Porta Lógica XOR Digital Neural

Eduardo Vila Real Mendes
Rubens de Melo Marinho Jr

Departamento de Física
Instituto Tecnológico de Aeronáutica
Brasil

E-mail: ita.eduardo@gmail.com
marinho@ita.br

Resumo. O presente trabalho consiste em implementar a porta lógica XOR com a técnica de redes neurais artificiais utilizando o algoritmo de retropropagação

1. Introdução

Redes neurais tem sido usadas nas mais diversas áreas do conhecimento. A rede neural nada mais é do que a simulação do funcionamento do cérebro humano na resolução de problemas. Recentemente progressos significativos tem sido obtidos na aplicação de redes neurais em processamento de sinais. Filtros baseados em redes neurais multicamadas tem sido usados com exito. Como uma etapa preliminar na construção de um filtro digital neural com o objetivo de tratar dados de detectores de ondas gravitacionais, na busca de sinais imersos em ruido, foi tratado um problema mais simples para o perfeito entendimento do funcionamento de uma rede neural. Utilizamos esta técnica para implementar a porta lógica XOR.

Foram estudados os conceitos envolvendo as redes neurais, e em particular o caso da rede neural com multicamadas, com o método da retropropagação ou backpropagation. A bibliografia básica para o desenvolvimento deste trabalho foi o livro do Kovacs[1]. Este livro permitiu desde o aprendizado básico, bem como algoritmos avançados de implementação de uma rede neural. Ainda, na segunda parte do livro, existem alguns assuntos relacionados com a aplicação de redes neurais, em particular, o tópico: Redes Neurais em Processamento de Sinais. Este tópico está diretamente relacionado com o nosso próximo objetivo que consiste em processar sinais de ondas gravitacionais.

Foi desenvolvida uma rotina para a criação de rede neural, através do método de backpropagation, que tem como parâmetros de entrada o número de camadas, número de neurônios em cada camada, o vetor de entrada, enfim, uma rotina geral para a criação de uma rede neural.

Como problema protótipo para o desenvolvimento de um programa de rede neural, foi utilizada a função ou exclusivo, que deve retornar verdadeiro se, e somente se, os dois bits de entrada forem iguais. Com o desenvolvimento deste programa, o passo seguinte foi a generalização da solução particular da função ou exclusivo, para uma rede neural com um número de camadas variável, com números de neurônios em cada camada também variável.

Por fim, com a elaboração deste último programa, foi criada uma rotina no MatLab, para a chamada desta função de rede neural, permitindo maior praticidade no seu uso.

Este artigo será organizado da seguinte maneira: na seção (2) será dada uma conceituação das redes neurais artificiais e suas aplicações; na seção (3) apresentaremos o algoritmo de retropropagação para um caso particular; na seção (4) foi generalizado o algoritmo da rede neural para um número qualqer de neurônios de camadas; na seção (5) apresentaremos a forma de implementar um filtro neural digital para o trabalho de detecção do sinal de ondas gravitacionais imersas em ruído e na ultima seção apresentamos as conclusões.

2. Redes Neurais

Para uma conceituação de uma rede neural, um conceito que não pode ser esquecido é o neurônio biológico, cuja função nos seres humanos serviu de base para a montagem e elaboração da rede neural. O neurônio, como outra célula de nosso organismo, possui uma membrana celular, que além das funções semelhantes às outras células, possui grande relevância para o funcionamento elétrico da célula nervosa. Além do corpo celular, cada neurônio possui dendritos e axônio. Através

desses últimos filamentos é possível estabelecer uma ligação entre os neurônios, e assim, servir para o transporte de estímulos nervosos.

A partir do século XIX, as propriedades elétricas dessas células começaram a ser observada, e assim, começaram os estudos a respeito desse fenômeno. Observou-se as propriedades dos neurônios, onde era possível haver mais de uma entrada, que ocorrem através das ligações sinápticas, e apenas uma saída. Portanto, cada neurônio deveria absorver cada informação passada a esta célula, processar e emitir uma determinada resposta. Assim, através desse modelo simples, foi possível dar início ao desenvolvimento das redes neurais artificiais.

A partir do raciocínio anterior é possível estabelecer uma função de ativação do neurônio:

$$f_t = g\left(\int_i \alpha_i(t)x_i(t)dt\right), \tag{1}$$

onde α_i representam os ganhos sinápticos e os $x_i(i)$ as informações que chegam aos neurônios. Esta função pode possuir um comportamento de função degrau, ou seja, em um determinado intervalo esta pode possuir um determinado valor (pode ser constante ou não), e em outro intervalo esta pode assumir um valor distinto (pode ser constante ou não).

Uma evolução no sistema de rede neural foi possível com o neurônio booleano de McCulloch. Partindo da função de ativação (1), o neurônio de McCulloch pode ser entendido, através de fórmulas, caso as entradas e saídas sejam binárias, isto é, assume valor unitário caso seja verdadeira e valor nulo caso seja falsa (adota-se essa convenção neste trabalho). Dessa forma, em cada neurônio deve haver uma função, similar à equação (1), e o valor obtido deve ser comparado a um determinado valor limiar. Caso esta seja superior, a resposta é um, e caso contrário zero, por exemplo. Em termos matemáticos, pode-se ter a seguinte formulação:

$$y = H\left(\sum_{i=1}^{n} w_i x_i - \Theta\right) = H\left(w^t x - \Theta\right) \longmapsto y \in [0;1], \tag{2}$$

onde o vetor w contém as componentes dos ganhos associadas às entradas x_i, Θ é o valor do limiar, $H(v)$ é a função degrau unitário.

Um aperfeiçoamento deste método de rede neural já visto, pode ser a incorporação de mais de uma camada de neurônio. Os neurônios que recebem diretamente as informações de entrada constituem a camada de entrada. Os neurônios que recebem as respostas da primeira camada constituem a segunda camada e assim esta denominação é levada até a última camada. Em geral, as camadas intermediárias (exceto a primeira e última) são chamadas de camadas ocultas.

Outro fator de grande relevância para uma rede neural está no aprendizado e treinamento da mesma. A seguir, discutiremos este ponto.

Agora, será discutido melhor o algoritmo de retropropagação, ou backpropagation. Considere a função de ativação:

$$y = g\left(\sum_{i=1}^{n} w_i x_i\right). \tag{3}$$

Para treiná-la, será aplicado o método do gradiente conjugado. Assim, deverá ser encontrado um vetor de parâmetros que minimize o erro quadrático sobre um dado conjunto de treinamento. Considere o erro quadrático para um discriminador arbitrário de parâmetros w:

$$E(w) = \sum_{l=1}^{L} \left(g\left(w^t x_l^d\right) - y_l^d\right)^2. \tag{4}$$

Calculando então seu gradiente, tem-se o seguinte resultado:

$$\frac{dE(w)}{dw} = -2\sum_{l=1}^{L} \delta_l x_l^d, \tag{5}$$

onde

$$\delta_l = (y_l^d - y_l)\frac{dg(v)}{dv}.$$

O parâmetro w é atualizado da seguinte maneira:

$$w(k+1) = w(k) + \Delta w(k). \tag{6}$$

Então tem-se que:

$$\Delta w(k) = -\eta\frac{dE(w)}{dw}, \tag{7}$$

onde η determina o passo que se dará naquela direção. Substituindo (7) e (5) em (6), tem-se que:

$$w(k+1) = w(k) + 2\eta \sum_{l=1}^{L} \delta_l^d x_l^d. \tag{8}$$

Assim, em linhas gerais, é realizado o treinamento de uma rede neural através do backpropagation.

3. Realização da função XOR através da rede neural

Seguindo com o estudo para a implementação de uma rede neural, a elaboração de um algoritmo, ou código fonte para a implementação de uma rede neural, com o método do backpropagation, foi a primeira dificuldade prática neste projeto. Até então, houve apenas discussões teóricas e conceituais englobando o método do backpropagation nas redes neurais, e nessa etapa do projeto começou-se a utilizar o software MatLab.

Conforme já justificado anteriormente, a proposta inicial seria utilizar as rotinas do pacote de rede neural do próprio MatLab, entretanto, o projeto sofreu alterações, onde estas rotinas não seriam mais utilizadas e sim, a rede neural seria desenvolvida no MatLab. Portanto, a implementação da função XOR representou um importante passo no desenvolvimento de um algoritmo, e conseqüentemente, uma rotina para uma rede neural geral no MatLab.

A função XOR consiste basicamente em uma função de duas entrada, sendo cada uma delas binárias, e apenas uma saída. A lógica desta função é mostrada na tabela I abaixo. Para que esta função pudesse ser implementada através de

Tabela 1. Tabela verdade da função XOR

Entrada I	Entrada II	Saída
1	1	0
1	0	1
0	1	1
0	0	0

redes neurais, através do backpropagation, foi adotado o seguinte algoritmo, que se encontra no livro do Kovács[1], na página 67, que é aplicado para implementar uma rede neural seguindo o método do backpropagation, mostrado nas linhas posteriores. Seja dado um conjunto de treinamento de L exemplos:

$$\Psi = (x_l^d, y_l^d)_{l=1}^{L}.$$

Uma rede de M camadas com $\{J0, J1, \ldots, JM\}$ elementos em cada camada é treinada sobre Ψ escolhendo, primeiro, um conjunto de parâmetros iniciais $W(0)$, e uma taxa de aprendizado η . Em seguida, estabele-se um critério onde o processo de iterações será interrompido, que pode ser feito limitando o número de iterações ou ainda indicando o erro máximo. Após a escolha desse critério, deve-se calcular os erros de saída, da camada de saída, bem como das camadas subseqüentes: com $l = 1, 2, \ldots \quad L$;

$$\delta_{y(JM)l} = \left(y_{jMl}^d - y_{jMl}\right) \tag{9}$$

e

$$\delta_{JMl} = \delta_{y(JM)l} \frac{dg(v_{jM})}{dv_{jM}} \tag{10}$$

sendo m esteja variando de: $m = (M-1), \ldots, 1, 0$:

$$\delta_{jml} = \left(\sum_{j(m+1)-1}^{J(m+1)} \delta_{j(m+1),l} w_{m+1,j(m+1),jm} \right) \frac{dg(h_{jm})}{dh_{jm}}. \tag{11}$$

O passo seguinte consiste em atualizar os parâmetros, w. Este processo é realizado da seguinte maneira:

$$\Delta w_{m,jm,j(m-1)} = 2\eta (\sum_{l=1}^{L} \delta_{jm,l} u_{j(m-1),l}^d),$$

notando que $u_{j(0),l}^d = x_{j(0),l}^d$.

Por fim, para a primeira iteração, calcula-se $w(1) = w(0) + \Delta w$, e calcula-se os erros de saída, da camada de entrada e das camadas subseqüentes novamente. Este processo deve ser repetido até que o critério de parada seja satisfeito. Seguindo este algoritmo, em seguida será mostrado o código fonte do programa, em MatLab, para a solução da função XOR, através do método do backpropagation.

```
% Implementacao do xor com uma rede de neuronios com 3 neuronios
% sendo 2 na camada de entrada e 1 na camada de saida
% Usaremos a notacao do Kovacs capitulo 5.

clear

%
% A funcao de ativacao sera a tanh
%

%
% Taxa de aprendizado
%

eta = 0.1; % Parametro que determina o tamanho do passo na direcao oposta ao gradiente
M = 2; % Numero de camadas
Jm = [2 1]; % Numero de neuronios em cada camada
En = [2 2]; % Numero de entradas em cada camada

% O primeiro indice de W é o numero de camadas
% O segundo indice de W designa o neuronio da camada indicada no primeiro indice
% O terceiro indice de W e o indice da componente do vetor de entrada no neuronio
% Veja Kovacs capitulo 5 pag 72

%
% Inicializa a matriz dos pesos com numero aleatorios
%

for m = 1 : M
    W{m} = rand ( Jm(m), En(m) );
end

%
%  Treinamento da rede para um conjunto com L dados
%

Xd = rand (2,10000);
id = find( Xd¿0.5);
Xd(id) = 1;

id = find( Xd¡=0.5);
Xd(id) = 0;

Yd = Xd(1,:) + Xd(2,:);

id = find( Yd==2);
Yd(id)=0;

%Xd = [1 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 1 1 0 1 1 0 0 0 0 1 1 1                    ; % Dados de entrada
%         0 1 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0 0 1 1
%                 ]; % Cada coluna é um conjunto
% Yd = [1 1 0 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 0]; % Dados de saida

L = length( Xd );
for l = 1 : L % Treina a rede com cada amostra l
    X = Xd(  :  , l ); % X é o U0  Nao foi usado assim porque nao existe indice 0 no matlab
    V{1} = W{1} * X; % Se pudesse seria V1 = W1'*U0
    U{1} = tanh(V{1});
    for m = 2 : M % Calcula todas as saidas da rede
        V{m} = W{m} * U{m-1};
```

```
        U{m} = tanh ( V{m} );
    end

    Y = U{M}; % UM sao as saidas Y
    E = (Y - Yd(:,l))'*(Y - Yd(:,l)); % Erro para uma dada realizacao da rede com pesos W

    delta{M} = (Yd(:,l)-Y).*(sech(V{M}).^2);
    W{M} = W{M} + 2*eta*delta{M}*U{M-1}';
    for m =   M -1 : -1 : 1      % verificar se nao vai ate 0
        delta{m}=W{m+1}'*delta{m+1}.*(sech( V{m} ).^2);
        if m == 1
            W{m} = W{m} + 2*eta*delta{m}*X';
       else
            W{m} = W{m} + 2*eta*delta{m}*U{m-1}';
       end
    end
end

%
% Simulacao
%

X = [1; 1]
U{1} = tanh( W{1}*X );
for m = 2 : M
    U{m} = tanh( W{m}*U{m-1} );
end
U{M}
```

Neste código fonte, pode-se notar que a taxa de aprendizado eta foi colocada com um valor de 0.1. Este valor foi obtido com várias tentativas para a saída da função XOR. Com esta taxa de aprendizado, obtém-se o seguinte resultado para a função XOR, mostrada na tabela 2. Como se pôde observar com os resultados, dentro dos limites de exigência, os

Tabela 2. Resultados obtidos para a função XOR, com o código fonte anterior.

Entrada I	Entrada II	Saída
1	1	-0.0041
1	0	0.9327
0	1	0.9393
0	0	0

resultados foram bastante satisfatórios. Estes resultados podem ser, cada vez mais próximos da resposta desejada, caso seja aumentado o conjunto de treinamento.

4. Rotina da Rede Neural usando o backpropagation

Partindo da função XOR descrita anteriormente, o passo seguinte no desenvolvimento de uma rede neural, em geral, foi generalizar o número de neurônios em cada camada, bem como o número de camadas de uma rede neural. Para melhor análise deste algoritmo, a seguir, será mostrado o código fonte.

```
function Return = redeneural2 (eta, M, Jm, En, X)
% Implementacao do xor com uma rede de neuronios com 3 neuronios
% sendo 2 na camada de entrada e 1 na camada de saida
% Usaremos a notacao do Kovacs capitulo 5.

%
% A funcao de ativacao sera a tanh
%
```

```
%
% Taxa de aprendizado
%

% eta = 0.1; % Parametro que determina o tamanho do passo na direcao oposta ao gradiente
% M = 2; % Numero de camadas
% Jm = [2 1]; % Numero de neuronios em cada camada
% En = [2 2]; % Numero de entradas em cada camada

% O primeiro indice de W é o numero de camadas
% O segundo indice de W designa o neuronio da camada indicada no primeiro indice
% O terceiro indice de W e o indice da componente do vetor de entrada no neuronio
% Veja Kovacs capitulo 5 pag 72

%
% Inicializa a matriz dos pesos com numero aleatorios
%

for m = 1 : M
    W{m} = rand ( Jm(m), En(m) );
end

%
%  Treinamento da rede para um conjunto com L dados
%

Xd = rand (2,10000);
id = find( Xd¿0.5);
Xd(id) = 1;

id = find( Xd¡=0.5);
Xd(id) = 0;

Yd = Xd(1,:) + Xd(2,:);

id = find( Yd==2);
Yd(id)=0;

%Xd = [1 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 1 1 0 1 1 0 0 0 0 1 1 1                    ; % Dados de entrada
%         0 1 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 1 0 0 1 1 0 1 0 0 0 1 1
%                 ]; % Cada coluna é um conjunto
% Yd = [1 1 0 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 0]; % Dados de saida

L = length( Xd );
for l = 1 : L % Treina a rede com cada amostra l
    X = Xd(  :  , l ); % X é o U{0}  Nao foi usado assim porque nao existe indice 0 no matlab
    V{1} = W{1} * X; % Se pudesse seria V{1} = W{1}'*U{0}
    U{1} = tanh(V{1});
    for m = 2 : M % Calcula todas as saidas da rede
        V{m} = W{m} * U{m-1};
        U{m} = tanh ( V{m} );
    end

    Y = U{M}; % U{M} sao as saidas Y
    E = (Y - Yd(:,l))'*(Y - Yd(:,l)); % Erro para uma dada realizacao da rede com pesos W

    delta{M} = (Yd(:,l)-Y).*(sech(V{M}).^2);
    W{M} = W{M} + 2*eta*delta{M}*U{M-1}';
    for m =   M -1 : -1 : 1      % verificar se nao vai ate 0
        delta{m}=W{m+1}'*delta{m+1}.*(sech( V{m} ).^2);
        if m == 1
```

```
            W{m} = W{m} + 2*eta*delta{m}*X';
        else
            W{m} = W{m} + 2*eta*delta{m}*U{m-1}';
        end
    end
end

%
% Simulacao
%

% X = [0; 1]
U{1} = tanh( W{1}*X );
for m = 2 : M
    U{m} = tanh( W{m}*U{m-1} );
end
U{M};
return = U{M}
```

Conforme pode ser visto no código acima, foi criada uma rotina correspondente a função redeneural2 que recebe como argumentos a taxa de aprendizado, o número de camadas, dois vetores linha, significando, respectivamente, o número de neurônios em cada camada, e o número de entradas em cada camada. O último argumento é o vetor de entrada.

Verificando os resultados obtidos, para esta rotina gerada para simular novamente a função xor, pode-se notar que os resultados estão próximos do esperado. Ainda, simulando esta rotina com os mesmos argumentos mais de uma vez, pode-se verificar que as saídas não são exatamente iguais. Por exemplo, tomando como argumentos, a taxa de aprendizado com valor de 0.1, número de camadas igual a 2, número de neurônios em cada camada [2 1], [2 2] sendo o número de entradas em cada camada, e [0; 1] é o vetor de entrada. Simulando esta função três vezes, por exemplo, chegou-se aos valores 0.9362, 0.9414 e 0.9382.

5. Redes Neurais em Processamento de Sinais

No estudo da aplicação das redes neurais no processamento de sinais, primeiro será analisado, do ponto de vista teórico, os sinais vistos em engenharia e na física. Matematicamente, sinais são funções ou uma seqüência que transportam informações de uma fonte de mensagens para um destinatário. Os sinais, ainda, dependem do canal de comunicações que é definido pelo tipo de distorção que introduz neles. Existem três formas fundamentais de distorção, a deterministica linear que ocorre por exemplo na limitação de banda de freqüência de sinais; a deterministica não linear que ocorre por exemplo nas saturações; a aleatória que ocorre por exemplo na presença de ruídos, que será o objeto de análise para projeto do filtro digital neural.

Com a utilização das técnicas de redes neurais pode-se realizar a classificação desses sinais. Este método consiste na elaboração de um treinamento para a rede neural de multicamadas com entrada de N amostras das séries temporais associadas a cada um dos sinais, e como saída as M categorias em que se pretende classificá-los. Assim, a técnica de implementação de uma rede neural deve estabelecer, em suma, as seguintes metas:

- O números de parâmetros e o tipo destes na determinação de um sinal pode resultar em diferentes desempenhos da rede. Por exemplo, para classificação de sinais uma rede neural deve-se utilizar parâmetros que permitam, com maior facilidade, a identificação de um tipo de sinal, e que o número de informações a serem passadas para a rede neural seja a menor possível, para que o rendimento seja o maior possível, em termos de recursos computacionais e resposta desejável como saída da rede neural.

- Deve-se escolher um conjunto de sinais representativos para o treinamento do classificador. Isto porque quanto melhor for a escolha resulta em um melhor desempenho.

- Este conjunto de sinais de treinamento é processado para gerar o conjunto de parâmetros definidos no primeiro item abordado aqui. Este será o conjunto de treinamento para a rede de multicamada.

- Opta-se por uma arquitetura de rede: adaline, descriminadores, RHW ou FBR, ou uma combinação destas. Avalia-se a dimensão das camadas.

- Treina-se a rede.

- Avalia-se o seu desempenho com um outro conjunto independente.

6. Conclusões

O trabalho realizado foi determinante para o projeto Filtro Digital Neural. Nesta etapa do plano de trabalho, foi desenvolvido um algoritmo geral para uma rede neural no qual, através do MatLab, foi desenvolvida uma rotina cujos parâmetros da Rede Neural são a taxa de aprendizado, número de camadas, número de neurônios em cada camada, número de entradas em cada camada e o vetor de entrada.

Neste projeto, primeiro foi realizada uma análise sobre as redes neurais, desde sua criação até os métodos para a sua implementação através de algoritmos. Dessa forma, foram analisados os processos de aprendizado e treinamento através do método do backpropagation.

Para melhor entendimento dos algoritmos de redes neurais foi implementada através do MatLab a função XOR, por meio do método do backpropagation inicialmente para um caso particular e em seguida sua generalização. Com este trabalho preliminar estamos agora aptos a implementar o filtro neural digital que será usado para extrair sinal de ondas gravitacionais de ruído.

7. Agradecimentos

Agradeço ao CNPq pelo incentivo à produção científica no Brasil, em particular pelo apoio aos estudantes da graduação do ITA permitindo que estudantes universitários possam ter um contato mais próximo das pesquisas científicas, oferecendo oportunidade de divulgação e publicação dos trabalhos realizados pelos estudantes.

References

[1] KOVÁCS ZL. , "Redes Neurais Artificiais: Fundamentos e Aplicações", 2ª ed. ; Edição Acadêmica; São Paulo; 1996