Wavelets e Componentes Principais no pré-processamento de sinais aplicado a Redes Neurais Artificiais

ADRIANO MARTINS MOUTINHO[1], Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia Elétrica, UERJ, Rio de Janeiro, Brasil.

LUIZ BIONDI NETO[2], Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia Elétrica, UERJ, Rio de Janeiro, Brasil.

JOÃO CARLOS C. B. SOARES DE MELLO[3], Universidade Federal Fluminense, UFF, Rio de Janeiro, Brasil.

ELIANE GONÇALVES GOMES[4], Embrapa Monitoramento por Satélite, Rio de Janeiro, Brasil.

Resumo.Uma rede neural pode não apresentar uma boa performance de reconhecimento mesmo quando treinada com uma seqüência de dados consistente. Para obtermos redes classificatórias mais confiáveis, há necessidade de se tratar o sinal de entrada, retirando-se a informação não relevante.

Este trabalho tem como intuito comparar os mais diversos métodos usados no processamento dos sinais antes da sua aplicação às redes neurais, e escolher aquele ou a combinação deles que possibilite a melhor taxa de reconhecimentos corretos. Para realizar a análise comparativa é usado um caso simples de reconhecimento de dígitos manuscritos de 0 a 9. Os sinais de entrada são submetidos a tratamento por métodos de compressão Wavelet, de componentes principais, métodos de centralização e outros.

Palavra-chave: Wavelet, Componentes Principais, Redes Neurais Artificiais.

1. Introdução

Redes neurais artificiais (RNA) vêm sendo usadas com sucesso no reconhecimento de padrões gráficos há muitos anos. O projeto de uma RNA consiste basicamente em duas etapas: o treinamento e os testes.

[1]adrianomoutinho@hotmail.com
[2]lbiondi@embratel.com.br
[3]jcsmello@bol.com.br
[4]eliane@cnpm.embrapa.br

Durante a etapa de treinamento, a rede recebe uma seqüência de padrões, isto é, um conjunto de dados e o alvo a ser atingido. No caso da rede projetada para reconhecer os padrões manuscritos dos números de 0 a 9, durante a etapa de treinamento a rede receberá diversas seqüências contendo estes números.

Uma relativa melhora na capacidade de reconhecimento pode ser obtida a partir de um certo tratamento aplicado aos dados de entrada. Na maioria dos casos, isto não é opcional, pois normalmente os dados que se deseja classificar possuem dimensões muito grandes e carregam pouca informação, ou seja, baixa entropia.

2. Fundamentação matemática

2.1. Redes Neurais

As redes neurais artificiais têm seu princípio baseado no funcionamento do cérebro humano onde, estima-se, existirem cerca de 10 bilhões de neurônios e 60 trilhões de sinapses apenas no córtex [2].Pode-se fazer uma analogia do funcionamento do neurônio biológico e propor um modelo para o mesmo. O modelo mais comumente aceito para uma unidade neural artificial é o que considera o neurônio como um somatório de diversos sinais de entrada [1], também denominados estímulos, e que na verdade são os equivalentes matemáticos dos mais variados sinais recebidos pelo cérebro humano como, por exemplo, imagens, sons, informações de tato, de olfato e outros. Dessa forma, cada neurônio recebe este o conjunto de entradas, sendo que cada uma é ponderada por um certo peso W_j, também denominado peso sináptico. Todas as entradas multiplicadas por seus devidos pesos são somadas a um certo valor fixo externo chamado bias (b_k). A expressão final (ν_J) é então aplicada à função de ativação, resultando finalmente na saída y_j, eventualmente aplicada a entrada de uma nova unidade processadora neural adjacente.As equações (2.1) e (2.2) descrevem o modelo do neurônio artificial:

$$\nu_j = \sum_{m=n=1}^{n=j\ m=i} \tag{2.1}$$

$$y_j = \varphi(\nu_j + b_j) \tag{2.2}$$

A função de ativação ν_j é um limitador que geralmente normaliza a saída entre 0 e 1, ou entre -1 e 1. Na maioria dos casos, utiliza-se uma função sigmóide pois a mesma exibe um bom balanceamento entre características lineares e não lineares. Esta função é mostrada na equação (2.3).

$$\varphi(\nu_j) = \frac{1}{(1 + e^{-\alpha\nu_j})} \tag{2.3}$$

Os valores de todos os pesos sinápticos de cada unidade neural podem ser ajustados de forma a fazer a rede desempenhar qualquer função desejada. O processo que ajusta os pesos para uma determinada função é chamado de treinamento, onde vários padrões são apresentados em conjunto com a resposta correta presumível. O

bias e os pesos são atualizados por um processo conhecido como retropropagação do erro [3]. A retropropagação permite que todos os pesos dos neurônios sejam ajustados ainda que estes não pertençam à camada de saída. O algoritmo tem a capacidade de propagar o erro até as camadas intermediárias, e assim efetuar a correção [1]. Para os neurônios de saída, o erro pode ser calculado da seguinte forma.

$$e_j(n) = d_j(n) - y_j(n) \tag{2.4}$$

Onde e_j é o erro, d_j é o valor esperado e y_j é o valor atual do neurônio de saída. A energia do erro será o somatório da energia de todos os erros, portanto:

$$\xi(n) = \frac{1}{2}\sum e_j^2(n) \tag{2.5}$$

A retropropagação aplica então, a cada peso sináptico, um certo ΔW_j proporcional a derivada parcial $\partial\xi(n)/\partial W_j(n)$, dada pela equação (2.6).

$$\frac{\partial\xi(n)}{\partial W_j(n)} = \frac{\partial\xi(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial \nu_j(n)}\frac{\partial \nu_j(n)}{\partial W_j(n)} = -e_j(n)\cdot\varphi'(\nu_j(n))\cdot y_j \tag{2.6}$$

Onde y_j é a saída do neurônio em questão, ν_j é o valor antes da aplicação da função de ativação, $\xi(n)$ é energia do erro em relação ao valor esperado e W_j é o peso a ser ajustado. Assim, os neurônios de saída são modificados por:

$$\Delta W_j(n) = -\eta\frac{\partial\xi(n)}{\partial W_j(n)} = \eta\cdot\partial_j(n)\cdot y_j(n) \tag{2.7}$$

O parâmetro η é a taxa de aprendizado a ser definida pelo usuário para controlar a velocidade de convergência do método. Já $\partial_j(n)$ é definido como gradiente local, e é dado por $e_j(n)\cdot\varphi'(\nu(n))$.

Caso a rede possua neurônios intermediários, o que é muito importante pois capacita o sistema a aprender tarefas mais complexas, precisa-se propagar o erro até a camada em que se deseja corrigir os pesos. Assim, para um neurônio intermediário j imediatamente seguido de um neurônio de saída k, o gradiente local é redefinido como:

$$\partial_j(n) = \varphi'(\nu(n))\cdot\sum_k \partial_K\cdot W_K \tag{2.8}$$

O somatório de todos os gradientes multiplicados pelos pesos do neurônio posterior entram no cálculo do gradiente local, e posteriormente de ΔW_j. Quando todos os pesos da rede são atualizados por ΔW_j um novo erro é calculado e o processo se reinicia até que o objetivo seja atingido.

2.2. Método de Redução de Dimensões por Componentes Principais (MCA)

O método de componentes principais (MCA) [2] é uma forma de reduzir a dimensão dos dados de entrada sem eliminar informação relevante. É bastante útil no tratamento dos dados aplicados à rede neural, possibilitando treinamentos mais rápidos

e o aumento da generalização. Inicialmente consideramos um vetor X de dados de entrada tendo média nula, ou seja $E[X] = 0$. Considere então, que o vetor X é projetado em outro vetor unitário q. A projeção A é o produto interno dos dois vetores conforme mostra a equação (2.9):

$$A = X^T q = q^T X \tag{2.9}$$

A variância do vetor projetado é função de q, e é representado por $\psi(q)$

$$\psi(q) = \sigma^2 = E[A^2] = E[(q^T X)(X^T q)] = q^T R q \tag{2.10}$$

O vetor R na equação (2.11) é definido como matriz autocorrelação, sendo igual a:

$$R = E[X X^T] \tag{2.11}$$

Devemos encontrar então quais vetores q possibilitam que a variância da projeção seja extrema ou estacionária. Neste caso, pequenas perturbações em q não modificam o valor da variância:

$$\psi(q) = \psi(q + \partial q) = (q + \partial q)^T R(q + \partial q) \tag{2.12}$$

Desenvolvendo a equação e eliminando os termos de segunda ordem temos:

$$\psi(q + \partial q) = \psi(q) + 2(\partial q)^T R q \tag{2.13}$$

Assim, como a equação (2.12) mostra que as pequenas perturbações não influenciam na variância, temos:

$$\partial q^T R q = 0 \tag{2.14}$$

Adicionalmente, como a perturbação ∂q não pode mudar o módulo do vetor unitário q, admitiremos que ∂q e q são ortogonais, ou seja, apenas uma mudança de direção será tolerada:

$$\partial q^T q = 0 \tag{2.15}$$

Combinando as equações (2.14) e (2.15), sendo que nesta última adicionaremos um fator de escala λ pois os elementos q são adimensionais:

$$(\partial q)^T (Rq - \lambda q) = 0 \tag{2.16}$$

$$Rq = \lambda q \tag{2.17}$$

A equação (2.17) é um problema típico de álgebra linear e possui solução não trivial quando temos λ igual aos autovalores e q aos autovetores da matriz R. Assim, se existem m autovalores para a matriz R, teremos:

$$Rq_j = \lambda q_j \; para \; j = 1, 2, \cdots, m \tag{2.18}$$

Ou, equivalentemente, considerando que $q_j^T = q_j^{-1}$:

$$q_j^T R q_j = \lambda_j \; para \; j = 1, 2, \cdots, m \tag{2.19}$$

Se considerarmos novamente o vetor a_j como a projeção de X sobre o unitário q, conforme a equação (2.20):

$$a_j = q_j^T X \; para \; j = 1, 2, \cdots, m \tag{2.20}$$

A equação (2.20) é considerada como a fórmula de análise de CPA, que decompõe o vetor de entrada em projeções sobre q. O vetor X pode ser reconstituído completamente com a fórmula da equação (2.20), conhecida como equação de síntese:

$$X = \sum_{j=1}^{m} a_j q_j \tag{2.21}$$

Se ao invés de reconstituirmos o vetor X completamente com todas as suas projeções, truncarmos o somatório da equação (2.21) após n termos, um novo vetor $\bar{X}$ terá sua dimensão reduzida a n. Por causa deste truncamento, teremos um erro igual ao somatório de todas as variâncias das projeções eliminadas, ou seja:

$$\xi_{Truncamento} = \sum_{j=n+1}^{m} a_j q_j \tag{2.22}$$

Segundo a equação (2.10), a variância das projeções é $qTRq$ e, segundo a equação (2.19), $qTRq$ é igual ao autovalor associado à projeção. Assim, a variância da projeção eliminada é igual ao autovalor associado:

$$\sigma^2_{Eliminada} = \sum_{j=n+1}^{m} \lambda_j \tag{2.23}$$

Portanto, se calcularmos anteriormente os m autovalores da matriz autocorrelação R de X e colocarmos os mesmos em ordem crescente, poderemos eliminar apenas as projeções dos menores autovalores, o que garantirá uma redução na dimensão de X com a menor perda possível de variância, ou seja, de informação. [2].

2.3. Método de decomposição e compressão por Wavelet

A transformada de Fourier [1] é talvez a mais conhecida técnica de análise de sinais, possibilitando uma visão espectral do mesmo. Porém, alguns sinais possuem abruptas variações no tempo, características transitórias e outras informações que não são mostradas claramente na transformada de Fourier.

Uma das alternativas para resolver este problema é a transformada Wavelet. A diferença entre a transformada de Fourier e a transformada de Wavelet é que a primeira decompõe o sinal $f(t)$ em várias componentes senoidais de diferentes amplitudes e freqüências, enquanto a segunda decompõe o sinal em várias componentes de uma onda wavelet, em várias escalas e amplitudes diferentes. Existem vários tipos de onda wavelet. A figura (1) mostra uma das mais comuns: Algumas características dos sinais são mais facilmente extraídas usando a transformada de wavelet

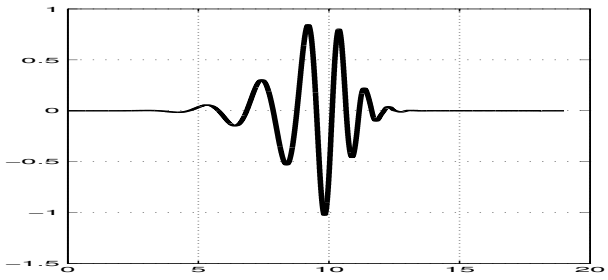

Figura 1: Uma onda Wavelet (db10)

do que a de Fourier, pois a forma de onda wavelet é semelhante às características usualmente encontradas em sinais de voz e imagem.

A aplicação da transformada de Wavelet é bastante simples. Inicialmente, uma das formas de onda Wavelet é escolhida como base e, a seguir, esta forma de onda é comparada com uma janela do sinal original, calculando-se assim um coeficiente de similaridade. Este coeficiente é calculado conforme a equação (2.24):

$$C(escala, posicao) = \int_{-\infty}^{\infty} f(t) \cdot \psi(escala, posicao) dt \tag{2.24}$$

3. Modelagem do Problema

Para comparar as aplicações dos métodos mostrados na seção 2 iremos utilizar um problema típico de identificação de padrões de dígitos manuscritos. Um database contendo 1000 dígitos será usado para treinar a rede, enquanto outro database também de 1000 dígitos será usado como validação. A validação é um método de auxílio ao treinamento, onde a cada apresentação completa dos padrões, o calcula-se a soma dos erros quadráticos e coloca-se no mesmo gráfico junto à evolução do erro do database de treinamento. A validação é especialmente útil, pois mostra claramente a generalização da rede, bem como o ponto ótimo de parada do algoritmo.

Usamos para a comparação uma rede neural típica contendo uma camada de entrada de 64 neurônios, duas camadas intermediárias de 20 neurônios, e uma camada de saída de 10 neurônios. Todas as camadas tem função de ativação sigmóide.

4. Implementação

Inicialmente, iremos verificar a performance da rede sem nenhum tipo de pré-processamento. A curva preta na Figura (2) mostra a evolução do treinamento pela soma dos erros de classificação elevados ao quadrado, ou SSE. Na curva tracejada é mostrada o SSE do conjunto 1000 dígitos de validação. Podemos verificar, através das curvas da Figura (2), que o desempenho da rede não é satisfatório. O

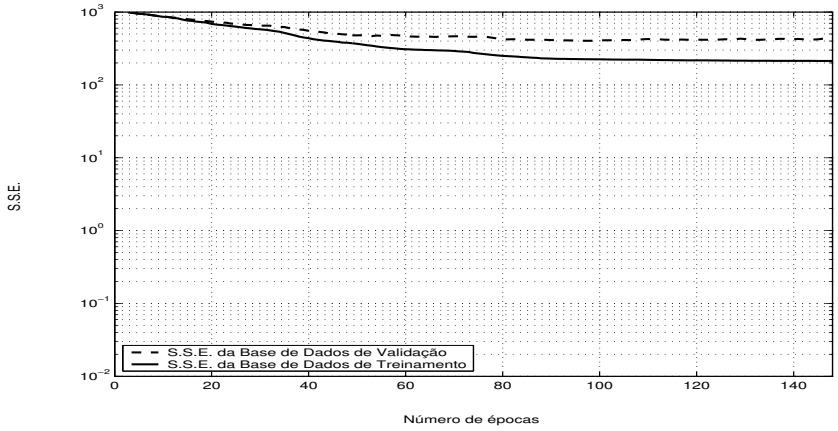

Figura 2: Dados sem nenhum pré-processamento

melhor ponto, ou seja, o valor mínimo da curva tracejada, está em torno da época de número 100. Conseguiremos no máximo um número de erros de classificação próximo a 400 [5] a cada 1000, uma taxa de reconhecimento de apenas 60%. Essa constatação confirma a necessidade de se fazer um pré-processamento.

4.1. Compressão Wavelet

Se usarmos compressão Wavelet para reduzir o tamanho das figuras que a RNA deve aprender, e aplicarmos como novo sinal à rede neural, teremos um avanço na capacidade de reconhecimento, como pode ser visualizada na curva tracejada da Figura (3). O ponto de mínimo da curva tracejada está em 300 possuindo assim, 300 erros a cada 1000 testes, o equivale a uma taxa de reconhecimentos corretos de 70%. Obviamente isto ainda não é o suficiente para um reconhecimento confiável, precisamos utilizar outros métodos de tratamento aliados à compressão Wavelet.

4.2. Retirada de Espaços Brancos

Outra informação irrelevante nas matrizes bitmaps é o tamanho do dígito. Se um determinado número ocupar todo o quadro em 16x16 ou apenas parte dele, teremos modificado totalmente a matriz de entrada, o que torna difícil o aprendizado e a posterior classificação.

Um método eficaz para minimizar estas variações é a técnica da retirada de brancos. Um programa de computador detecta qual a janela mínima que corta o dígito numérico sem perda de nenhum pixel. A Figura (4) mostra o resultado final

[5] Na verdade, este valor é o SSE mínimo obtido no database de validação durante o treinamento, podendo ser dito, aproximadamente, que se trata do número de erros de classificação deste database, uma boa aproximação para a generalização da RNA.

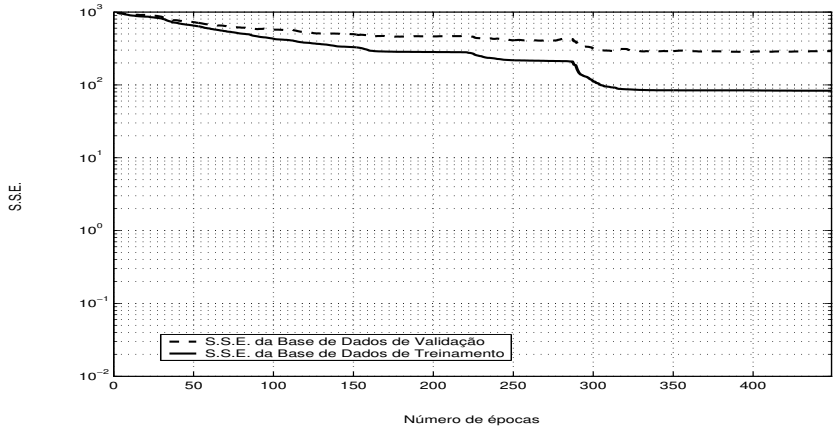

Figura 3: Dados comprimidos com Wavelet

e a figura inicial, onde o dígito 2 é cortado e novamente ampliada até o tamanho original. Esta técnica foi criada pelos autores e mostrou-se muito útil quando apli-

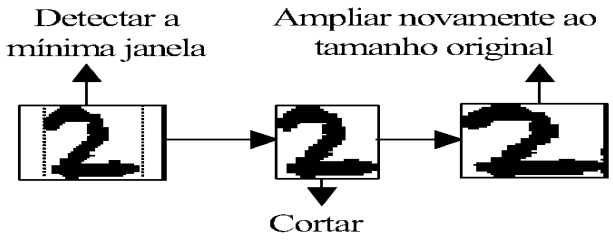

Figura 4: Método de retirada de brancos aplicada a uma dígito manuscrito

cada a dígitos numéricos manuscritos, pois, como se pode perceber na Figura (5), houve melhorias na performance da rede, que chega a 78% de reconhecimento.

4.3. Método dos Componentes Principais (MCA)

Ao aplicarmos o método MCA, descrito em detalhes na seção 2.2, deveremos ter um conjunto de dados o maior possível, para que a análise conclua corretamente quais são os elementos das imagens que realmente não possuem pouca informação relevante.

Os resultados serão superiores aos que já foram apresentados, desde que obviamente não se retire muitos elementos das imagens, apenas aqueles cuja variância seja menor que cerca de 0.2% do total. A figura (6) mostra o treinamento, onde obteve-se o mínimo da curva tracejada de validação perto de 170 erros em 1000,

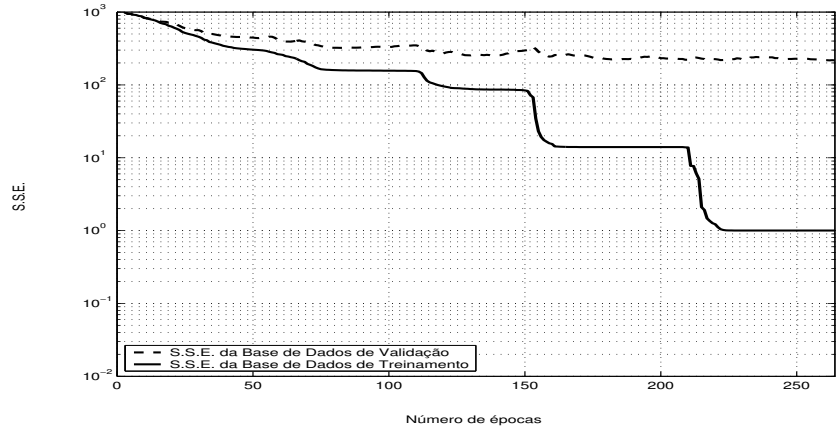

Figura 5: Wavelet e Técnica de Retirada de Brancos

uma taxa de reconhecimento de 83%. Dessa vez foi utilizada a técnica MCA a partir de uma matriz usando wavelet aliada à técnica de retirada de brancos.

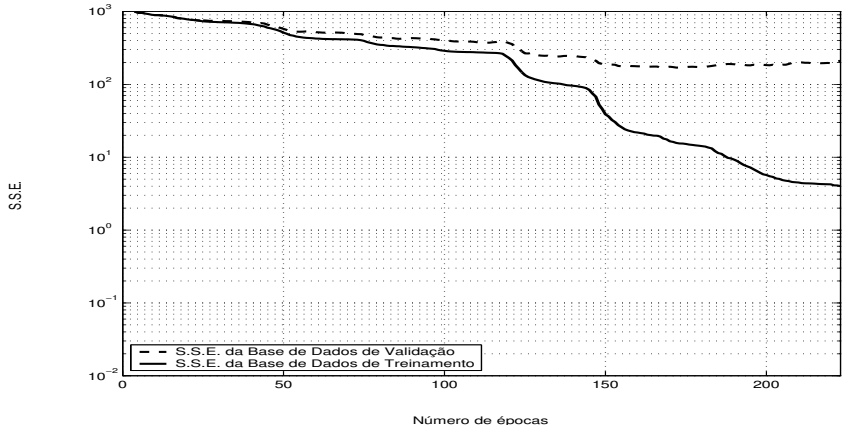

Figura 6: Dados usando Wavelet, retirada de brancos e comp. principais.

5. Conclusões

A técnica MCA se mostrou muito importante no pré-processamento dos dados antes da sua aplicação à RNA. As vantagens foram a simplificação da rede, e o aumento da generalização.

O método de compressão wavelet mostrou-se superior ao método nearest neighborhood, o mais usado em redução de imagens, obtendo a melhor generalização. Já o método da retirada de brancos substituiu a simples centralização dos dígitos, pois esta técnica tanto centraliza os dígitos quanto os normaliza em tamanho. A melhor combinação de técnicas foi o uso do método de retirada de brancos aliado à compressão Wavelet, seguido de MCA. Os resultados atingiram 83% de reconhecimentos corretos, podendo-se melhorar, aumentando-se o database de treinamento. A tabela 1 resume os resultados encontrados. Em face dos bons resultados obtidos na pesquisa, vários outros problemas de classificação utilizando redes neurais já estão sendo estudados utilizando-se as técnicas citadas aqui neste trabalho.

Tabela 1: Quadro comparativo.

Métodos usados	Mínimo da validação	% Acertos
Nenhum	400	60%
Somente Wavelet	300	70%
Wavelet e R. Brancos	220	78%
Wavelet, R. Brancos e Comp. Principais	170	83%

Abstract.A neural network may not achieve a good recognizing performance even if it is trained with a consistent database. If we want to obtain reliable classification, it is necessary to process the input signal, removing unnecessary information.

This paper is a comparison between some methods used in signal pre-processing before these signals are applied to neural networks. After this comparison, it is possible to choose a method, or a combination of them, that give us the best results. Our analysis is based on a simple case where it is necessary to recognize digits from 0 to 9. The input signals are preprocessing using wavelet compression, main components analysis, centralization methods and others.

Key-words: Wavelet, Main Components Analysis, Artificial Neural Networks

Referências

[1] Simon, Haykin; Neural Networks, a comprehensive Foundation; 2ª edition. Prentice Hall Press, New Jersey, 1999.

[2] Shepherd, G.M. and C. Koch. The synaptic organization of the brain. Oxford University Press, New York, 1990.

[3] Jacek M. Zurada, Introduction to Artificial Neural Systems, West Publishing Company, New York, 1992.

[4] Moutinho e Biondi, Comparação de métodos de pré-processamento de sinais aplicado a redes neurais artificiais, Congresso Brasileiro de computação (CBCOMP 2002), itajaí, SC, Brasil (2002).