MINIMIZAÇÃO DO ERRO NO ALGORITMO *BACK-PROPAGATION* APLICADO AO PROBLEMA DE MANUTENÇÃO DE MOTORES

ROSEMARY ANTONIA LOPES FARACO
PYRAMO PIRES DA COSTA JR.
Programa de Pós-graduação em Engenharia Elétrica
Pontifícia Universidade Católica de Minas Gerais

FREDERICO RODRIGUES BORGES DA CRUZ
Departamento de Estatística
Universidade Federal de Minas Gerais

Resumo: Este artigo trata do problema de manutenção preventiva de motores de indução com rotor em gaiola, considerados importantes elementos para os processos produtivos. As habilidades das redes neurais em classificar e reconhecer padrões são algumas das características que as tornam uma ferramenta útil em tais aplicações. Apresentam-se aqui uma rede neural aplicável ao problema e um estudo sobre o conhecido algoritmo *back-propagation*, onde são analisadas técnicas para redução, na fase de treinamento, das oscilações no processo de convergência e do tempo de processamento dispendido. Resultados computacionais parecem indicar que o uso de tais técnicas é efetivo e promissor, nesta área de aplicação.

Palavras-chave: Manutenção Preventiva de Motores; Redes Neurais; Algoritmo *Back-propagation*.

Abstract: The aim of this paper is to treat the preventive maintenance problem in squirrel-cage induction motors. Neural networks are applied to the problem taking advantage of their well-known classification and pattern recognition abilities. The focus of the study is on the learning process, which is made by means of the back-propagation algorithm, discussing techniques that may improve its convergence, reducing oscillations and the CPU time required. Computational results seem to indicate that these techniques are quite effective and promising in this application field.

Keywords: Preventive Motor Maintenance; Neural Networks; Back-propagation Algorithm;.

1. Introdução

O contexto de aplicação das redes neurais a ser explorado neste trabalho é o de manutenção de máquinas elétricas do tipo motores de indução, importantes elementos para os processos produtivos, cuja parada não prevista, por defeitos elétricos ou mecânicos, representa elevados prejuízos financeiros, seja pela paralisação da produção, seja pelo preço da manutenção ou por ambos. Quando as falhas elétricas são detectadas em seus estágios iniciais, pode-se programar a manutenção, reduzindo-se custos. Quando passam despercebidas, podem evoluir rapidamente para uma falha geral da máquina. A habilidade em reconhecer rapidamente padrões é a característica das redes neurais que as torna uma ferramenta útil para tais aplicações práticas.
Neste artigo, tomam-se como base os trabalhos de Chow *et al*. (1990,1991), que tratam do estudo de basicamente dois tipos de faltas incipientes em um motor de indução com rotor do tipo gaiola de esquilo, quais sejam, (i) faltas no enrolamento estatórico e (ii) deterioração dos rolamentos rotóricos.
As considerações básicas para a análise são enumeradas a seguir:

- curto-circuitos causam deterioração do isolamento dos enrolamentos estatóricos devido à elevação prolongada de temperatura acima do que a sua classe de isolamento permite. A degradação do isolamento também pode ser devida à idade ou ao uso fora da tensão, corrente ou freqüência nominais. O envelhecimento do isolamento devido ao excesso de temperatura caracteriza uma falta incipiente e causará uma mudança no número de Ampères-espira do enrolamento estatórico;
- rolamentos estão sujeitos a deterioração por lubrificação contaminada ou excessiva, aplicações inadequadas ou desalinhamentos. O desgaste dos mancais reflete-se no coeficiente de atrito do motor.

Relacionar a velocidade do motor (W) e sua corrente elétrica (I), em regime permanente, ao coeficiente de atrito (Bc) e ao número de espiras equivalente do estator (Nc) é uma tarefa complexa, devido às não-linearidades do sistema. Em Chow *et al.* (1991) encontramos o seguinte equacionamento não-linear:

$$g\,(I, W, Nc\,, Bc) = 0 \qquad (1)$$

onde $g = [\, g_1, g_2\,]^{T}$. Para maiores detalhes, ver o artigo em referência.
No entanto, o desenvolvimento de um modelo matemático pode ser substituído pelo uso de uma rede neural que tenha "aprendido" o mapeamento entre entradas e saídas, sendo capaz de identificar as situações para as quais tenha sido treinada e sinalizando com a identificação do defeito ou

condição normal correspondente. A rede, uma vez treinada, representa as não-linearidades do mapeamento desejado.
Serão apresentados, na Seção 2, alguns conceitos básicos sobre redes neurais e sobre o algoritmo de treinamento a ser utilizado neste trabalho, o *back-propagation*. Na Seção 3, descrever-se-á a rede neural desenvolvida. Na Seção 4, serão discutidos alguns problemas que podem ser encontrados na fase de treinamento e apresentadas técnicas para aceleração do processo. Na Seção 5, serão apresentados resultados computacionais, obtidos usando-se os dados reais levantados por Chow *et al*. (1991). Finalmente, a Seção 6 é dedica às conclusões e observações finais.

2. A Rede Neural como Ferramenta da Inteligência Artificial

2.1. O que é uma Rede Neural ?

Conforme citado por Kartalopoulos (1996), Warren McCulloch e Walter Pitts publicaram, em 1943, a referência zero sobre a teoria de redes neurais artificiais e foram os primeiros a propor um modelo computacional para o neurônio biológico. Apesar de simples, o neurônio por eles modelado constituía-se em um dispositivo binário capaz de implementar algumas funções lógicas mais simples, como o "E" e o "OU". A rede não possuía qualquer mecanismo de comparação entre entrada e saída ou qualquer processo de ajuste de pesos.
A Figura 1 apresenta o modelo do neurônio artificial (elemento neural) de McCulloch e Pitts, o qual é composto por um conjunto de entradas v_1, v_2,…,v_n, uma unidade de processamento Σ_i, uma função de ativação não linear f_i e uma saída O_i. Cada entrada *j* recebe um estímulo v_j que é ponderado pelo peso w_{ij}, que pode ser excitatório ou inibitório. A unidade de processamento coleta os sinais ponderados e os agrega juntamente com o termo *bias*, ou limiar, θ_i. A soma de todas as entradas ponderadas e o limiar é então modulada pela função de ativação f_i que determina o nível de excitação gerado pelo elemento neural. Os neurônios biológicos respondem de forma similar, porém as funções de ativação utilizadas nos elementos neurais não são réplicas das biológicas e são utilizadas de acordo com o tipo de aplicação.

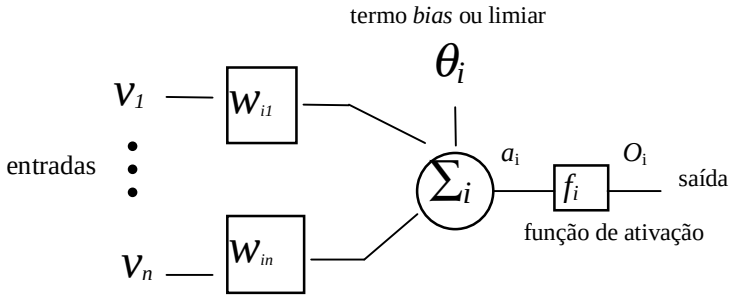
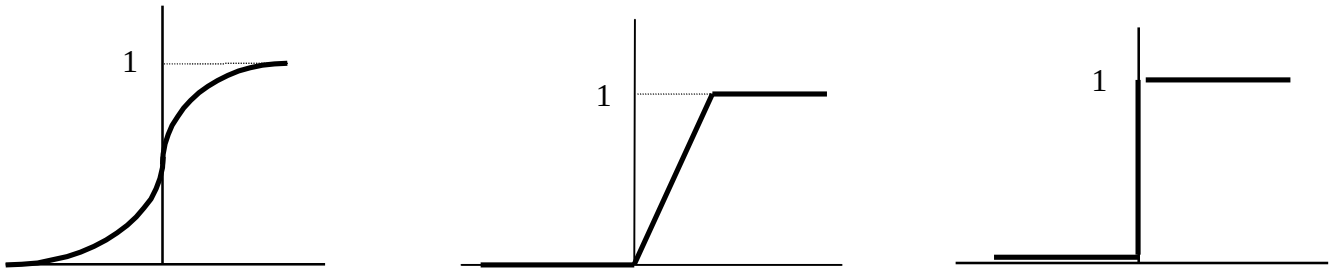


Figura 1 - Unidade Neural Básica

As funções de ativação mais populares são a sigmoidal, a rampa e a degrau, mostradas na Figura 2.



a) sigmoidal b) rampa c) degrau

Figura 2 - Funções de Ativação Típicas

A saída do modelo básico, mostrado na Figura 1, pode ser descrita pela seguinte relação:

$$O_i = f_i(a_i), \tag{2}$$

onde

$$a_i = \sum_j w_{ij} v_j - \theta_i. \tag{3}$$

Dentre a funções citadas, a sigmoidal, por ser uma função monotônica, limitada e ter uma derivada fácil de calcular é mais amplamente utilizada em aplicações que envolvem a utilização de redes neurais artificiais. Por exemplo, utilizando-se a função de ativação sigmoidal, a saída do modelo básico mostrado na Figura 1 será dada por:

$$O_i = f(a_i) = \frac{1}{1+e^{-a_i}}, \tag{4}$$

com uma derivada simples:

$$f'(a_i) = \frac{e^{-a_i}}{\left(1+e^{-a_i}\right)^2}. \tag{5}$$

De acordo com Kovács (1996), Rosenblatt cria, no final da década de 50, dando prosseguimento às idéias de McCulloch e Pitts, uma rede com múltiplos elementos neurais dispostos em camadas, a qual chama *perceptron*. Este se tornou um sistema de classificação capaz de reconhecer padrões geométricos e abstratos. Assim surgiram as redes neurais artificiais, sistemas compostos por conjuntos de elementos neurais interligados. Elas simulam o comportamento de uma rede de neurônios biológicos, através de modelos computacionais que realizam a generalização dos sinais de entrada a partir de um modelo de classificação obtido durante a fase chamada de treinamento. Atualmente, as redes neurais têm sido largamente utilizadas em diferentes áreas do conhecimento humano como detecção de faltas, diagnósticos médicos, modelamento de sistemas dinâmicos, controle de robôs e muitas outras.

Uma rede neural artificial possui, como componentes básicos, (i) um conjunto de elementos neurais, (ii) uma arquitetura que define o número de elementos e a forma com se interconectam, e (iii) propriedades funcionais que definem como a rede aprende, recorda, associa, compara e classifica. A

Figura 3 ilustra a topologia de uma rede neural artificial multicamada. Os elementos que recebem diretamente as entradas da rede constituem o que se chama de camada de entrada. Os elementos que recebem como entradas, as saídas daqueles da camada de entrada, constituem a segunda camada e assim sucessivamente até a camada final, denominada camada de saída. As camadas intermediárias (internas) são geralmente referidas como camadas ocultas. Apesar de não representado, cada elemento neural da Figura 3 possui aquela estrutura apresentada na Figura 1.

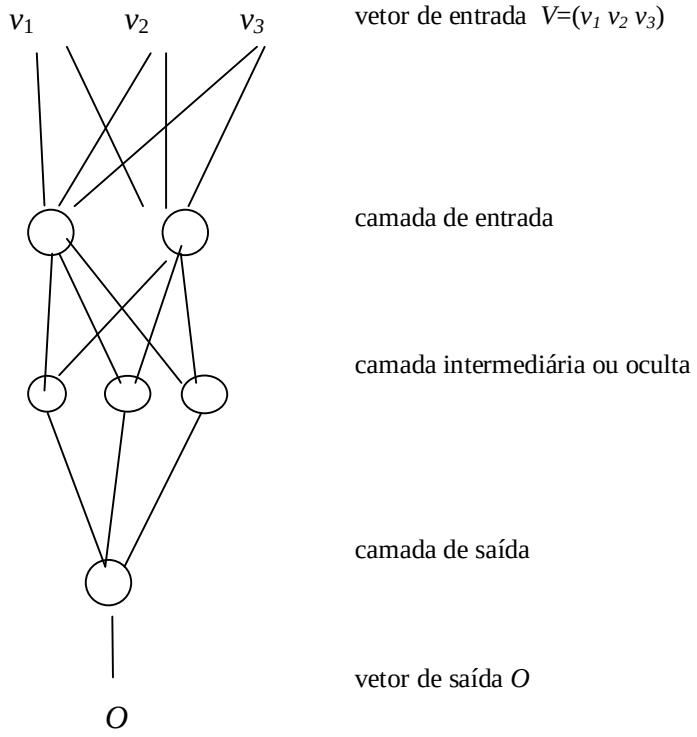


Figura 3 - Estrutura de uma Rede Neural Artificial Multicamada

As redes neurais artificiais podem ter topologias diversas incluindo uma, duas ou várias camadas. De acordo com Chow *et al.* (1991), uma topologia conveniente ao problema de manutenção de motores tratado neste trabalho é a chamada *feedforward* (propaga o sinal para a frente). Caracteriza-se por ser constituída de unidades altamente interconectadas, com uma ou mais camadas ocultas. Os elementos são interconectados por pesos. Cada unidade deve enviar sua saída para camadas de ordem maior que a sua e devem receber suas entradas de camadas abaixo da sua. Não existem interconexões entre os elementos dentro de uma mesma camada (Tsoukalas & Uhrig, 1997) e (Penman & Yin, 1992). Esta será a topologia utilizada neste trabalho.

2.2. Treinamento da Rede Neural

Sistemas biológicos possuem a propriedade de **aprender** uma função, sendo que um dos métodos de treinamento mais intuitivos é aquele feito através de exemplos. Este método também pode ser aplicado à rede artificial, bastando que padrões de comportamento, compostos por estímulos de entrada e saídas esperadas, sejam apresentados à rede, através de um algoritmo de **treinamento**. Durante o processo de treinamento, a rede ajusta os pesos entre elementos e aqueles que ponderam as entradas unitárias dos limiares. Este processo é feito continuamente, em resposta aos estímulos de entrada, o que é conhecido no contexto de redes neurais como **aprendizagem** (Haykin, 1994). Quando a saída estiver dentro dos limites estabelecidos como toleráveis, a fase de treinamento estará terminada, tendo a rede adquirido **conhecimento**.

Existem diferentes processos de aprendizado e nem todos possuem o mesmo grau de eficiência. Em redes neurais artificiais, é necessário escolher dentre as diferentes técnicas de aprendizado disponíveis, aquela que melhor se adapte às características de cada rede e à natureza do problema em consideração.
As duas filosofias de aprendizagem são:

- **aprendizagem supervisionada**, processo realizado mediante sucessivas apresentações de padrões de entrada-saída conhecidos, requerendo um "professor" ou "supervisor", responsável pelo processo de minimização da diferença entre resposta da rede e a resposta esperada, determina ainda os ajustes que devem ser feitos para que o erro seja minimizado;

- **aprendizagem não supervisionada**, que não requer um "professor", nem existe uma saída esperada. Requer que sejam definidas diretrizes para a formação de agrupamentos de características que constituirão diferentes classes. Durante a sessão de treinamento a rede recebe em suas entradas, diferentes padrões que são organizados segundo as categorias definidas. Quando, depois de treinada, a rede recebe um estímulo, é gerada uma resposta indicando a classe a qual tal estímulo pertence. Se a rede não for capaz de encontrar a classe para o estímulo recebido, uma nova classe será gerada, conforme as diretrizes para seleção de características de agrupamentos, incluídas no projeto da rede.

Há diversos algoritmos de treinamento disponíveis e o método e a velocidade com que os pesos são ajustados dependem do tipo utilizado. Os algoritmos de treinamento podem ser vistos como algoritmos de otimização que promovem a minimização da função erro, definida em relação aos valores esperados na saída, obedecendo aos limites de tolerância pré-estabelecidos. Dentre as diferentes técnicas de otimização aplicáveis, podem ser citados: a regra delta, o algoritmo de Boltzmann, o algoritmo *back-propagation*, o *simulated annealing*, algoritmos baseados em cadeias de Markov. Maiores detalhes podem ser encontrados em Kartalopoulos (1996).

Neste trabalho, será utilizado o algoritmo de treinamento *back-propagation*, o qual é, conceitualmente, uma generalização do algoritmo de aprendizado de Widrow e Hoff - "*Least Mean Square*" (LMS), também conhecido como Regra Delta. Ver Kosco (1992), Kovács (1996) e Kartalopoulos (1996). Neste algoritmo que enquadra-se na categoria de aprendizagem supervisionada, o desempenho da rede é medido por uma função erro que considera, para cada um dos diversos padrões p de entrada, o quadrado da diferença entre o valor esperado e a respectiva saída calculada. Em outras palavras, o erro é o somatório dos erros quadráticos, definido pela seguinte expressão:

$$E = \sum_{i,p} E_i^p = \frac{1}{2} \sum_{i,p} (d_i^p - O_i^p)^2, \tag{6}$$

onde E_i^p representa o erro no *i*-ésimo elemento neural, para o *p*-ésimo padrão de entrada, d_i^p é a saída esperada no *i*-ésimo elemento neural, para o *p*-ésimo padrão de entrada e O_i^p é a saída produzida, definida como:

$$O_i^p = f(a_i^p) = f\left(\sum_j w_{ij} v_j^p - \theta_i \right), \tag{7}$$

onde v_j^p é a *j*-ésima componente do padrão de entrada V^p.

O algoritmo *back-propagation* atua nos pesos sinápticos, minimizando a função erro, por meio da técnica do gradiente descendente. Neste método, os valores dos pesos são modificados proporcionalmente ao oposto da derivada do erro, de acordo com a seguinte expressão:

$$\Delta w_{ij} = -\gamma \frac{\partial E_i^p}{\partial w_{ij}}, \tag{8}$$

onde γ, denominado taxa de aprendizado, controla quão grande deve ser o "passo" a ser dado. Para maiores detalhes sobre o método do gradiente descendente, bem como sobre outras alternativas, recomenda-se, por exemplo, o texto de Mateus & Luna (1989).

Definindo-se $d_i^{p,k}$ como a saída esperada na *i*-ésima unidade da *k*-ésima camada, quando o *p*-ésimo padrão é apresentado à rede e,

$$O_i^{p,k} = f(a_i^{p,k}), \tag{9}$$

como a saída real na unidade, onde $a_i^{p,k} = \sum_j w_{ij}^k O_j^{p,k-1} - \theta_i$ e $O_i^{p,0} = v_i^p$, a função erro, na camada *k*, poderá ser escrita como:

$$E^k = \frac{1}{2} \sum_{i,p} (d_i^{p,k} - O_i^{p,k})^2. \tag{10}$$

A correção nos pesos na camada de **saída** K é dada pela aplicação da regra da cadeia:

$$\Delta w_{ij}^K = -\gamma \frac{\partial E^K}{\partial w_{ij}^K} = -\gamma \sum_p \frac{\partial E^K}{\partial O_i^{p,K}} \frac{\partial O_i^{p,K}}{\partial a_i^{p,K}} \frac{\partial a_i^{p,K}}{\partial w_{ij}^K} = \gamma \sum_p \delta_i^{p,K} O_j^{p,K-1}, \tag{11}$$

definindo-se:

$$\delta_i^{p,K} = f'(a_i^{p,K})(d_i^{p,K} - O_i^{p,K}). \tag{12}$$

Similarmente, pode ser mostrado que nas camadas **intermediárias** a correção é dada por:

$$\Delta w_{ij}^k = -\gamma \frac{\partial E^k}{\partial w_{ij}^k} = \gamma \sum_p \delta_i^{p,k} O_j^{p,k-1}, \tag{13}$$

definindo-se:

$$\delta_i^{p,k} = f'\left(a_i^{p,k}\right)\sum_j w_{ij}^{k+1}\delta_j^{p,k+1}. \qquad (14)$$

A seguir, é apresentado em forma algorítmica o aprendizado *back-propagation*:

algoritmo

inicialize w_{ij}^k com números aleatórios "pequenos"

enquanto $E \geq \varepsilon$ **faça**

para cada padrão p=1,2,…,P **faça**

/* inicialize entradas */

$O_i^{p,0} \leftarrow v_i^p \ \forall\ i$=1,2,…,I

/* propagar sinal à frente */

para cada camada k=1,2,…,K **faça**

$$O_i^{p,k} \leftarrow f\left(\sum_j w_{ij}^k O_j^{p,k-1} - \theta_i\right) \forall\ i=1,2,\ldots,I$$

fim para

/* calcular delta da camada de saída K */

$$\delta_i^{p,K} \leftarrow f'\left(\sum_j w_{ij}^K v_j^{p,K-1} - \theta_i\right)\left(d_i^{p,K} - O_i^{p,K}\right) \forall\ i=1,2,\ldots,I$$

/* calcular deltas para camadas intermediárias */

para cada camada k= K-1,…,1 **faça**

$$\delta_i^{p,k} \leftarrow f'\left(\sum_j w_{ij}^k O_j^{p,k-1} - \theta_i\right)\sum_j w_{ij}^{p,k+1}\delta_j^{p,k+1} \ \forall\ i=1,2,\ldots,I$$

fim para

/* atualizar pesos */

para cada camada k=1,2,…,K **faça**

$$\Delta w_{ij}^k \leftarrow \gamma\sum_p \delta_i^{p,k} O_j^{p,k-1},\ \forall\ i,j$$

$$w_{ij}^k \leftarrow w_{ij}^k + \Delta w_{ij}^k,\ \forall\ i,j$$

fim para

fim para

fim enquanto

fim algoritmo

O algoritmo *back-propagation* ajusta os pesos das unidades das camadas intermediárias a partir dos erros das unidades da camada de saída. Para iniciar este processo de aprendizado é necessário

estabelecer (i) um conjunto de padrões de entrada e respectivas saídas esperadas; (ii) a taxa de aprendizado γ, (iii) um critério de parada, com o qual se dará o treinamento como concluído e (iv) uma função de transferência. A aplicação do algoritmo *back-propagation* envolve um deslocamento para a frente, através da rede, para calcular as saídas de cada elemento neural da camada de saída e o seu respectivo erro. A segunda fase envolve um deslocamento no sentido contrário, durante o qual o sinal de erro é passado para cada elemento e as mudanças nos pesos são efetuadas. A rede é treinada através da apresentação de todos os dados de treinamento repetidas vezes. Em cada iteração, a diferença entre o valor da saída e o valor esperado, determina a correção que deverá ser feita nos valores dos pesos e dos limiares. Os valores dos pesos são ajustados após cada etapa de treinamento, até que a função erro seja reduzida a um valor tido como aceitável.

3. Uma Rede Neural Aplicável À Manutenção de Motores Elétricos

Neste artigo, a rede neural é apresentada como uma ferramenta para detectar, em tempo real, faltas incipientes em um motor de indução monofásico, com rotor em gaiola. Toda a análise é focada em 2 dos tipos de faltas incipientes mais comuns, quais sejam, (i) faltas na isolação espira-a-espira do enrolamento estatórico e (ii) desgaste dos rolamentos, sob condições de conjugado de carga constante. As entradas para a rede neural foram então definidas como a corrente estatórica I e a velocidade rotórica W. As saídas desejadas são as condições qualitativas (**bom**, **satisfatório** e **ruim)** do isolamento do enrolamento estatórico Nc e do desgaste dos mancais Bc.

O que se busca através do uso de redes neurais é eliminar as dificuldades inerentes ao mapeamento entre as entradas (I,W) e as saídas (Nc,Bc). A complexidade do modelamento matemático do motor de indução, devido ao alto grau de não linearidade da sua dinâmica, é evitada, pelo aprendizado que a rede neural pode realizar. Uma vez treinada, os seus pesos conterão as não-linearidades do mapeamento desejado (Penman & Yin, 1992).

Seguindo uma das estruturas propostas por Chow *et al*. (1991), foi desenvolvida uma rede neural, do tipo *feedforward*, com 3 camadas, tendo 5 nós na camada de entrada, 12 na camada intermediária e 2 na de saída. O conjunto de entradas representa as fontes de sinal que alimentam a rede. Neste trabalho, 5 sinais de entrada são apresentados à rede, a corrente elétrica (I), a rotação do motor (W), o quadrado da corrente (I^2), o quadrado da velocidade (W^2) e o produto destas duas grandezas (I x W). Deve-se ter em mente que cada elemento neural representado na Figura 4 tem a mesma constituição daquele elemento apresentado na Figura 1. Observe-se que o conjunto inicial de treinamento {I,W} foi expandido para 5 dimensões {I, W, I^2, W^2, I x W). Este é um artifício que propicia uma melhor precisão e reduz o tempo de treinamento (Chow *et al.*, 1991).

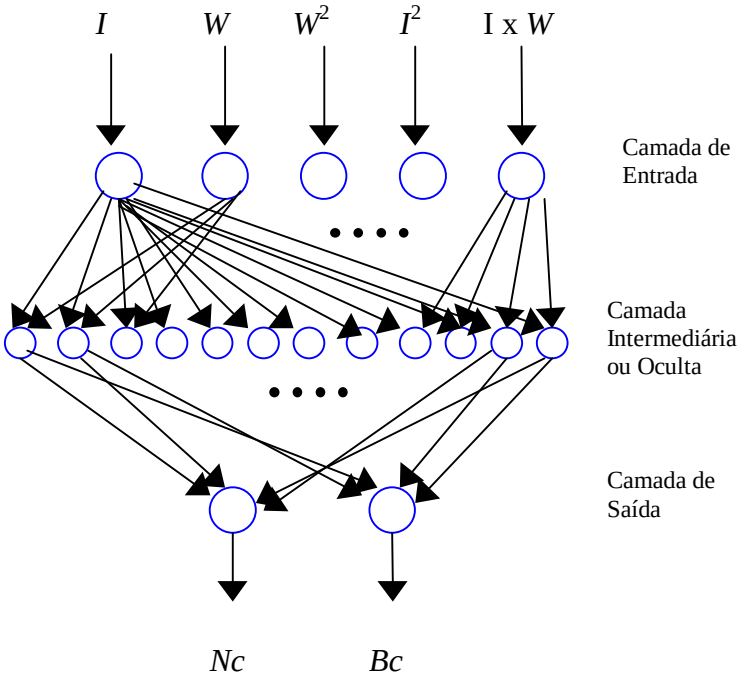


Figura 4 - Estrutura da Rede Neural Implementada

4. Melhorias no Processo de Treinamento

O algoritmo *back-propagation* tem sido utilizado com sucesso em muitas aplicações práticas. Entretanto, ele apresenta alguns problemas, tais como taxa de convergência lenta e propensão a ficar "preso" em um mínimo local.

Considerando-se, por exemplo, o uso da função de ativação sigmoidal, equação (4), um outro problema que pode surgir é a chamada "paralisia de treinamento". Esta paralisia, encontrada durante a fase de treinamento, caracteriza-se por altos valores $a_i^{p,k}$, causados por valores de pesos também elevados, o que conduz a um gradiente muito pequeno. Na região de gradiente pequeno, a correção dos pesos, que é proporcional ao gradiente, fica paralisada. As técnicas conhecidas para resolver este problema são a limitação dos valores dos pesos a um intervalo pré-estabelecido ou a redução dos valores iniciais que lhes são atribuídos, bem como aos limiares.

Às técnicas já citadas no item 2.2, somam-se outras conhecidas, tais como os métodos quasi-Newton (Bengio *et al*., 1994), medida da entropia do erro e incorporação de atratores terminais (Jones & Tsang, 1993). Neste trabalho, será utilizado o método *momentum* (Hertz *et al.*,1991 e Penman & Yin, 1992), para acelerar a convergência da rede e conferir estabilidade. Nesta técnica, a atualização dos pesos é uma ponderação entre o passo atual e os calculados anteriormente. Consiste em aplicar uma constante μ sobre a mudança no vetor pesos da última iteração, somando-se a esta a correção de direção da iteração corrente. Se a última mudança de pesos foi em uma direção particular, o termo *momentum* faz com que a próxima mudança de peso seja mais ou menos na mesma direção que a anterior. Dependendo da situação este pequeno desvio pode ser suficiente para evitar que a rede caia em um mínimo local.

A correção, incorporando o *momentum*, passa a ser dada pela expressão:

$$\Delta w_{ij}^{k}(t+1) = \gamma \frac{\partial E^{k}}{\partial w_{ij}^{k}} + \mu \Delta w_{ij}^{k}(t). \tag{15}$$

A taxa de convergência do processo é fortemente dependente dos valores da taxa de aprendizado γ. Valores muito grandes desta taxa levam a oscilação e instabilidade enquanto valores muito pequenos podem reduzir drasticamente a taxa de convergência. Teoricamente o *momentum* possibilita utilizar valores maiores para a taxa de aprendizado, porque constitui-se em um filtro passa-baixa. Desta forma, a velocidade de convergência é aumentada.

Jones & Tsang (1993) afirmam que não haveria razão para a escolha da função quadrática, equação (10), e sugere o uso da **medida da entropia relativa**, determinando uma nova função erro:

$$E^{k} = \sum_{i,p} \left[d_i^{p,k} \log\left(\frac{d_i^{p,k}}{O_i^{p,k}} \right) + \left(1 - d_i^{p,k}\right) \log\left(\frac{1 - d_i^{p,k}}{1 - O_i^{p,k}} \right) \right]. \tag{16}$$

Considerando-se o uso da função de ativação não-linear (4), as equações do gradiente descendente são modificadas para acomodar a nova função erro, eliminando-se da equação (12) o fator que contém a derivada, resultando em:

$$\delta_i^{p,k} = \left(d_i^{p,k} - O_i^{p,k}\right). \tag{17}$$

O resultado passa a ser o de incorporar pequenas correções nas regiões planas da superfície de erro e, desta forma acelerar a convergência.

5. Resultados Computacionais

O treinamento da rede neural da Figura 4 foi feito com os 35 padrões apresentados na Tabela 1, obtidos a partir de dados experimentais (Chow *et al*., 1991), coletados na condição de carga constante e normalizados no intervalo [0,1]. Embora a ativação dos nós de saída seja uma função contínua, o conjunto de dados de treinamento assume apenas os valores 0,9, 0,5 e 0,1, correspondendo às condições de funcionamento **bom**, **satisfatório** e **ruim**, respectivamente.

Um problema recorrente encontrado nesta aplicação foi a "paralisia de treinamento" mencionada na Seção 4. Tentou-se inicialmente limitar ao intervalo [-1, 1] os valores possíveis para os pesos w_{ij}^{k}, o que resultou em forte oscilação nos erros. O problema foi resolvido reduzindo-se os valores iniciais para os pesos e limiares. Assim, em todos os testes, os pesos foram inicializados pela distribuição uniforme U(-0,4, 0,4) e os limiares, pela distribuição uniforme U(-0,3, 0,3).

A rede foi submetida a diversas simulações para avaliar o desempenho em relação à ausência ou presença do termo *momentum* na equação de correção dos pesos. A Figura 5 compara, para uma mesma faixa de iterações, os erros encontrados com várias combinações da taxa de aprendizado γ e do *momentum* μ. Como critério para análise, na primeira simulação, foi estabelecido o *momentum* nulo.

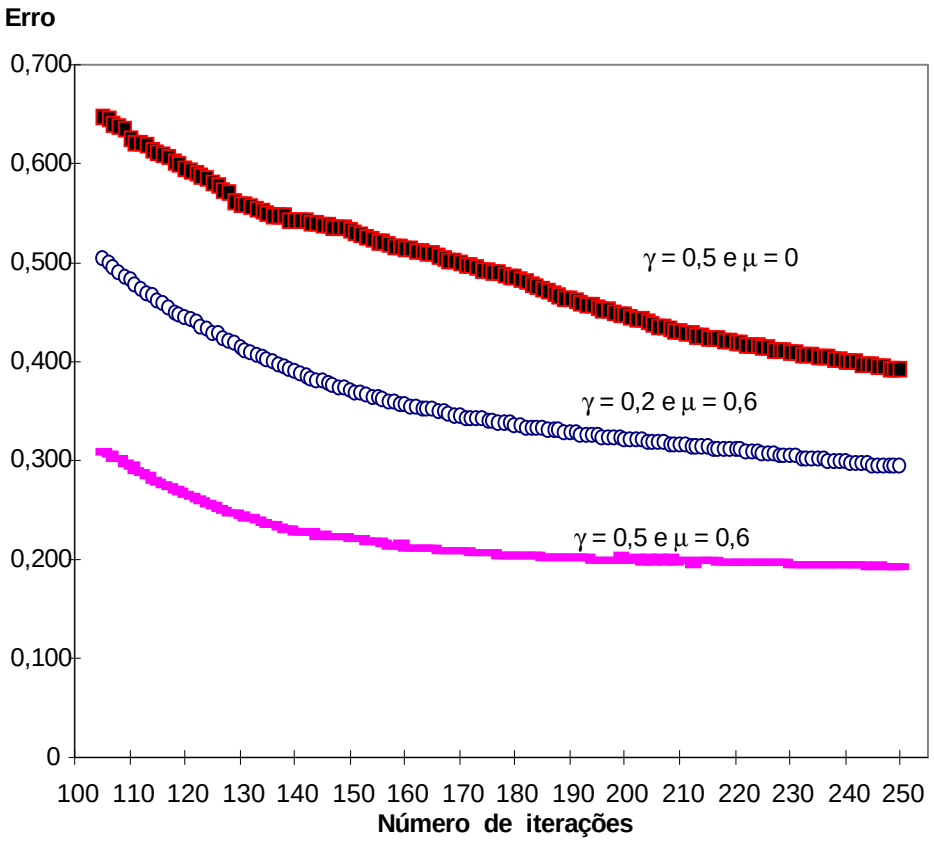
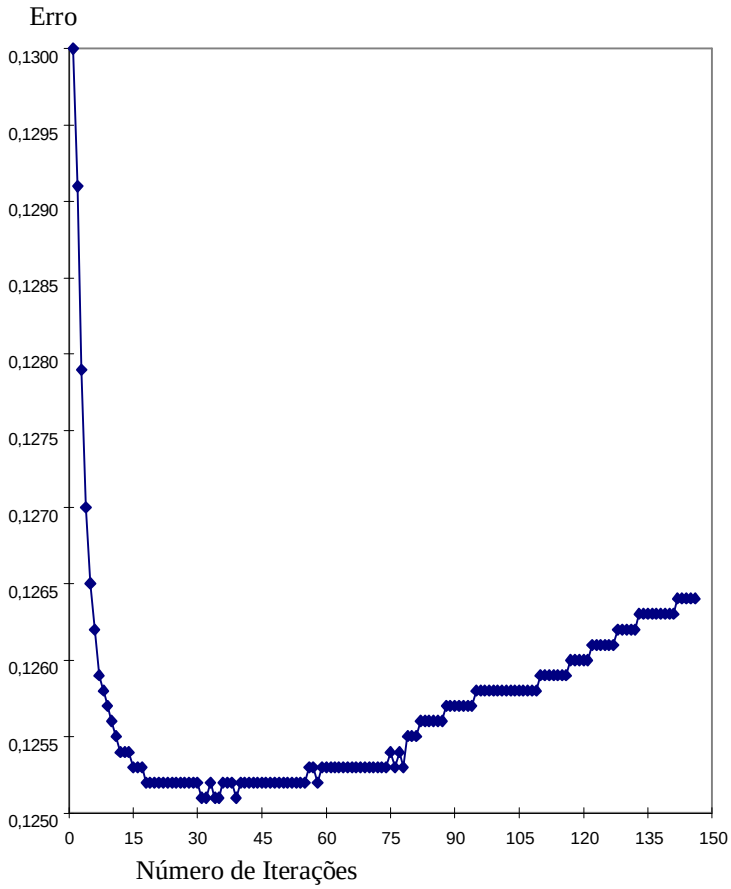


Figura 5 - Convergência para Diversas Combinações γ x μ

A melhoria propiciada pelo *momentum* é evidente. No entanto, para a melhor taxa de aprendizado (γ = 0,5), haveria um valor melhor para o *momentum*? Qual a melhor combinação entre o valor do *momentum* e da taxa de aprendizado, para a situação de interesse? Os estudos realizados até agora indicaram que a determinação destes valores não é direta, podendo acontecer que uma combinação que pareça inicialmente a ideal, tenha, em outra faixa de iteração, baixa taxa de convergência. Para o problema aqui tratado, a determinação foi empírica, determinada segundo o método de tentativa-e-erro.

A Figura 6 ilustra um destes estudos, onde são experimentadas diversas taxas de aprendizado, mantendo-se fixo o *momentum* em μ = 0,6. Conforme pode ser visto, entre a 1ª e a 66ª iteração, usando-se a taxa de aprendizado γ = 0,7, há uma redução no erro, seguida de crescimento. Essa, portanto, não deve ser uma taxa adequada. Prosseguindo-se com uma taxa γ = 0,4, entre a 67ª e a 74ª iteração, observa-se uma estabilização do erro. Reduzindo-se para γ = 0,1, nota-se uma oscilação. Aumentando-se para γ = 0,9, há novo crescimento do erro. Portanto, a taxa adequada deve estar em torno de γ = 0,4. Confirma-se essa suspeita, usando-se a taxa γ = 0,5, entre a 100ª e 106ª iteração, onde também é observada uma estabilização no erro, e usando-se a taxa γ = 0,3, quanto observa-se que o erro cresce.



FAIXA DE ITERAÇÕES:		
DE:	ATÉ:	Taxa de Aprendizado:
1	66	0,7
67	74	0,4
75	78	0,1
79	99	0,9
100	106	0,5
107	137	0,3

Figura 6 - Comportamento do Erro para Diversas Taxas de Aprendizado γ

Foi também testado o método que utiliza a medida da entropia relativa, tendo em vista a necessidade de acelerar o processo de convergência, em aplicações em tempo real. Os resultados obtidos, ilustrados na Figura 7, sugerem uma redução sistemática do erro, com um padrão de decaimento em forma de escada, bem como uma ausência aparente de oscilações no seu valor.

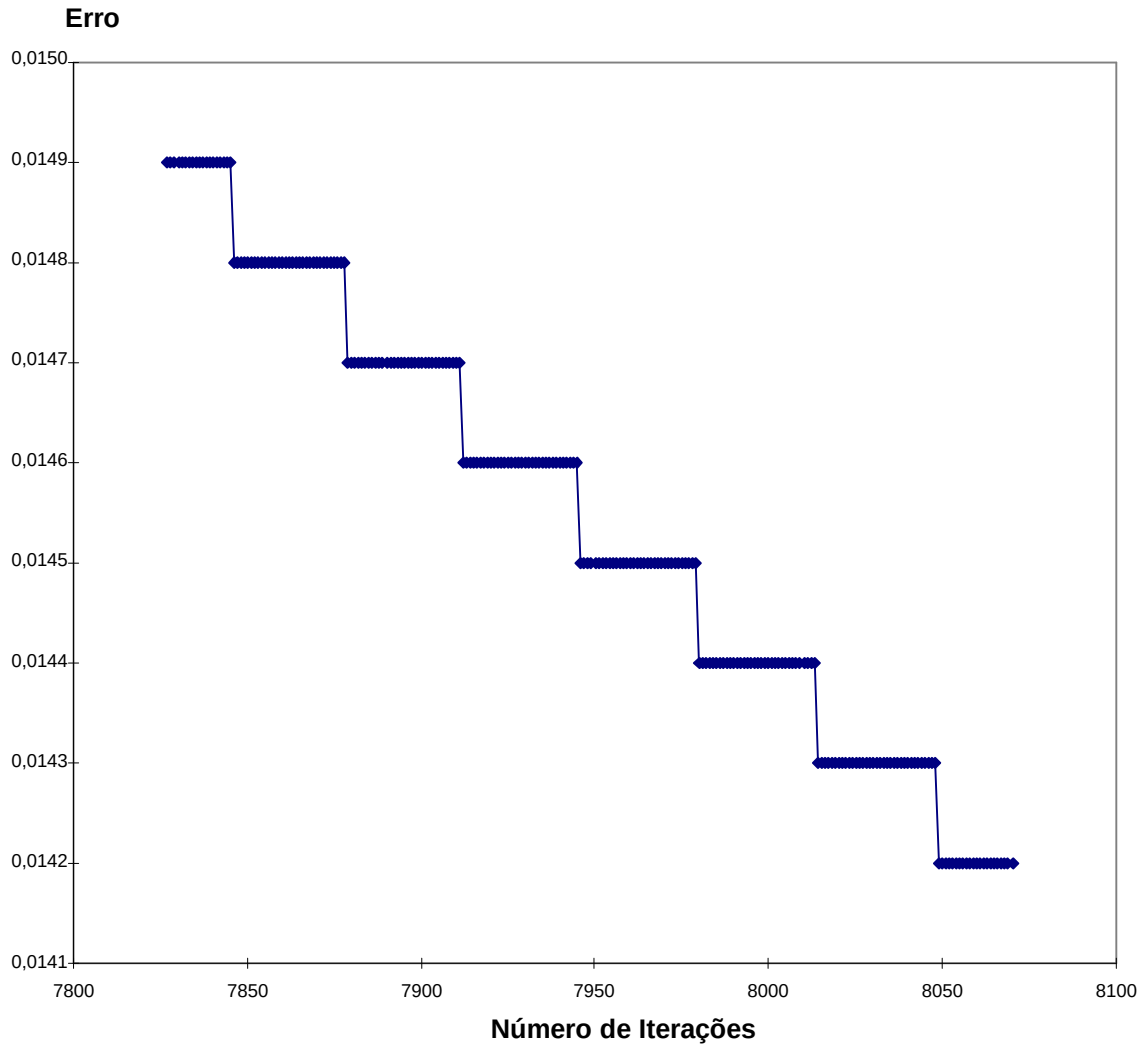


Figura 7 - Convergência pela Medida da Entropia Relativa

6. Conclusões e Observações Finais

Foi apresentada uma rede neural, do tipo *feedforward*, capaz de detectar dois tipos de faltas incipientes em motores de indução com rotor em gaiola, quais sejam (i) faltas de enrolamentos estatóricos e (ii) deterioração dos rolamentos rotóricos. Estudou-se em detalhes a etapa de treinamento, feita por meio do algoritmo *back-propagation*, onde foram apresentas técnicas para sua aceleração.

Conclui-se pelos resultados obtidos que a taxa de aprendizado γ e o termo *momentum* μ são altamente dependentes, sendo empírica, até o presente, a determinação dos valores que conduzem a uma convergência ótima. Maiores valores nestas constantes causam mudanças mais rápidas nos pesos e consequentemente mais rápido será o procedimento de aprendizado. No entanto, podem ocorrer oscilações, sacrificando a convergência. O resultados também sugerem que o efeito do uso da medida da entropia relativa é o de eliminar, aparentemente, as oscilações no erro, reduzindo-o sistematicamente com o número de iterações.

Como propostas de continuidade deste trabalho, sugere-se a realização de outros experimentos que possam complementar as conclusões aqui sintetizadas, dentre eles, variar automa-ticamente a taxa de aprendizado γ durante o treinamento, ou até mesmo tentar encontrar uma função que relacione e arquitetura da rede com os valores ótimos para taxa de aprendizado e o *momentum*.

Tabela 1- Padrões de Treinamento

Padrão	Corrente (*I*)	Velocidade (*W*)	Saída 1 (*Nc*)	Saída 2 (*Bc*)
1	0,00000	0,74359	0,900	0,900
2	0,07000	0,53846	0,900	0,900
3	0,14000	0,38462	0,900	0,500
4	0,21000	0,25641	0,900	0,100
5	0,28000	0,05128	0,900	0,100
6	0,02000	0,87180	0,900	0,900
7	0,00800	0,64103	0,900	0,900
8	0,15000	0,53846	0,900	0,500
9	0,22000	0,35897	0,900	0,100
10	0,29000	0,17949	0,900	0,100
11	0,03000	0,89744	0,900	0,900
12	0,11000	0,71795	0,900	0,900
13	0,17000	0,58974	0,900	0,500
14	0,24000	0,41026	0,900	0,100
15	0,31000	0,23077	0,900	0,100
16	0,10000	1.00000	0,500	0,900
17	0,18000	0,79487	0,500	0,900
18	0,25000	0,64103	0,500	0,500
19	0,32000	0,46154	0,500	0,100
20	0,38000	0,28205	0,500	0,100
21	0,20000	0,92308	0,500	0,900
22	0,27000	0,76923	0,500	0,900
23	0,33000	0,56410	0,500	0,500
24	0,40000	0,41026	0,500	0,100
25	0,47000	0,25641	0,500	0,100
26	0,62000	0,82051	0,100	0,900
27	0,70000	0,61538	0,100	0,900
28	0,76000	0,46154	0,100	0,500
29	0,83000	0,28205	0,100	0,100
30	0,92000	0,10256	0,100	0,100
31	0,70000	0,69231	0,100	0,900
32	0,77000	0,48718	0,100	0,900
33	0,85000	0,30769	0,100	0,500
34	0,92000	0,15385	0,100	0,100
35	1.00000	0,00000	0,100	0,100

Agradecimentos

Os autores gostariam de deixar expressos os seus agradecimentos ao CNPq e à FAPEMIG, pelo auxílio financeiro parcial prestado a este trabalho, a Carlos Antônio Ferreira, pelas sugestões feitas e a dois revisores da revista Pesquisa Operacional, pelas valiosas críticas e sugestões.

Referências

Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, **5**,157-166.

Chow, M., Mangum, P. M.& Yee, S. (1991). A neural network approach to real-time condition monitoring of induction motors. *IEEE Transactions on Industrial Electronics*, **38**, 448-453.

Chow, M., Mangum, P. M. & Yee, S. (1990). Real time application of artificial neural networks for incipient fault detection of induction machines. *Proceedings of the 3rd International Conference of Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 1030-1036, Charleston, USA.

Haykin, S. H. (1994). *Neural Networks: A comprehensive foundation*. MacMilliam College Publishing, New York, USA.

Hertz, J. A., Krogh, A. & Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, Redwood City, USA.

Jones, C. R. & Tsang, C. P. (1993). On the convergence of feed-forward neural networks: Incorporating terminal attractors. *Proceedings of the IEEE International Conference on Neural Networks*, 253-259, USA.

Kartalopoulos, S. V. (1996). *Understanding Neural Networks and Fuzzy Logic -Basic Concepts and Applications*. IEEE, Inc., New York, USA.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems - A dynamical systems approach to machine intelligence*, Prentice - Hall International, Inc., USA.

Kovács, Z. L. (1996). *Redes Neurais Artificiais - Fundamentos e Aplicações*. Edição Acadêmica, São Paulo, Brasil.

Mateus, G. R. & Luna, H. P. L. (1986). *Programação Não-Linear*. V Escola de Computação. UFMG, Belo Horizonte, Brasil.

Penman, J. & Yin, C. M. (1992). The application of artificial neural networks in identifying faults in induction machines. Department of Engineering, University of Aberdeen, Aberdeen, UK.

Tsoukalas, L.H. & Uhrig, R. E. (1997). *Fuzzy and Neural Approaches in Engineering*. John Wiley & Sons, Inc - New York, USA.