UTILIZAÇÃO DE UM ALGORITMO DE APRENDIZAGEM LMS PARA ENCONTRAR OS COEFICIENTES DE UM POLINÔMIO QUE DESCREVE UMA SUPERFÍCIE

Diego SOUZA (1); Bosco MOREIRA (2); Carlos FONSECA (3)

(1) CEFET-RN, Av. Sen. Salgado Filho, 1559, Tirol, Natal-RN – CEP: 59015-000, (84)88858027, fax, e-mail: diegouern@gmail.com
(2) UERN, e-mail: boskyn9@gmail.com
(3) UERN, e-mail: carlosandre@uern.br

RESUMO

Rede Neural Artificial (RNA) é uma maneira diferente de processar informações, o que é feito, por ela, de maneira paralela e não-linear. O poder computacional das RNAs vem da sua estrutura paralelamente distribuída e da sua habilidade de aprender. Neste trabalho foi desenvolvido, computacionalmente, um algoritmo de aprendizagem Least Mean Square Error (LMS) capaz de emular os coeficientes de uma reta na quarta dimensão. O sistema inicialmente escolhe, randomicamente, um exemplo, que no caso é uma reta com quatro coeficientes que serão multiplicados por três constantes e somada com uma quarta constante. A equação da reta será: (4x+6y+9z+15), onde os valores aleatórios são: x, y, z. O aplicativo possui como entradas fatores de correção do sistema, número de iterações e a quantidades de exemplos dados pelo usuário. O sistema é totalmente Java, possui uma interface agradável e é possível salvar em modo texto os resultados do sistema.

Palavras-chave: algoritmo de aprendizagem LMS, Java, Redes Neurais Artificiais

1. INTRODUÇÃO

As motivações para o uso das Redes Neurais Artificiais (RNA) são a sua grande capacidade de aprendizado, generalização e adaptação.

O objetivo deste artigo é fornecer uma introdução às RNAs, através de conceitos e do estudo de caso de um aplicativo que utiliza o algoritmo Least Mean Square (LMS) que é um Perceptron de Camada Única (PCU). O algoritmo LMS é de simples implementação, contudo muito efetivo em relação à sua aplicação. Ele é o principal algoritmo da filtragem adaptativa linear. O filtro adaptativo tem sido aplicado com sucesso em campos tão diversos, como antenas, sistemas de comunicação, sistemas de controle, radar, sonar, sismologia e engenharia biométrica.

O aplicativo apresentado gera, inicialmente, dois pontos, pelos quais passa uma única reta. A partir dessa reta gerada, seu objetivo passa a ser originar uma equação representante de outra reta que se aproxime da equação da reta gerada. Isso deve ser feito com uma tolerância, para mais ou para menos, previamente estabelecida. O que torna o aplicativo ainda mais interessante é que ele trabalha com quatro dimensões, e não apenas com dois ou três, como é usual. Resumindo o aplicativo gera uma reta aleatória na quarta dimensão e depois cria outra reta que tenta se aproximar ao máximo da reta gerada.

2. CONCEITOS FUNDAMENTAIS

2.1. O que são RNAs?

São sistemas formados por unidades de processamento (nodos) organizados de maneira distribuída e paralela, através dos quais é possível calcular funções matemáticas diversas. Essas unidades são organizadas em uma ou mais camadas intensamente conectadas. Suas conexões podem ter pesos ou não, tais pesos armazenam o conhecimento e servem para ponderar as entradas recebidas por cada neurônio da rede.

Muitos fundamentos das RNAs são baseados nos conceitos da biologia, a fim de reproduzir seu comportamento e dinâmica. Atualmente, as RNAs diferem bastante das redes biológicas, devido a características tecnológicas.

2.2. Neurônio Artificial

O neurônio é a unidade principal de processamento de informações de uma RNA. Os neurônios artificiais, assim como os neurônios biológicos, são unidades de processamento independentes e geralmente são bastante numerosos em uma RNA. A primeira representação artificial de um neurônio biológico foi idealizada pelo neuroanatomista Warren McCulloch em conjunto com o matemático Walter Pitts. Esse neurônio é referenciado pelo nome de Neurônio de McCulloch e Pitts (McP).

O modelo do neurônio McP desenvolvido inicialmente é uma simplificação do que se sabia na época sobre o neurônio. O modelo consiste em m terminais de entrada (x_1, x_2, ... x_m), representando os dendritos do neurônio biológico, e apenas uma saída, representando o axônio.

Pesos acoplados às entradas são utilizados para emular o comportamento das sinapses biológicas (w_{k1}, w_{k2}, ..., w_{km}), onde k diz respeito ao índice neurônio, e os índices de 1 até m dizem respeito às entradas.

As sinapses podem ser inibitórias ou excitatórias, dependendo dos valores dos pesos, se são negativos ou positivos, respectivamente. A entrada multiplicada pelo peso correspondente representa o efeito de uma sinapse nervosa, como se pode observar na Figura 1, esse efeito é dado por $x_i\, w_i$.

O Neurônio McP possui um combinador linear, que é um somador de todas as entradas ponderadas pelos seus respectivos pesos somados ao bias, o que resulta no Campo Local Induzido (v) do neurônio. Uma função de ativação (φ) é usada para restringir a amplitude da saída do neurônio.

2.2.1. Neurônio Artificial

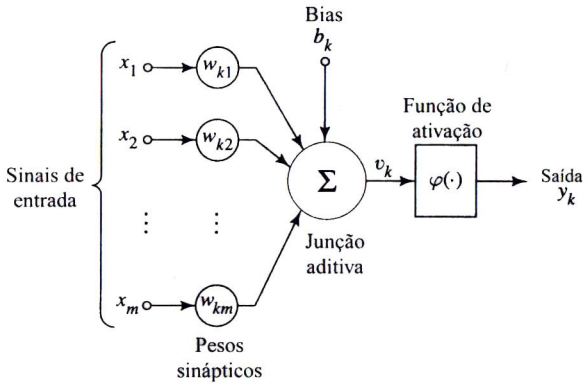

Figura 1 – Neurônio de McP

Tabela 1 – Legenda dos Símbolos Utilizados

Símbolo	Significado
x	Entrada
y	Saída
w	Peso
b	Bias
φ	Função de Ativação
u	Combinador Linear
η	Eta
e	Erro
v	Campo Local Induzido

Em termos matemáticos, podemos descrever um neurônio *k* escrevendo as seguintes equações:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \qquad \text{[Eq. 01]}$$

$$v_k = u_k + b_k \qquad \text{[Eq. 02]}$$

$$y_k = \varphi(v_k) \qquad \text{[Eq. 03]}$$

2.3. Função de Ativação

A Função de Ativação (FA) recebe como entrada o campo local induzido do neurônio e fornece uma resposta, de acordo com o tipo da FA. Há vários tipos de FA, entre elas encontram-se a função de ativação do tipo linear, sigmóide e limiar. A escolha da função de ativação a ser utilizada depende da aplicação, pois é ela que irá restringir a saída do neurônio para valores utilizáveis pela aplicação.

No aplicativo implementado foi utilizada a FA do tipo linear. Este tipo de FA foi escolhido, pois limita a variação da saída do neurônio a uma faixa determinada, nesse caso específico de -327 a 357. A FA recebe como entrada o valor do Campo Local Induzido, *v*. Se o *v* for maior que o Limite Máximo (LMax), 357, a saída será LMax, se for menor que Limite Mínimo (LMin), a saída será LMin, -327, e caso esteja entre esses valores a saída será o próprio *v*. No aplicativo foi implementado um índice de aprendizado (η) variável, para resolução de possíveis loops infinitos e uma aceleração da analise do problema, toda vez em que a saída do sistema ultrapassa os limites máximos e mínimos do problema, o eta (*e)* é reajustado.

$$\varphi(v) = \begin{cases} LMax\ se\ v \geq LMax, \\ LMin\ se\ v \leq LMin \\ v\ se\ LMax \geq v \geq LMin \end{cases} \qquad \text{[Eq. 04]}$$

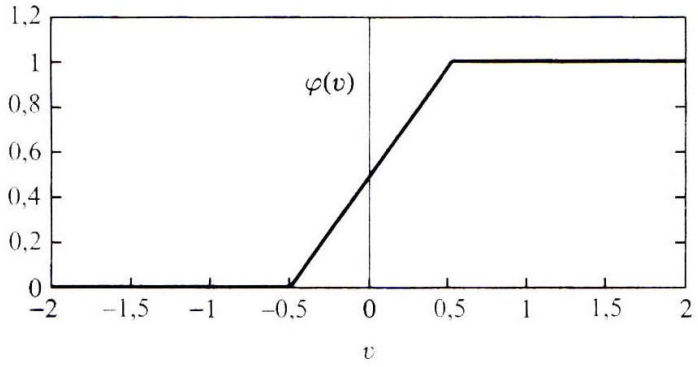

Figura 2 – Função de Ativação Linear

2.4. Processo de Aprendizagem

A aprendizagem é a etapa onde a RNA tem os seus pesos ajustados através do algoritmo de aprendizagem. Esse ajuste pode ser feito, basicamente, de duas maneiras: aprendizado supervisionado e aprendizado não supervisionado. O ajuste de peso do LMS é demonstrado pela [Eq. 05].

O LMS é um algoritmo de aprendizagem supervisionada, ou seja, ele conhece a saída que se deseja para o problema proposto e a partir dessa saída desejada ele faz uma comparação com a saída que obteve e verifica se está realmente aprendendo. Para esta verificação utiliza-se a Soma dos Erros Quadrados (SEQ) demonstrada na [Eq. 06]. Quanto mais próximo de zero estiver a SEQ mais próximo está a RNA da solução ideal. Nem todo problema exige uma solução exata. É possível que solução ideal esteja dentro de uma determinada faixa na reta dos números reais. Por isso existe, no LMS desenvolvido, uma variável precisão utilizada para verificar se o SEQ está dentro de uma faixa que vai de zero até o valor dessa variável, ou seja, não é necessário que o SEQ seja exatamente zero.

$$w(n+1) = w(n) + \eta x(n) e(n) \qquad \text{[Eq. 05]}$$

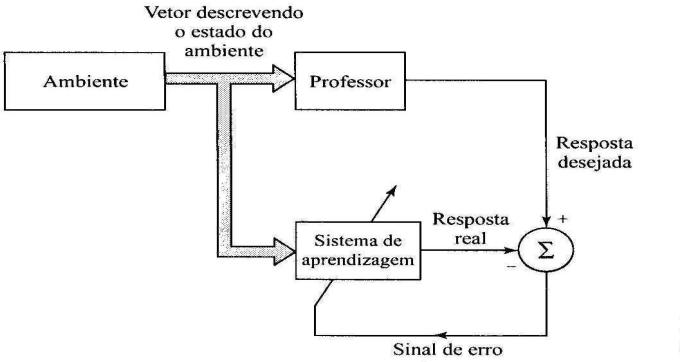

Figura 3 – Diagrama de aprendizagem com professor

2.4.1. Processo de Aprendizagem com Professor

Funciona ajustando parâmetros da rede, de forma a encontrar uma ligação entre os pares entrada e saída fornecidos. É o tipo de aprendizagem mais amplamente utilizado. Um professor indica explicitamente se as saídas dos neurônios estão adequadas. A rede tem sua saída comparada com a saída desejada gerando o sinal de erro da sua resposta atual, que será utilizado no ajuste dos pesos conforme mostrado na equação [Eq. 08]. A SEQ [Eq. 07] de todas as saídas é utilizada como medidor de desempenho da rede e como condição de parada se o resultado chegar a zero. O erro, *e*, é a diferença entre a saída desejada e a saída obtida [Eq. 06], onde a saída obtida y_k é a soma ponderada das entradas pelos pesos [Eq. 01], somada ao bias b_k [Eq. 02], passadas pela função de ativação [Eq. 03].

$$e(j) = d(j) - y(j) \qquad \text{[Eq. 06]}$$

$$md = \sum e^2 \qquad \text{[Eq. 07]}$$

$$w_k(j+1) = w_{kj} + \eta e_j x_{kj} \qquad \text{[Eq. 08]}$$

Onde w_{kj} é o peso atual somado ao produto da entrada x_{kj} pela taxa de aprendizado η e o erro.

3. PROGRAMA

O motivo para o qual foi desenvolvimento este aplicativo ocorreu da necessidade de visualizar o funcionamento de um perceptron de camada única usando o algoritmo de LMS, do modo como ocorre o aprendizado do neurônio, da influência que cada parâmetro tem nesta fase, para facilitar a compreensão desta técnica da inteligência artificial.

Objetivou-se com este programa a obtenção de uma ferramenta capaz de auxiliar o ensino de redes neurais artificiais, para tanto, criou-se uma interface gráfica de forma a tornar simples ao usuário a visualização do processo de aprendizagem, a influência de cada fator nesta etapa, e a validação do conhecimento de um neurônio artificial.

O aplicativo dividiu-se basicamente em duas fases, são elas: fase de aprendizagem e fase de reconhecimento. Na fase de treinamento, o usuário só precisa indicar o número de iterações desejadas, o coeficiente do fator de correção e o número de pontos necessários para o exemplo atual, é muito importante lembra que, para uma melhor precisão do sistema é viável que o valor do campo *Exemplos* seja maior que dois. O objetivo do aplicativo é gerar uma equação de uma reta na quarta dimensão que se aproxima, com um determinado grau de precisão, da equação desejada. Ao acionarmos ação do Butão *Reconhecimento* o sistema entra na fase de reconhecimento Figura 5, nessa fase o usuário poderá confirmar a precisão do aplicativo, pois o mesmo disponibiliza uma interface gráfica desta fase, que captura os pesos emulados,

então o usuário poderá indicar entradas do neurônio e verificar se o resultado da equação realmente condiz com o esperado.

3.1. Interface Gráfica

O aplicativo foi implementado em Java, com uma interface amigável e intuitiva. Através da interface todos os dados do programa podem ser configurados exceto os pontos do problema, pois esses são escolhidos randomicamente, o sistema guarda ainda resultados anteriores para uma analise de dados futuro. O usuário pode, também, acompanhar o desenvolvimento do aprendizado através da aba saída.

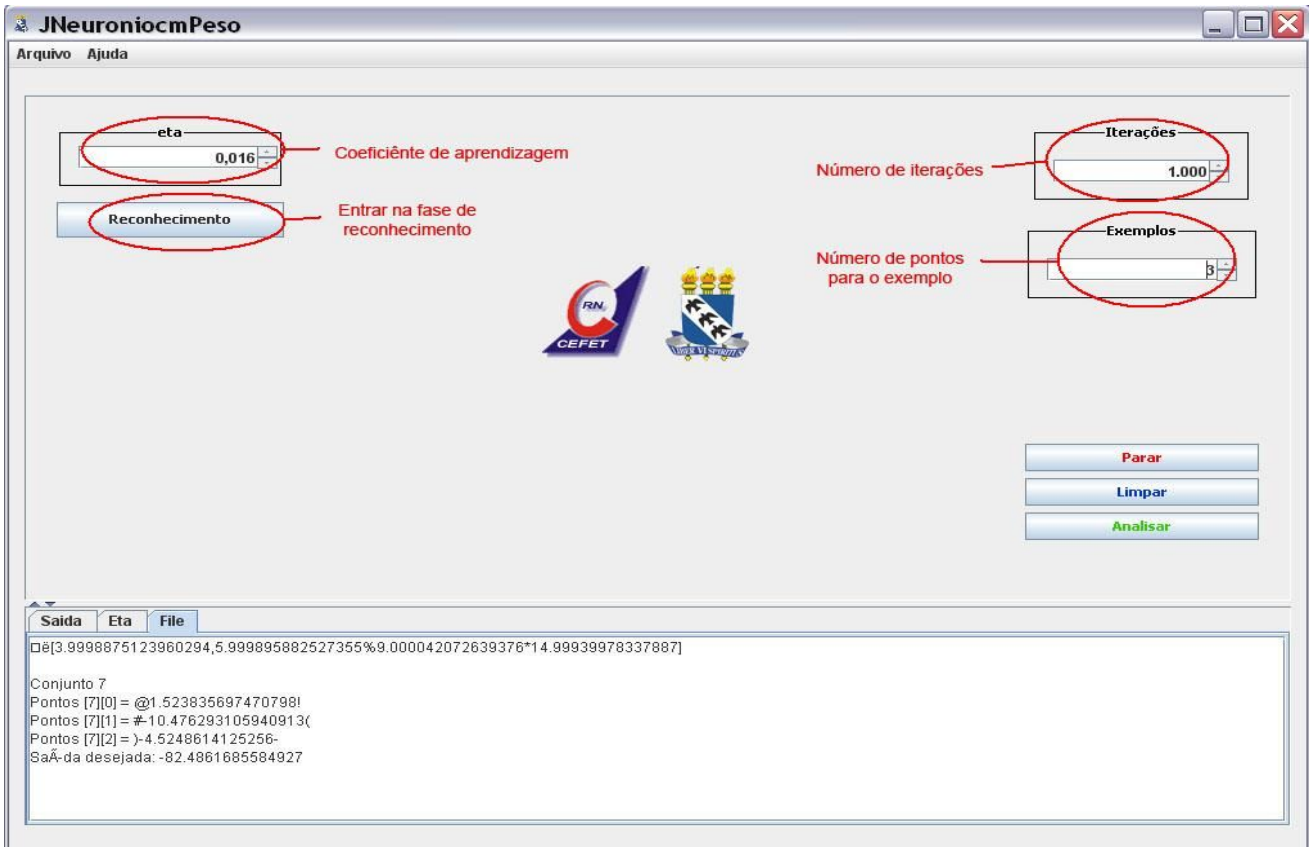

Figura 4 – Fase de treinamento

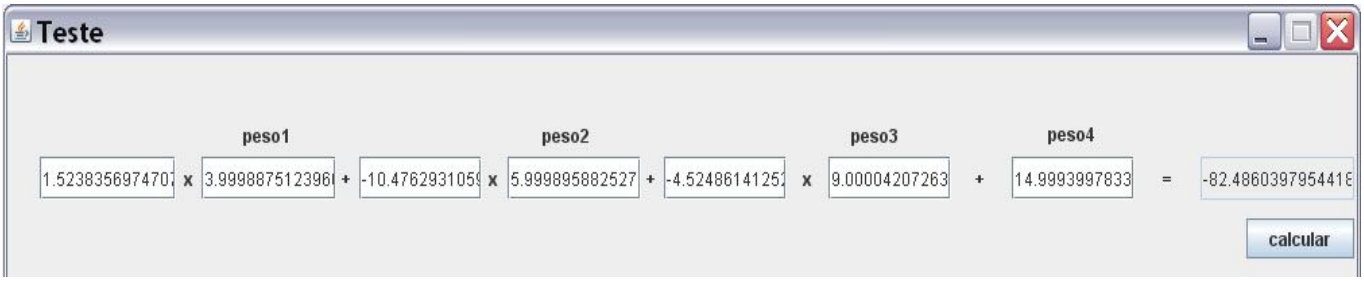

Figura 5 – Equação emulada pelo sistema (Fase de reconhecimento)

3.2. Configurando as Entradas

Como entrada, o sistema necessita apenas do número de iterações o coeficiente de aprendizagem e o número de pontos para os exemplos. Relembrando que é muito importante para o funcionamento ideal que o valor desse campo seja maior que dois. O motivo para isso é que, na quarta dimensão um ou dois pontos passam infinitas retas, logo o sistema possivelmente não acertará a equação esperada.

3.3. Saídas

Nas áreas de texto estão disponíveis três tipos de saídas distintas, cada saída em sua aba própria determinada. A principal aba, denominada *Saída*, mostra todo o calculo para o ajuste de pesos do neurônio. A aba *ETA* está disponível todas a iterações necessárias e o valor do eta(*e*) na respectiva iteração. Na aba *File* é mostrado o que será persistido, esses valores são: resultado dos pesos emulados, pontos gerados randomicamente pelo sistema e a saída desejada pelo sistema.

3.4. Persistência

A persistência do aplicativo é feito em modo texto, o aplicativo armazena todas as informações da aba *File*. Isso significa que essas informações podem ser restauradas e o treinamento atual pode ser salvo para uso futuro.

4. Conclusão

Uma grande dificuldade durante a carreira acadêmica é o desacoplamento da prática e a teoria. Este trabalho tem com objetivo esta integração, auxiliando assim a compreensão do funcionamento de um perceptron de única camada utilizando o algoritmo LMS. Ao acompanharmos o desenvolvimento do aplicativo varias duvidas são desvendadas e alguns conceitos antes obscuros, foram esclarecidos. E o melhor de tudo é poder acompanhar de perto uma utilização pratica do conhecimento adquirido, evitando passar em branco por uma área tão interessante como RNA.

REFERÊNCIAS

HAYKIN, S. **Redes Neurais.** 2. ed. São Paulo: Bookman, 2001.

BRAGA, A.P.; CARVALHO, A.C.; LUDERMIR, T.B. **Redes Neurais Artifciais.** 1. ed. Rio de Janieiro: LTC, 2000.

C.A.G. Fonseca, F.M.U. de Araújo, A.V. Medeiros, e A.L. Maitelli, “Algoritmos Genéticos para Otimização de um Controlador Nebuloso para Supressão de Vibrações”, Anais do VI SBAI, Bauru, São Paulo, 2003, pp. 959-963.

A.P.L.F. Carvalho **Redes Neurais Artificiais.** Disponível em:
< http://www.icmc.usp.br/~andre/research/neural/index.htm> Acesso em: 09 ago2008.

AGRADECIMENTOS

Deixamos expressos nossos sinceros agradecimentos às seguintes instituições e pessoas, sem as quais o presente trabalho teria sido impossível:

- Universidade do Estado do Rio Grande do Norte, no apoio técnico e a disponibilidade da sua estrutura física para o desenvolvimento deste trabalho.

- Centro Federal de Educação Tecnológica do Rio Grande do Norte, no apoio técnico e incentivo no desenvolvimento.

- Nossos amigos, Flávio Herique e Waldney Andrade, que nos ajudou bastante no apoio ao desenvolvimento do aplicativo.