clustering algorithm. *Computers & Geosciences*, Elsevier, v. 10, n. 2, p. 191–203, 1984.

BISHOP, C. M. *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc., 1995.

CAIQING, Z.; PEIYU, M. Short-term electricity price forecasting based on pso algorithm and rbf neural network algorithm. In: IEEE. *Measuring Technology and Mechatronics Automation (ICMTMA), 2010 International Conference on*. [S.l.], 2010. v. 3, p. 334–337.

CHANG, W.-Y. An rbf neural network combined with ols algorithm and genetic algorithm for short-term wind power forecasting. *Journal of Applied Mathematics*, Hindawi Publishing Corporation, v. 2013, 2013.

CHANGBING, L.; WEI, H. Application of genetic algorithm-rbf neural network in water environment risk prediction. In: IEEE. *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*. [S.l.], 2010. v. 7, p. V7–239.

CHEN, J. Y.; QIN, Z.; JIA, J. A pso-based subtractive clustering technique for designing rbf neural networks. In: IEEE. *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. [S.l.], 2008. p. 2047–2052.

CHEN, Z.; QIAN, P. Application of pso-rbf neural network in network intrusion detection. In: IEEE. *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on*. [S.l.], 2009. v. 1, p. 362–364.

CHUN-TAO, M.; KUN, W.; LI-YONG, Z. A new training algorithm for rbf neural network based on pso and simulation study. In: IEEE. *Computer Science and Information Engineering, 2009 WRI World Congress on*. [S.l.], 2009. v. 4, p. 641–645.

CLERC, M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: IEEE. *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. [S.l.], 1999. v. 3.

CUI, L.; WANG, C.; YANG, B. Application of rbf neural network improved by pso algorithm in fault diagnosis. *Journal of Theoretical and Applied Information Technology*, v. 46, 2012.

DELICAN, Y.; OZYILMAZ, L.; YILDIRIM, T. Evolutionary algorithms based rbf neural networks for parkinson's disease diagnosis. In: IEEE. *Electrical and Electronics Engineering (ELECO), 2011 7th International Conference on*. [S.l.], 2011. p. II–311.

DING, S. et al. An optimizing method of rbf neural network based on genetic algorithm. *Neural Computing and Applications*, Springer, v. 21, n. 2, p. 333–336, 2012.

DUNN, J. C. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Taylor & Francis, 1973.

EBERHART, R. C.; SHI, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In: IEEE. *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*. [S.l.], 2000. v. 1, p. 84–88.

EBERHART, R. C.; SHI, Y. Particle swarm optimization: developments, applications and resources. In: IEEE. *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*. [S.l.], 2001. v. 1, p. 81–86.

ESMAEILI, A.; MOZAYANI, N. Adjusting the parameters of radial basis function networks using particle swarm optimization. In: IEEE. *Computational Intelligence for Measurement Systems and Applications, 2009. CIMSA'09. IEEE International Conference on*. [S.l.], 2009. p. 179–181.

FATHI, V.; MONTAZER, G. A. An improvement in rbf learning algorithm based on pso for real time applications. *Neurocomputing*, Elsevier, v. 111, p. 169–176, 2013.

GOLBABAI, A.; SAFDARI-VAIGHANI, A. Width optimization of gaussian function by genetic algorithm in rbf networks. *World Journal of Modelling and Simulation*, v. 7, n. 4, p. 307–311, 2011.

GUERRA, F. A.; COELHO, L. Radial basis neural network learning based on particle swarm optimization to multistep prediction of chaotic lorenz's system. In: IEEE. *Hybrid Intelligent Systems, 2005. HIS'05. Fifth International Conference on*. [S.l.], 2005. p. 3–pp.

HAYKIN, S. *Redes Neurais: princípios e prática*. 2nd. ed. [S.l.]: Bookman, 2001.

HU, X.; SHI, Y.; EBERHART, R. C. Recent advances in particle swarm. In: *IEEE congress on evolutionary computation*. [S.l.: s.n.], 2004. v. 1, p. 90–97.

JANG, J.-S. R.; SUN, C.-T.; MIZUTANI, E. Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [book review]. *Automatic Control, IEEE Transactions on*, IEEE, v. 42, n. 10, p. 1482–1484, 1997.

JI, Y.; ZHOU, W.; YU, L. Rbf neural networks base on particle swarm optimization and its application in control system of flatness and gauge. In: IEEE. *Natural Computation (ICNC), 2012 Eighth International Conference on*. [S.l.], 2012. p. 312–315.

JIN-YUE, L.; BAO-LING, Z. Research on the non-linear function fitting of rbf neural network. In: IEEE. *Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on*. [S.l.], 2013. p. 842–845.

JUN, L.; ZUHUA, G. Network traffic prediction using radial basis function neural network optimized by ant colony algorithm. 2014.

KAYHAN, G.; OZDEMIR, A. E.; EMINOGLU, İ. Reviewing and designing pre-processing units for rbf networks: initial structure identification and coarse-tuning of free parameters. *Neural Computing and Applications*, Springer, v. 22, n. 7-8, p. 1655–1666, 2013.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proceedings of 1995 IEEE International Conference on Neural Networks*. [S.l.: s.n.], 1995. p. 1942–1948.

KENNEDY, J.; MENDES, R. Population structure and particle swarm performance. IEEE computer Society, 2002.

KORANI, W. *Particle Swarm Optimization*. MATLAB Central File Exchange, June 2008. Disponível em: <`http://www.mathworks.com/matlabcentral/fileexchange/20205-particle-swarm-optimization/content/PSO_Wael/PSO/PSO.m`>.

LI, B.; CONG, L.; ZHANG, W. Research on optimized rbf neural network based on ga for sewage treatment. In: IEEE. *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2013 5th International Conference on*. [S.l.], 2013. v. 2, p. 520–523.

LI, S. et al. Study of personal credit evaluation method based on pso-rbf neural network model. *American Journal of Industrial and Business Management*, Scientific Research Publishing, v. 3, p. 429, 2013.

LI, X.-f.; DONG, J.-h.; ZHANG, Y.-z. Modeling and applying of rbf neural network based on fuzzy clustering and pseudo-inverse method. In: IEEE. *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*. [S.l.], 2009. p. 1–4.

LIANGHAI, W.; YIMING, C. Study of prediction based on rbf neural network optimized by pso. In: IEEE. *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*. [S.l.], 2010. v. 6, p. V6–496.

LIN, G.-F.; WU, M.-C. An rbf network with a two-step learning algorithm for developing a reservoir inflow forecasting model. *Journal of Hydrology*, Elsevier, v. 405, n. 3, p. 439–450, 2011.

LIN, Y.-H.; TSAI, M.-S.; CHEN, C.-S. Applications of fuzzy classification with fuzzy c-means clustering and optimization strategies for load identification in nilm systems. In: IEEE. *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*. [S.l.], 2011. p. 859–866.

LIU, Z.; CHEN, J.; SONG, C. A new rbf neural network with ga-based fuzzy c-means clustering algorithm for sins fault diagnosis. In: IEEE. *Control and Decision Conference, 2009. CCDC'09. Chinese*. [S.l.], 2009. p. 208–211.

MARUZUKI, M.; ISHAK, S.; SETUMIN, S. Malaysia car plate recognition based on rbf neural network and particle swarm optimization. In: IEEE. *Control System, 2012 International Conference on*. [S.l.], 2012.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

MENG, K. et al. Electricity reference price forecasting with fuzzy c-means and immune algorithm. In: IEEE. *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. [S.l.], 2007. p. 2337–2343.

MING, Z. Y.; BIN, Z. Y.; ZHONG, L. L. Application of genetic algorithm and rbf neural network in network flow prediction. In: IEEE. *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*. [S.l.], 2010. v. 2, p. 298–301.

NIROS, A. D.; TSEKOURAS, G. E. A novel training algorithm for rbf neural network using a hybrid fuzzy clustering approach. *Fuzzy Sets and Systems*, Elsevier, v. 193, p. 62–84, 2012.

NITANDA, N.; HASEYAMA, M.; KITAJIMA, H. Audio-cut detection and audio-segment classification using fuzzy c-means clustering. In: IEEE. *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*. [S.l.], 2004. v. 4, p. iv–325.

NOMAN, S.; SHAMSUDDIN, S. M.; HASSANIEN, A. E. Hybrid learning enhancement of rbf network with particle swarm optimization. In: *Foundations of Computational, Intelligence Volume 1*. [S.l.]: Springer, 2009. p. 381–397.

OLIVEIRA, J. V. de; PEDRYCZ, W. et al. *Advances in fuzzy clustering and its applications*. [S.l.]: Wiley Online Library, 2007.

OZDEMIR, A. E.; EMINOGLU, I. A two-pass hybrid training algorithm for rbf networks. In: IEEE. *Electrical and Electronics Engineering (ELECO), 2013 8th International Conference on*. [S.l.], 2013. p. 617–620.

PAN, Y. et al. A forecasting model of rbf neural network based on genetic algorithms optimization. In: IEEE. *Natural Computation (ICNC), 2011 Seventh International Conference on*. [S.l.], 2011. v. 1, p. 48–51.

POLI, R. An analysis of publications on particle swarm optimization applications. *Essex, UK: Department of Computer Science, University of Essex*, 2007.

POLI, R.; KENNEDY, J.; BLACKWELL, T. Particle swarm optimization. *Swarm intelligence*, Springer, v. 1, n. 1, p. 33–57, 2007.

POWELL, M. J. D. Algorithms for approximation. In: MASON, J. C.; COX, M. G. (Ed.). New York, NY, USA: Clarendon Press, 1987. cap. Radial Basis Functions for Multivariable Interpolation: A Review, p. 143–167.

QING-WEI, Z.; ZHI-HAI, X.; JIAN, W. Prediction of electricity consumption based on genetic algorithm-rbf neural network. In: IEEE. *Advanced Computer Control (ICACC), 2010 2nd International Conference on*. [S.l.], 2010. v. 5, p. 339–342.

ROCHA, T. et al. Tutorial sobre fuzzy-c-means e fuzzy learning vector quantization: Abordagens híbridas para tarefas de agrupamento e classificação. *Revista de Informática Teórica e Aplicada*, v. 19, p. 120–163, 2012.

ROSENBROCK, H. H. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, Br Computer Soc, v. 3, n. 3, p. 175–184, 1960.

SANTOS, F. A. O. *Uma Abordagem para Parametrização de Redes RBF baseada na Combinação de Procedimentos não supervisionados e uma Nova Proposição de Escalonamento de Parâmetros*. 2014. Qualificação de Doutorado (UNIFEI).

SCHILLING, R. J.; JR, J. J. C.; AL-AJLOUNI, A. F. Approximation of nonlinear systems with radial basis function neural networks. *Neural Networks, IEEE Transactions on*, IEEE, v. 12, n. 1, p. 1–15, 2001.

SCHWENKER, F.; KESTLER, H. A.; PALM, G. Three learning phases for radial-basis-function networks. *Neural networks*, Elsevier, v. 14, n. 4, p. 439–458, 2001.

SHI, Y.; EBERHART, R. A modified particle swarm optimizer. In: IEEE. *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. [S.l.], 1998. p. 69–73.

SHUZHI, G.; JIE, S.; XIANWEN, G. Soft-sensor modeling of rectification of vinyl chloride based on improved pso-rbf neural network. In: IEEE. *Control and Decision Conference (CCDC), 2012 24th Chinese*. [S.l.], 2012. p. 1122–1126.

SILVA, I. da; SPATTI, D.; FLAUZINO, R. *Redes Neurais Artificiais para Engenharia e Ciências Aplicadas*. [S.l.]: Artliber Editora Ltda, 2010.

SONG, S. et al. The value of short-time traffic flow prediction in the pso-rbfnn study. In: IEEE. *Computer Science and Information Processing (CSIP), 2012 International Conference on*. [S.l.], 2012. p. 1051–1054.

SU-WEI, G. Software component retrieval method based on pso-rbf neural network. In: IEEE. *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*. [S.l.], 2010. v. 7, p. V7–339.

SUN, B.; LI, T.-K. Forecasting and identification of stock market based on modified rbf neural network. In: IEEE. *Industrial Engineering and Engineering Management (IE&EM), 2010 IEEE 17Th International Conference on*. [S.l.], 2010. p. 424–427.

SUN, Y. et al. Optimal partition algorithm of the rbf neural network and its application to financial time series forecasting. *Neural Computing & Applications*, Springer, v. 14, n. 1, p. 36–44, 2005.

TAO, W.; JIANPING, L.; BINGXIN, L. Research of lung cancer screening algorithm based on rbf neural network. In: IEEE. *Computer and Management (CAMAN), 2011 International Conference on*. [S.l.], 2011. p. 1–4.

UROLAGIN, S.; PREMA, K.; REDDY, N. S. Generalization capability of artificial neural network incorporated with pruning method. In: *Advanced Computing, Networking and Security*. [S.l.]: Springer, 2012. p. 171–178.

WANG, P.; XIE, L.; SUN, Y. Application of pso algorithm and rbf neural network in electrical impedance tomography. In: IEEE. *Electronic Measurement & Instruments, 2009. ICEMI'09. 9th International Conference on*. [S.l.], 2009. p. 2–517.

WEI, Z.; YANG, S. Wavelet neural networks model used for runoff forecast based on fuzzy c-means clustering. In: IEEE. *Biomedical Engineering and Informatics, 2009. BMEI'09. 2nd International Conference on*. [S.l.], 2009. p. 1–5.

WU, J.-D.; LIU, J.-C. A forecasting system for car fuel consumption using a radial basis function neural network. *Expert Systems with Applications*, Elsevier, v. 39, n. 2, p. 1883–1888, 2012.

XUE-QIN, S.; TONG-DI, H. Water quality evaluation based on rbf neural network with parameters optimized by pso algorithm. In: IEEE. *Engineering and Technology (S-CET), 2012 Spring Congress on*. [S.l.], 2012. p. 1–4.

YAN, X.-B. et al. Time series forecasting with rbf neural network. In: IEEE. *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*. [S.l.], 2005. v. 8, p. 4680–4683.

YU-LIANG, Q. et al. Fault diagnosis for generator unit based on rbf neural network optimized by ga-pso. In: IEEE. *Natural Computation (ICNC), 2012 Eighth International Conference on*. [S.l.], 2012. p. 233–236.

ZADEH, L. A. Fuzzy sets. *Information and control*, Elsevier, v. 8, n. 3, p. 338–353, 1965.

ZHANG, R. et al. Improved gap-rbf network for classification problems. *Neurocomputing*, Elsevier, v. 70, n. 16, p. 3011–3018, 2007.

ZHU, Y.; HE, Y. Short-term load forecasting model using fuzzy c means based radial basis function network. In: IEEE. *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*. [S.l.], 2006. v. 1, p. 579–582.

ZIYANG, Z. et al. Learning method of rbf network based on fcm and aco. In: IEEE. *Control and Decision Conference, 2008. CCDC 2008. Chinese*. [S.l.], 2008. p. 102–105.

APÊNDICE A – Programa para construção do procedimento prévio e proposto para o ajuste dos parâmetros da rede RBF referente aos exemplos

A implementação do procedimento prévio e proposto para o ajuste dos parâmetros da rede RBF referente aos exemplos citados nesta dissertação foi realizada no software MatLab R2013a. Os códigos apresentados nas listagens são utilizados para a construção dos procedimentos referentes ao Exemplo 1 dos experimentos. Para os demais exemplos os códigos são semelhantes. A implementação do algoritmo de PSO foi baseada no código de Korani (2008).

Listagen A.1: Geração dos dados de treinamento e teste

```
% Dados da Funcao y = x1*x2 + x2^2
clear all;
% Dados da Funcao
np = 200;
for n=1:np
    s1 = rand; s2= rand;

    if s1 < 0.5
        s1 = -1;
    else
        s1 = 1;
    end

    if s2 < 0.5
        s2 = -1;
    else
        s2 = 1;
    end
```

```
20      x1 = s1*rand; x2 = s2*rand;
        Vetx1(n)=x1; Vetx2(n)=x2;
        y = x1*x2 + x2^2;
        Vety(n)=y;

25  end

    plot3(Vetx1,Vetx2,Vety,'k*');grid

    x1 = Vetx1(1:100); x2 = Vetx2(1:100); y = Vety(1:100);
30  DadosTreino = [x1' x2' y'];
    save DadosTreino;

    x1 = Vetx1(101:200); x2 = Vetx2(101:200); y = Vety(101:200);
    DadosTeste = [x1' x2' y'];
35  save DadosTeste;
```

Listagen A.2: Determinação dos centros e desvios da rede RBF

```
    % Dados dos Clusters (via Fuzzy C-Means) da funcao linear x1x2+x2^2
    clear all;

5   % Dados da funcao
    load DadosTreino;
    DadosFunc = DadosTreino;
    x1=DadosFunc(:,1); x2=DadosFunc(:,2); y = DadosFunc(:,3);

10  % Aplicacao do algoritmo Fuzzy C-Means
    dados = [x1 x2 y];
    [centros,U,obj_fcn] = fcm(dados,6); % Definido 6 Agrupamentos
    maxU = max(U);
    index1 = find(U(1,:) == maxU);
15  index2 = find(U(2,:) == maxU);
    index3 = find(U(3,:) == maxU);
    index4 = find(U(4,:) == maxU);
    index5 = find(U(5,:) == maxU);
    index6 = find(U(6,:) == maxU);

20

    % Indicacoes Graficas dos Agupamentos Resultantes
    plot3(dados(index1,1),dados(index1,2),dados(index1,3),'r.','MarkerSize',12)
    hold on
    plot3(dados(index2,1),dados(index2,2),dados(index2,3),'b.','MarkerSize',12)
```

```
plot3(dados(index3,1),dados(index3,2),dados(index3,3),'g.','MarkerSize',12)
plot3(dados(index4,1),dados(index4,2),dados(index4,3),'k.','MarkerSize',12)
plot3(dados(index5,1),dados(index5,2),dados(index5,3),'y.','MarkerSize',12)
plot3(dados(index6,1),dados(index6,2),dados(index6,3),'c.','MarkerSize',12)
xlabel('x1'); ylabel('x2'); zlabel('y');
plot3(centros(:,1),centros(:,2),centros(:,3),'ko', 'MarkerSize',12,'
    LineWidth',2)
grid; hold off;

% Centros dos Clusters
c1a=centros(1,1); c2a=centros(1,2);
c1b=centros(2,1); c2b=centros(2,2);
c1c=centros(3,1); c2c=centros(3,2);
c1d=centros(4,1); c2d=centros(4,2);
c1e=centros(5,1); c2e=centros(5,2);
c1f=centros(6,1); c2f=centros(6,2);

% Desvios padroes normais dos dados dos agrupamentos correspondentes
d1an=std(dados(index1,1)); d2an=std(dados(index1,2));
d1bn=std(dados(index2,1)); d2bn=std(dados(index2,2));
d1cn=std(dados(index3,1)); d2cn=std(dados(index3,2));
d1dn=std(dados(index4,1)); d2dn=std(dados(index4,2));
d1en=std(dados(index5,1)); d2en=std(dados(index5,2));
d1fn=std(dados(index6,1)); d2fn=std(dados(index6,2));

% Desvios padroes modificados (escalados) para as RBF
d1a=0.5*abs(max(dados(index1,1))-min(dados(index1,1)))/d1an; d2a=0.5*abs(
    max(dados(index1,2))-min(dados(index1,2)))/d2an;
d1b=0.5*abs(max(dados(index2,1))-min(dados(index2,1)))/d1bn; d2b=0.5*abs(
    max(dados(index2,2))-min(dados(index2,2)))/d2bn;
d1c=0.5*abs(max(dados(index3,1))-min(dados(index3,1)))/d1cn; d2c=0.5*abs(
    max(dados(index3,2))-min(dados(index3,2)))/d2cn;
d1d=0.5*abs(max(dados(index4,1))-min(dados(index4,1)))/d1dn; d2d=0.5*abs(
    max(dados(index4,2))-min(dados(index4,2)))/d2dn;
d1e=0.5*abs(max(dados(index5,1))-min(dados(index5,1)))/d1en; d2e=0.5*abs(
    max(dados(index5,2))-min(dados(index5,2)))/d2en;
d1f=0.5*abs(max(dados(index6,1))-min(dados(index6,1)))/d1fn; d2f=0.5*abs(
    max(dados(index6,2))-min(dados(index6,2)))/d2fn;

% Salva os dados
DadosPar = [c1a;c2a;c1b;c2b;c1c;c2c;c1d;c2d;c1e;c2e;c1f;c2f;d1a;d2a;d1b;
d2b;d1c;d2c;d1d;d2d;d1e;d2e;d1f;d2f];
save DadosPar;
```

Listagen A.3: Aplicação do procedimento prévio e proposto via PSO

```
% Programa principal
%Criacao da rede RBF para a aproximacao da funcao nao linear y=f(x1,x2)
clear all;
tic
% Carrega os dados de treinamento da rede
load DadosTreino;
DadosFunc = DadosTreino;
x1=DadosFunc(:,1); x2=DadosFunc(:,2); y=DadosFunc(:,3);

% Carrega os parametros (centros e desvios) da RBF via tecnica de
    agrupamento
load DadosPar;
c1a=DadosPar(1); c2a=DadosPar(2); c1b=DadosPar(3); c2b=DadosPar(4);
    c1c=DadosPar(5); c2c=DadosPar(6);
c1d=DadosPar(7); c2d=DadosPar(8); c1e=DadosPar(9); c2e=DadosPar(10);
    c1f=DadosPar(11); c2f=DadosPar(12);
d1a=DadosPar(13); d2a=DadosPar(14); d1b=DadosPar(15); d2b=DadosPar(16);
    d1c=DadosPar(17); d2c=DadosPar(18);
d1d=DadosPar(19); d2d=DadosPar(20); d1e=DadosPar(21); d2e=DadosPar(22);
    d1f=DadosPar(23); d2f=DadosPar(24);

%Fator multiplicativo
fm = 6;
d1a=fm*d1a; d2a=fm*d2a;
d1b=fm*d1b; d2b=fm*d2b;
d1c=fm*d1c; d2c=fm*d2c;
d1d=fm*d1d; d2d=fm*d2d;
d1e=fm*d1e; d2e=fm*d2e;
d1f=fm*d1f; d2f=fm*d2f;

% Calcula os valores das funcoes de base radial (ativacao) para cada
    amostra de treinamento
f1 = exp(-((x1-c1a).*(x1-c1a)/(2*d1a^2)+(x2-c2a).*(x2-c2a)/(2*d2a^2)));
f2 = exp(-((x1-c1b).*(x1-c1b)/(2*d1b^2)+(x2-c2b).*(x2-c2b)/(2*d2b^2)));
f3 = exp(-((x1-c1c).*(x1-c1c)/(2*d1c^2)+(x2-c2c).*(x2-c2c)/(2*d2c^2)));
f4 = exp(-((x1-c1d).*(x1-c1d)/(2*d1d^2)+(x2-c2d).*(x2-c2d)/(2*d2d^2)));
f5 = exp(-((x1-c1e).*(x1-c1e)/(2*d1e^2)+(x2-c2e).*(x2-c2e)/(2*d2e^2)));
f6 = exp(-((x1-c1f).*(x1-c1f)/(2*d1f^2)+(x2-c2f).*(x2-c2f)/(2*d2f^2)));
```

```
35
   [zL,zC]=size(y);

   % Regressores
   Fi=[f1 f2 f3 f4 f5 f6 ones(1,zL)']; Y=y;
40
   % Calcula os pesos da camada de saida da rede
   W=pinv(G)*Y;
   w1=W(1); w2=W(2); w3=W(3); w4=W(4); w5=W(5); w6=W(6); w0=W(7);

45 % Calcula a saida da rede para cada amostra de treinamento (modelo
       resultante)
   ye = w0 + w1*f1 + w2*f2 + w3*f3 +w4*f4 + w5*f5 + w6*f6;

   % Calculo da somatoria dos erros quadraticos (SEQ) sobre o conjunto de
       treinamento
   er =y-ye; eq=er.*er;
50 seqInicialTreino = 0.5*sum(eq)

   % Carrega os dados de teste da rede
   load DadosTeste;
   DadosFunc = DadosTeste;
55 x1=DadosFunc(:,1); x2=DadosFunc(:,2); y=DadosFunc(:,3);

   % Calcula os valores das funcoes de base radial (ativacao) para cada
       amostra de teste
   f1 = exp(-((x1-c1a).*(x1-c1a)/(2*d1a^2)+(x2-c2a).*(x2-c2a)/(2*d2a^2)));
   f2 = exp(-((x1-c1b).*(x1-c1b)/(2*d1b^2)+(x2-c2b).*(x2-c2b)/(2*d2b^2)));
60 f3 = exp(-((x1-c1c).*(x1-c1c)/(2*d1c^2)+(x2-c2c).*(x2-c2c)/(2*d2c^2)));
   f4 = exp(-((x1-c1d).*(x1-c1d)/(2*d1d^2)+(x2-c2d).*(x2-c2d)/(2*d2d^2)));
   f5 = exp(-((x1-c1e).*(x1-c1e)/(2*d1e^2)+(x2-c2e).*(x2-c2e)/(2*d2e^2)));
   f6 = exp(-((x1-c1f).*(x1-c1f)/(2*d1f^2)+(x2-c2f).*(x2-c2f)/(2*d2f^2)));

65 % Calcula a saida da rede para cada amostra de teste
   ye = w0 + w1*f1 + w2*f2 + w3*f3 +w4*f4 + w5*f5 + w6*f6;

   [nL,nC]=size(ye);
   vn=1:nL;
70
   % Calculo da somatoria dos erros quadraticos (SEQ) sobre o conjunto de
       teste
   er =y-ye; eq=er.*er;
   seqInicialTeste = 0.5*sum(eq)
```

```
%Tempo de processamento do procedimento de pre-ajuste de parametros
tempoGasto=toc

%##################################OTIMIZACAO COM PSO#################################
tic
seqTreino = zeros(10,1);
seqTeste = zeros(10,1);
numEpocas=ones(10,1);

%Executa o algoritmo PSO por t vezes
for t=1:10

% Carrega os dados de treinamento da rede
load DadosTreino;
DadosFunc = DadosTreino;
x1=DadosFunc(:,1); x2=DadosFunc(:,2); y=DadosFunc(:,3);

n = 50;              % tamanho do exame
num_passos = 2000; % maximo numero de passos
dim = 12;            % dimensao da particula

wmax=0.9; %fator de inercia inicial
wmin=0.4; %fator de inercia final
a1=2; %coeficiente de aceleracao (social)
a2=2; %coeficiente de aceleracao (cognitivo)
tol=1e-06; %define criterio de parada

%Inicializa o fitness das particulas
fitness=0*ones(n,num_passos);
fitness_atual =0*ones(n,1);

%Inicializacao das posicoes das particulas com os valores de desvios
    calculados previamente
Ad1=d1a*ones(1,n)';Ad2=d2a*ones(1,n)';
Bd1=d1b*ones(1,n)';Bd2=d2b*ones(1,n)';
Cd1=d1c*ones(1,n)';Cd2=d2c*ones(1,n)';
Dd1=d1d*ones(1,n)';Dd2=d2d*ones(1,n)';
Ed1=d1e*ones(1,n)';Ed2=d2e*ones(1,n)';
Fd1=d1f*ones(1,n)';Fd2=d2f*ones(1,n)';

posicao_atual = [Ad1 Ad2 Bd1 Bd2 Cd1 Cd2 Dd1 Dd2 Ed1 Ed2 Fd1 Fd2]';

```

```
%Define o limite do espaco de busca
pMin = 1e-06;
pMax = 1000;
%Define o limite da velocidade
vMax = 10;
vMin = -vMax;

%Inicializacao da velocidade da particula
velocidade = vMax*rand(dim,n);


%Calcula fitness inicial das particulas
for i=1:n
        fitness_atual(i) = rbf_fitness(posicao_atual(:,i),DadosFunc,
            DadosPar) ;
end

%Inicializa o PBEST
pbest_fitness  = fitness_atual ;
pbest_pos  = posicao_atual ;

%Inicializa o GBEST
[gbest_fitness,g] = min(pbest_fitness) ;
for i=1:n
    gbest_pos(:,i) = pbest_pos(:,g) ;
end

for iter=1:num_passos

%Decrementa linearmente o fator de inercia
w = wmax - ((wmax-wmin)/num_passos)*iter;

%Calcula nova velocidade da particula
for i=1:n
    velocidade(:,i) = w*velocidade(:,i) + a1*(rand*(pbest_pos(:,i)-
        posicao_atual(:,i))) + a2*(rand*(gbest_pos(:,i)-posicao_atual(:,i)))
        ;
end

%Limita a velocidade da particula
for k=1:n
    index = velocidade(:,k)>vMax;
    velocidade(index,k)=vMax;
```

```
        index = velocidade(:,k)<vMin;
        velocidade(index,k)=vMin;
    end

    %Calcula nova posicao da particula
    posicao_atual = posicao_atual + velocidade;

    %Limita o espaco de busca da particula
    for k=1:n
        index = posicao_atual(:,k)>pMax;
        posicao_atual(index,k)=pMax;

        index = posicao_atual(:,k)<pMin;
        posicao_atual(index,k)=pMin;
    end

    %Avalia novo fitness da particula
    for i=1:n,
            fitness_atual(i) = rbf_fitness(posicao_atual(:,i),DadosFunc,
                DadosPar) ;
    end

    %Calcula pbest
    for i=1:n
        if fitness_atual(i) < pbest_fitness(i)
            pbest_fitness(i)  = fitness_atual(i);
            pbest_pos(:,i) = posicao_atual(:,i);
        end
    end

    [current_gbest_fitness,g] = min(pbest_fitness);

    %Calcula gbest
    if current_gbest_fitness < gbest_fitness
        gbest_fitness = current_gbest_fitness;
        for i=1:n
            gbest_pos(:,i) = pbest_pos(:,g);
        end

    end

    %Criterio de parada
```

```
    if(gbest_fitness<tol)
        break
    end

end %fim do algoritmo

%Carrega novos desvios
d=gbest_pos(:,1);

%Armazena gBest do algoritmo na execucao t (valor do SEQ correspondente)
seqTreino(t) = gbest_fitness;

%Armazena o numero de passos do algoritmo na execucao t
numEpocas(t) = iter;

% Calcula os valores das funcoes de base radial (ativacao) para cada
    amostra de treinamento  (utilizando os novos desvios)
f1 = exp(-((x1-c1a).*(x1-c1a)/(2*d(1)^2)+(x2-c2a).*(x2-c2a)/(2*d(2)^2)));
f2 = exp(-((x1-c1b).*(x1-c1b)/(2*d(3)^2)+(x2-c2b).*(x2-c2b)/(2*d(4)^2)));
f3 = exp(-((x1-c1c).*(x1-c1c)/(2*d(5)^2)+(x2-c2c).*(x2-c2c)/(2*d(6)^2)));
f4 = exp(-((x1-c1d).*(x1-c1d)/(2*d(7)^2)+(x2-c2d).*(x2-c2d)/(2*d(8)^2)));
f5 = exp(-((x1-c1e).*(x1-c1e)/(2*d(9)^2)+(x2-c2e).*(x2-c2e)/(2*d(10)^2)));
f6 = exp(-((x1-c1f).*(x1-c1f)/(2*d(11)^2)+(x2-c2f).*(x2-c2f)/(2*d(12)^2)));

[zL,zC]=size(y);

% Regressores
Fi=[f1 f2 f3 f4 f5 f6 ones(1,zL)'];Y=y;

% Calculo dos novos pesos da camada de saida
W=pinv(G)*Y;
w1=W(1); w2=W(2); w3=W(3); w4=W(4); w5=W(5); w6=W(6); w0=W(7);


% Carrega os dados de teste da rede
load DadosTeste;
DadosFunc = DadosTeste;
x1=DadosFunc(:,1); x2=DadosFunc(:,2); y=DadosFunc(:,3);

% Calcula os valores das funcoes de base radial (ativacao) para cada
    amostra de teste
f1 = exp(-((x1-c1a).*(x1-c1a)/(2*d(1)^2)+(x2-c2a).*(x2-c2a)/(2*d(2)^2)));
f2 = exp(-((x1-c1b).*(x1-c1b)/(2*d(3)^2)+(x2-c2b).*(x2-c2b)/(2*d(4)^2)));
```

```
    f3 = exp(-((x1-c1c).*(x1-c1c)/(2*d(5)^2)+(x2-c2c).*(x2-c2c)/(2*d(6)^2)));
    f4 = exp(-((x1-c1d).*(x1-c1d)/(2*d(7)^2)+(x2-c2d).*(x2-c2d)/(2*d(8)^2)));
    f5 = exp(-((x1-c1e).*(x1-c1e)/(2*d(9)^2)+(x2-c2e).*(x2-c2e)/(2*d(10)^2)));
    f6 = exp(-((x1-c1f).*(x1-c1f)/(2*d(11)^2)+(x2-c2f).*(x2-c2f)/(2*d(12)^2)));

    % Calcula a saida da rede para cada amostra de teste
    ye = w0 + w1*f1 + w2*f2 + w3*f3 +w4*f4 + w5*f5 + w6*f6;

    % Calculo do SEQ sobre o conjunto de teste
    er =y-ye; eq=er.*er;
    seq = 0.5*sum(eq);
    seqTeste(t)=seq;

    end %fim for t=1:10

    %####################Resultado Final####################

    %SEQ atraves do procedimento previo
    seqInicialTreino
    seqInicialTeste

    %SEQ atraves do procedimento via PSO
    seqFinalTreino=mean(seqTreino)
    seqFinalTeste=mean(seqTeste)
    finalPassos = mean(numEpocas)

    %Tempo medio de processamento do procedimento de refinamento de parametros
    tempoGasto = toc/10
```

Listagen A.4: Cálculo do fitness da partícula

```
    function fitness=rbf_fitness(d,DadosFunc,DadosPar)

    x1=DadosFunc(:,1); x2=DadosFunc(:,2); y=DadosFunc(:,3);
    c1a=DadosPar(1); c2a=DadosPar(2); c1b=DadosPar(3); c2b=DadosPar(4);
        c1c=DadosPar(5); c2c=DadosPar(6);
    c1d=DadosPar(7); c2d=DadosPar(8); c1e=DadosPar(9); c2e=DadosPar(10);
        c1f=DadosPar(11); c2f=DadosPar(12);

    % Calcula os valores das funcoes de base radial (ativacao) para cada
        amostra de treinamento
    f1 = exp(-((x1-c1a).*(x1-c1a)/(2*d(1)^2)+(x2-c2a).*(x2-c2a)/(2*d(2)^2)));
    f2 = exp(-((x1-c1b).*(x1-c1b)/(2*d(3)^2)+(x2-c2b).*(x2-c2b)/(2*d(4)^2)));
```

```
f3 = exp(-((x1-c1c).*(x1-c1c)/(2*d(5)^2)+(x2-c2c).*(x2-c2c)/(2*d(6)^2)));
f4 = exp(-((x1-c1d).*(x1-c1d)/(2*d(7)^2)+(x2-c2d).*(x2-c2d)/(2*d(8)^2)));
f5 = exp(-((x1-c1e).*(x1-c1e)/(2*d(9)^2)+(x2-c2e).*(x2-c2e)/(2*d(10)^2)));
f6 = exp(-((x1-c1f).*(x1-c1f)/(2*d(11)^2)+(x2-c2f).*(x2-c2f)/(2*d(12)^2)));

[zL zC]=size(y);

%Regressores
G=[f1 f2 f3 f4 f5 f6 ones(1,zL)'];

% Calcula os pesos da camada de saida da rede
W=pinv(G)*y;

% Calcula a saida da rede para cada amostra de treinamento
ye = G*W;

%Calculo do fitness (SEQ)
er =y-ye; eq=er.*er;
fitness = 0.5*sum(eq);
```