SELEÇÃO DE MODELOS NEURAIS UTILIZANDO EVOLUÇÃO DIFERENCIAL ATRAVÉS DO CONTROLE DE ERRO E NORMA DO VETOR DE PESOS

Honovan P. Rocha, Cristiano L. Castro, Antônio P. Braga

Departamento de Engenharia Eletrônica, Universidade Federal de Minas Gerais – UFMG
Av. Antônio Carlos, 6.627 – Campus UFMG Pampulha 30.161-970, Belo Horizonte, MG, Brasil
E-mails: honovan@gmail.com, crislcastro@gmail.com, apbraga@ufmg.br

Resumo – O algoritmo Evolução Diferencial (*Differential Evolution - DE*) faz parte de uma área emergente e promissora e tem se mostrado uma técnica bem sucedida em muitas aplicações, sendo uma abordagem recomendável à resolução de problemas de otimização não linear com variáveis contínuas. O treinamento de redes neurais artificiais é, em sua essência, um problema de otimização não linear onde os pesos da rede são variáveis continuas, o que torna o DE uma boa alternativa para o treinamento. Neste trabalho é apresentada uma abordagem utilizando o algoritmo Evolução Diferencial para treinamento de redes neurais artificiais do tipo Perceptron de múltiplas camadas (MLP). Nesta abordagem utiliza-se o DE para treinar a rede MLP visando a minimização do erro médio quadrático através da busca estocástica pelos parâmetros ideais da rede. Estes parâmetros consistem dos vetores de pesos da rede e tem magnitude limitada através de valores fixos de norma estabelecidos, sendo esta restrição acoplada à codificação utilizada no algoritmo. Os resultados da abordagem proposta aplicada alguns problemas de regressão mostram resultados promissores, sendo estes contrastados a resultados obtidos por versões clássicas do backpropagation visando apenas minimização do erro.

Palavras-chave – algoritmo Evolução Diferencial, restrição da norma, magnitude dos pesos.

Abstract – The Differential Evolution algorithm (*DE*) is part of an emerging and promising area and has proved a successful technique in many applications, and a recommended approach to solving nonlinear optimization problems with continuous variables. The training of artificial neural networks is, in essence, a nonlinear optimization problem where the weights of the network are continuous variables, which makes a *DE* a good alternative for training. This work presents an approach using the Differential Evolution algorithm for training of artificial neural networks like multilayer perceptron (MLP). This approach uses the *DE* to train the MLP network aiming at minimizing of the mean square error by stochastic search for ideal parameters of the network. These parameters consist of the weight vectors of the network and have magnitude limited by fixed values of the norm, and this restriction is attached to the encoding used in the algorithm. Results of the proposed approach applied to some regression problems show promising results, which are contrasted to results obtained in classic versions of the backpropagation aiming only minimization error.

Keywords – Differential Evolution algorithm, norm restriction, magnitude of weights.

1 Introdução

A seleção de modelos visando à melhoria da capacidade de generalização de Redes Neurais Artificiais (RNAs) tem sido obtida através de técnicas como a utilização de métodos de validação, algoritmos construtivos, algoritmos de poda e regularização. Nas abordagens por validação uma medida de erro sobre um conjunto de validação [1] é usada para selecionar os modelos sem, no entanto, fazer referências explícitas à complexidade dos mesmos. Algoritmos de poda ou construtivos [2] são baseados na manipulação da estrutura (ou topologia) da rede que, na prática, resultam em modelos menos complexos apresentando um número menor de parâmetros. Aplicações de técnicas de regularização, como o conhecido método *Weight Decay* (WD) [3], são geralmente baseadas na idéia de controlar a complexidade dos modelos a partir de restrições na magnitude dos pesos da rede. Nessa mesma linha, novos algoritmos de aprendizado baseados em técnicas de otimização multiobjetivo (MOBJ) têm sido aplicados ao aprendizado de RNAs [4, 5, 6]. Estes métodos visam a seleção modelos através da minimização de dois objetivos conflitantes: o erro de treinamento e a norma do vetor de pesos. Após o processo de otimização, um conjunto de soluções de eficientes (não dominadas) constituindo uma estimativa do conjunto Pareto-Ótimo [7] é obtida. Dentre as soluções obtidas, aquela que apresenta complexidade compatível com a tarefa de aprendizagem em questão deve ser selecionada. Os algoritmos MOBJ, no entanto, utilizam métodos de otimização baseados na derivada primeira ou segunda do vetor de objetivos, o que pode dificultar a obtenção de soluções globais. Algoritmos Evolucionários e, em particular, o DE, apresentam-se como uma alternativa para o treinamento multi-objetivo de RNAs com maior eficiência na busca por um mínimo global devido às características da busca populacional, além da utilização de uma também eficiente busca local através da operação mutação diferencial.

Tendo em vista as considerações acima, objetiva-se neste trabalho a utilização do algoritmo DE para o treinamento de uma RNA do tipo Perceptron de Múltiplas Camadas (MLP) [8]. A abordagem proposta utiliza a técnica de otimização multiobjetivo ϵ-restrito (P_ϵ). Nesta técnica, como pode ser visto em [7], transforma-se o problema de otimização multiobjetivo em subproblemas mono-objetivo restritos, tomando como restrição a norma do vetor de pesos da rede e minimizando o erro médio quadrático. Desta maneira pode-se utilizar qualquer algoritmo de otimização não linear com restrições para resolução do problema. Na abordagem aqui utilizada aproveita-se o poder da busca estocástica inerente aos algoritmos evolucionários e

acopla-se a restrição na codificação das variáveis, onde a amplitude dos pesos é limitada por valores fixos para a norma, visando assim à melhoria da capacidade de generalização da rede. Esta restrição é representada através dos valores mínimos e máximos definidos para as variáveis de cada solução proposta pelo algoritmo. Os resultados obtidos através do DE mostram que esta abordagem apresenta-se como promissora para o treinamento MOBJ de RNAs.

O trabalho está organizado da seguinte forma: a seção 2 apresenta alguns conceitos relativos ao treinamento de redes MLP, na seção 3 é descrito o algoritmo Evolução Diferencial, na seção 4 é definido o algoritmo proposto, a seção 5 é explica a metodologia utilizada, na seção 6 são expostas algumas simulações e resultados obtidos e, por fim, a seção 7 expressa as conclusões do trabalho.

2 Treinamento de Redes MLP

Em [9] mostra-se que diversos fatores influenciam o treinamento de redes MLP, tais como, as definições da quantidade de neurônios nas camadas ocultas, taxa de aprendizagem, heurísticas para inicialização dos pesos, etc. Ao fim do treinamento busca-se um modelo que apresenta boa aproximação para os dados de treinamento e também, boa capacidade de generalização. O ajuste da complexidade do modelo junto à minimização do erro busca encontrar um modelo que não seja demasiadamente rígido a ponto de não modelar os dados e nem flexível demais a ponto de modelar o ruído. Este ajuste entre o erro de treinamento e a complexidade do modelo é conhecido como dilema entre o viés e a variância [10]. Considerando-se uma rede MLP, conhecendo-se a relação entre a capacidade de aprendizagem e o tamanho dos pesos da rede [11], pode-se definir a norma dos pesos da rede $\|w\|$ como uma medida de controle da variância. Desta maneira pode-se definir um modelo que controle erro e a complexidade em termos de um modelo de otimização mono-objetivo com restrições, descrito como:

$$\min \; J(w) = \frac{1}{N}\sum_{i=1}^{N}\left(d_i - y_i\right)$$
$$s.a\ \|w\| \leq \varepsilon \qquad (1)$$

onde d_i e y_i são os valores para a saída desejada e a saída da rede para i-ésima amostra, respectivamente, sendo que, *J(w)* é o erro quadrático médio e μ é um valor arbitrário que limita a norma dos pesos da rede.

A partir desta definição pode-se treinar uma rede MLP buscando maior capacidade de generalização através da minimização do erro médio quadrático e da norma euclidiana dos pesos, com a utilização do método P_ϵ, consistindo em variar os valores de μ de forma a encontrar diversas soluções de mínimo erro estimado. A faixa de valores de ε é definida de forma uniforme dentro dos limites de soluções extremas, que são aquelas onde se minimiza cada objetivo sem considerar o outro, obtendo-se um valor mínimo de ε como sendo a norma mínima da MLP sem considerar o erro e o valor máximo de ε sendo o valor da norma considerando-se apenas a minimização do erro. Em ambos os casos os valores de ε foram arredondados para um valor inteiro próximo.

3 Algoritmo Evolução Diferencial

O algoritmo evolução diferencial se enquadra na classe de algoritmos evolutivos, mesmo não possuindo uma inspiração em sistemas naturais. A mutação diferencial, operação que define o algoritmo, possuí fundamento baseado em princípios matemáticos e heurísticos, mas por utilizar uma população de soluções que evoluem durante as iterações do algoritmo, além de operadores heurísticos relacionados a mecanismos gerais de adaptação natural, este possui características encontradas nos algoritmos evolutivos de forma geral.

Proposto em [12] o algoritmo Evolução Diferencial foi desenvolvido para resolver problemas de otimização, mas no intuito ser uma técnica com rápida convergência e de fácil utilização, utilizando poucos parâmetros a serem definidos pelo usuário. Trata-se num algoritmo simples e eficiente que têm se destacado no âmbito da otimização não linear com variáveis continuas. A ideia fundamental que sustenta o algoritmo é o esquema de mutação diferencial onde se gera um "vetor tentativa" (ou vetor diferencial), obtido através da diferença vetorial ponderada entre dois indivíduos da população, adicionada a um terceiro indivíduo. Este vetor diferencial gerado é comparado a um indivíduo pré-determinado da população e substitui este caso possua um valor mais adequado à função objetivo, dada a tarefa de otimização em questão, se minimização ou maximização.

É mostrado em [13] que durante o processo de busca do algoritmo os tamanhos e orientações dos vetores diferenciais são modificados de acordo com a paisagem da função o objetivo de forma a se ajustarem a ela, conferindo desta forma uma característica de auto adaptação ao DE. Com esta característica a mutação diferencial provê ao algoritmo maior robustez, versatilidade e eficiência para a resolução de diversos problemas.

A estrutura básica do DE é semelhante à maioria dos algoritmos evolutivos, como pode ser visto com detalhes em [14], consistindo dos passos a seguir na implementação utilizada:

1 – Inicialização da população inicial de forma aleatória dentro dos limites dos parâmetros.

2 – Avaliação dos indivíduos da população quanto sua adequação à função objetivo.

3 – Para cada indivíduo w_i da população efetua-se o cruzamento do individuo com o vetor diferencial (v) gerado com a operação mutação diferencial, obtendo-se um indivíduo u_i.

4 – Seleção da população para a nova iteração mantendo o indivíduo mais adequado à função objetivo tomando-se a comparação entre u_i e w_i na iteração atual.

Esta sequência de passos se repete a partir do passo 2 até que se alcance um critério de convergência para o algoritmo.

A inicialização da população inicial ocorre após definir-se os valores de limite inferior (w_{min}) e superior (w_{max}) para os parâmetros utilizados, em seguida, gera-se indivíduos distribuídos uniformemente dentro desta faixa determinada, sendo definido por:

$$w_i = w_{min} + rand * (w_{max} - w_{min}) \quad (2)$$

onde *rand* é um valor aleatório no intervalo [0,1] obtido a partir de uma distribuição uniforme. Todas as variáveis de um indivíduo w_i possuem seus valores dentro da mesma faixa.

Após definir-se a população inicial percorre-se esta passando por cada indivíduo efetuando-se uma operação de cruzamento (ou recombinação discreta) do individuo atual no processo com um vetor diferencial gerado pelo processo de mutação diferencial. Este vetor é obtido, em uma versão original do DE, por:

$$v = w_{r1} + \eta(w_{r2} - w_{r3}) \quad (3)$$

onde w_{r1}, w_{r2} e w_{r3} são vetores obtidos aleatoriamente da população, sendo que estes são obtidos de forma mutuamente excludentes. w_{r1} é denominado vetor base e η é um fator de escala aplicado ao vetor diferença.

A operação de recombinação discreta é definida por:

$$\mu_i^j = \begin{cases} v^j, se\ rand \leq C\ ou\ j=\delta \\ w_i^j, caso\ contrário \end{cases} \quad (4)$$

onde $j = 1,...,n$ é o índice de uma determinada variável do individuo, C é a probabilidade de cruzamento, definido no intervalo [0,1] e, $\delta \in \{1,...,n\}$ é um índice aleatório definido para que pelo menos uma variável do indivíduo herde características da solução mutante.

4 A Implementação do DE utilizada

A abordagem proposta utiliza o algoritmo DE para modificação dos pesos da rede de forma a minimizar erro e norma, semelhante à forma que ocorre em [4], substituindo-se o algoritmo de otimização determinístico utilizado. Utilizou-se o erro médio quadrático como medida de erro para efeito de comparação com outros métodos geralmente utilizados na literatura. A medida de fitness utilizada para avaliação das soluções encontradas é o valor do erro médio quadrático encontrado após submeter-se todos os padrões de treinamento à rede.

Visando a melhoria na capacidade de generalização, utiliza-se valores fixos para a norma dos pesos para definição da amplitude destes. Sendo que os valores mínimos e máximos para cada peso são definidos como:

$$w_{min} = -\frac{\varepsilon}{\sqrt{nvar}} \quad (5)$$

$$w_{max} = +\frac{\varepsilon}{\sqrt{nvar}} \quad (6)$$

onde *nvar* é a quantidade de parâmetros livres na rede.

A operação de mutação diferencial utilizada nesta implementação é diferente daquela vista na estrutura básica do DE descrita na seção anterior, sendo que esta forma também pode ser vista em [12], sendo definida por:

$$v = w_i + \lambda(w_{best} - w_i) + \eta(w_{r2} - w_{r3}) \quad (7)$$

Onde w_{best} é o indivíduo com maior valor de fitness na iteração atual e λ é um fator de escala utilizado para o novo elemento gerado em substituição ao vetor base definido na seção anterior.

Os indivíduos utilizados no algoritmo, só são considerados caso tenham respeitem os valores limite das variáveis durante as operações de mutação diferencial e cruzamento.

O critério de parada utilizado pelo algoritmo é o número de gerações.

5 Metodologia Utilizada

Para o treinamento da MLP foram atribuídos os seguintes valores aos parâmetros do algoritmo DE: Np=60, Ngen = 30, $\lambda = 0.99$, $\eta = 0.99$, $C = 0.9$ onde Np é a quantidade indivíduos na população e Ngen é quantidade de iterações do algoritmo. Os parâmetros λ, η e C foram definidos empiricamente após alguns testes.

Após a definição dos parâmetros necessários ao DE, foram efetuados treinamentos da MLP com valores para de ε variando entre 2 e 16, sendo estes, valores encontrados com a obtenção das soluções extremas. Com variação de 0.5, obteve-se um total de 34 soluções. Dentre as soluções obtidas a partir do treinamento foi escolhida como melhor solução aquela com menor erro de validação.

Para se ter uma base de comparação foram utilizados problemas de regressão, onde avaliou-se o desempenho de duas versões do back-propagation em relação à abordagem proposta. Os critérios utilizados para comparação foram o erro médio quadrático (EMQ) para o conjunto de validação e o tempo médio de execução (TME), definido em segundos. Em todos os métodos utilizados as camadas escondidas e de saída da rede utilizaram função de ativação do tipo tangente hiperbólica e linear, respectivamente, sendo que a arquitetura da MLP foi definida como 1-15-1. Com a arquitetura definida nestes termos sabe-se da ocorrência do problema de over-fitting utilizando apenas minimização do erro para solução do problema aqui utilizado, como descrito em [5].

6 Simulações e Resultados

Nas simulações foram utilizados quatro problemas de regressão tipicamente encontrados na literatura para este propósito. As funções que descrevem cada problema são:

$$f1(x) = seno(x) \tag{8}$$

$$f2(x) = 4.26(e^{-x} - 4e^{-2x} + 3e^{-3x}) \tag{9}$$

$$f3(x) = (x-2)(2x+1)/(1+x^2) \tag{10}$$

$$f4(x) = (e^{-0.2x}) + (2e^{-0.2x} * seno(2\pi * 0.2x - \pi/4) - 0.27) \tag{11}$$

Os conjuntos de treinamento consistem de 100 observações limitados nos intervalos [0;2π], [0;3.25], [-8;12] e [0, 10] para as funções *f*1, *f*2, *f*3 e *f*4 respectivamente. As observações foram obtidas através das funções geradoras com o acréscimo de um ruído gaussiano com média 0 e desvio padrão 0.2.

É possível observar através das figuras 1, 2, 3 e 4 as curvas geradas pelos algoritmos *Levemberg-Marquardt* (LM), RPROP (em inglês, *Resilient Back-propagation*) e pela solução utilizando o DE.

Observa-se pelas figuras que o LM e RPROP ajustam-se muito ao ruído enquanto a solução utilizando o DE aproxima-se mais da função geradora.

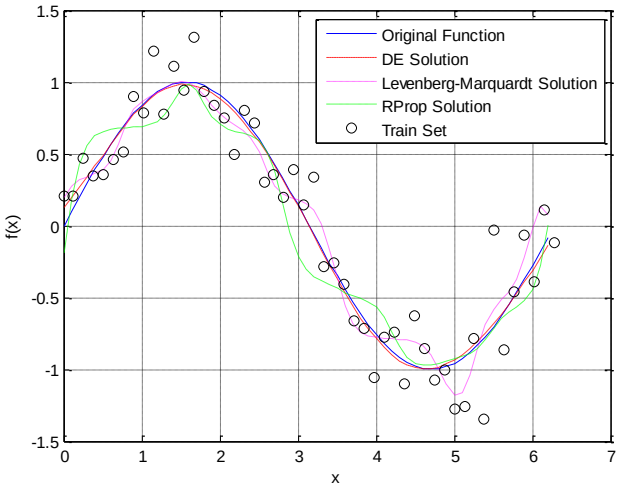

Figura 1 – Curvas geradas pelos algoritmos para a função *f*1(*x*).

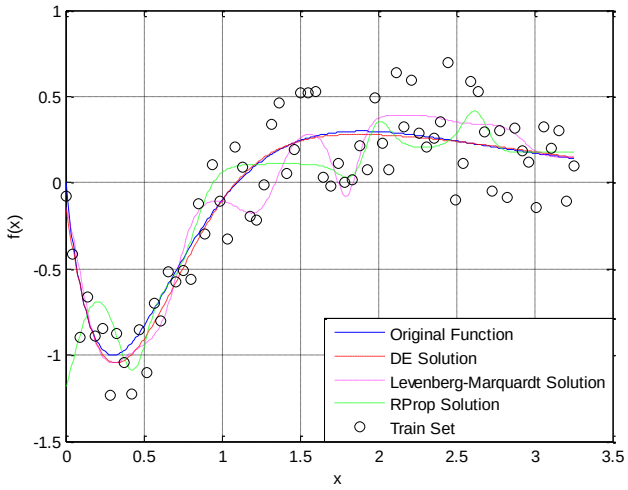

Figura 2 – Curvas geradas pelos algoritmos para a função $f2(x)$.

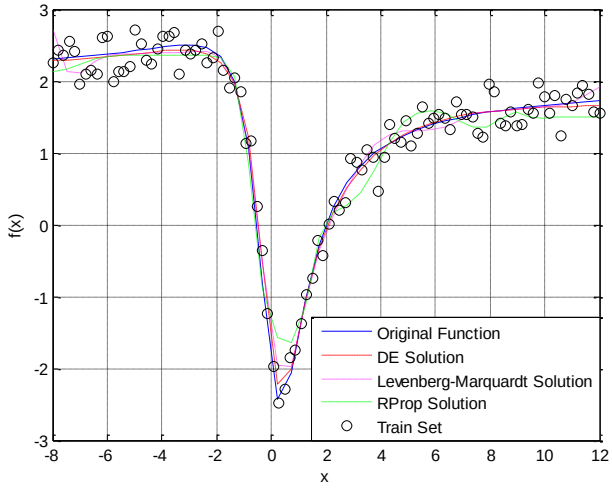

Figura 3 – Curvas geradas pelos algoritmos para a função $f3(x)$.

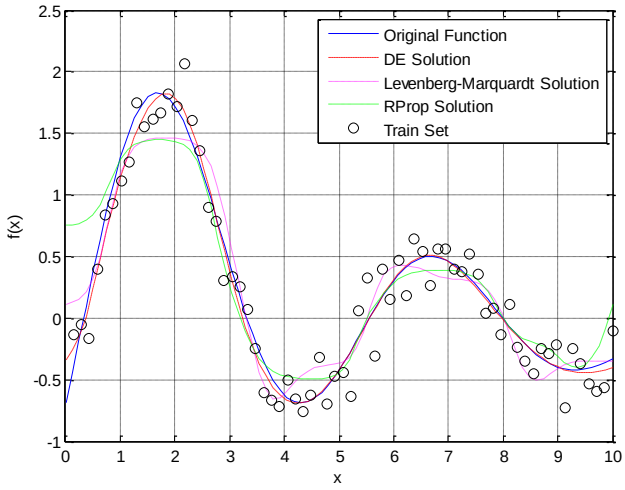

Figura 4 – Curvas geradas pelos algoritmos para a função $f4(x)$.

Através das tabelas 1, 2, 3 e 4 podem ser observados os valores de EQM e TME obtidos para cada algoritmo avaliado para as quatro funções. Pode-se observar que o DE supera os outros dois algoritmos em relação ao EQM e é superado por estes em relação ao TME.

Tabela 1: Relação dos valores de erro médio quadrático (EQM) e tempo médio de execução (TME) para a função $f1(x)$.

	DE	**LM**	**RPROP**
EQM	0.0065	0.4960	0.5317
TME	2.2932	1.7628	0.5148

Tabela 2: Relação dos valores de erro médio quadrático (EQM) e tempo médio de execução (TME) para a função $f2(x)$.

	DE	**LM**	**RPROP**
EQM	0.0018	0.7227	1.0176
TME	2.3180	1.7316	0.6240

Tabela 3: Relação dos valores de erro médio quadrático (EQM) e tempo médio de execução (TME) para a função $f3(x)$.

	DE	**LM**	**RPROP**
EQM	0.0014	1.2765	1.1723
TME	2.4492	2.3684	0.9828

Tabela 4: Relação dos valores de erro médio quadrático (EQM) e tempo médio de execução (TME) para a função $f4(x)$.

	DE	**LM**	**RPROP**
EQM	0.0059	0.5591	0.5495
TME	2.3953	1.5444	0.6708

Como visto nas tabelas o tempo de execução do algoritmo DE foi mais alto do que nos outros métodos, mas verifica-se que esta diferença é relativamente pequena dado que observa-se também que o DE obteve uma boa capacidade de generalização, onde o erro médio quadrático para os dados de validação foi inferior aos erros obtidos nos outros dois métodos. Outras versões do DE, com variações das operações realizadas, podem conduzir a melhores resultados no ponto de vista do tempo computacional, visto que utilizou-se neste trabalho uma versão encontrada no trabalho original onde define-se o algoritmo básico.

7 Conclusões

Neste trabalho adotou-se uma abordagem que utiliza o algoritmo Evolução Diferencial para treinamento de redes MLP visando um treinamento multiobjetivo que busca uma relação de compromisso entre complexidade e erro do modelo. A transformação do problema multiobjetivo em múltiplos problemas mono-objetivo restritos através do método P_ϵ possibilitou a utilização da restrição de forma acoplada à codificação das variáveis utilizadas no algoritmo, facilitando a implementação e utilização do mesmo, de forma que a restrição não precisa ser explicitamente tratada, pois isso é obtido assegurando-se que as variáveis de otimização estejam dentro dos limites definidos. A abordagem proposta, em relação ao custo computacional, teve um desempenho inferior aos algoritmos utilizados para comparação, sendo que esta diferença foi relativamente pequena e, quase inexistente em um dos problemas de teste. No entanto, a qualidade das aproximações mostra que a abordagem é promissora, de forma que o alto tempo de processamento é uma característica inerente aos métodos evolucionários, sendo estes, por outro lado, mais robustos na geração de soluções globais como pode ser visualizado nos resultados que mostraram maior eficiência desta abordagem no quesito erro médio quadrático para os dados de validação. Desta maneira sugere-se que pesquisas mais extensivas sejam realizadas na utilização desta abordagem. Variações do DE utilizado podem conduzir a melhores resultados quanto ao tempo computacional.

Para trabalhos futuros, sugere-se também a mudança na forma de tratamento do problema multiobjetivo, utilizando outras formas de se transformar o problema ou a utilização de um DE multiobjetivo baseado em pareto-dominância para obtenção das soluções.

10 Referencias

[1] M. Stone, Cross-validatory choice and assessment of statistical predictions, **Journal of the Royal Statistical Society B**, 36(1974), 111–147.

[2] R. Reed, Pruning algorithms - a survey, **IEEE Transactions on Neural Networks**, 4(1993), 740–746

[3] G. Hinton, Connections learning procedures, **Artificial Intelligence**, 40(1989), 185–234

[4] R. A. Teixeira, A. P. Braga, R. H. C. Takahashi, R. R. Saldanha, Improving Generalization of MLPs with Multi-Objective Optimization, **Neurocomputing** 35(2001), 189 – 194.

[5] M. A. Costa, A. P. Braga, B. R. Menezes, R A. Teixeira, G. G. Parma, Training Neural Networks with a Multi-Objective Sliding Mode Control Algorithm, **Neurocomputing**, 51(2002), 467 – 473.

[6] I. Kokshenev, A. P. Braga, A multi-objective approach to RBF network learning, **Neurocomputing** 71(2008), 1203–1209

[7] E. G. Horta, A. P. Braga, R. R. Saldanha, Acelerando o Treinamento Multiobjetivo de RNAs pelo Método de Gradiente Projetado, **Congresso Brasileiro de Automática**, (2008).

[8] S. Haykin, Redes Neurais: principios e prática, 2ª Edição, **Bookman**, (2001).

[9] A .P. Braga, T. B. Ludemir, A. C. P. F. Carvalho, Redes neurais artificiais – teoria e aplicações. 2ª Edição, **LTC Livros Técnicos e Científicos Editora S. A**, (2007).

[10] S. Geman, E. Bienenstock, R. Doursat, Neural networks and the bias/variance dilemma, **Neural Computation**, 4(1992), 1–58.

[11] P. L. Bartlett, For valid generalization the size of the weights is more important than the size of the network, **Advances in Neural Information Processing Systems**, 9(1997), 134 - 140.

[12] R. M. Storn, K. V. Price, Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces, **Technical Report TR-95-012, International Computer Science Institute**, 8(1995), 22.

[13] F. G. Guimarães, Algoritmos de Evolução Diferencial para Otimização e Aprendizado de Máquina, **IX Congresso Brasileiro de Redes Neurais**, (2009).

[14] A. S. M. Lacerda, Proposta de um Algoritmo Evolucionário Nebuloso para Solução de Problemas de Otimização Multiobjetivo, **Dissertação de mestrado apresentada ao Curso de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais - UFMG**, (2010).