CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Mestrado em Modelagem Matemática e Computacional

Aula Prática IV

Introdução às RNAs: o Perceptron

Nome: Samara Soares Leal

Disciplina: Inteligência Computacional - MMC

Data: 14 de julho de 2014.

1. Script que implementa as funções ‘E’ e ‘OU’ no Matlab:

```
function [ resp ] = redeneuralAlgoritmo( )
% Criar vetores de entrada - Cada coluna é um xo x1 x2 x3 - Tamanho: [8,4]
x = [1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1];

% valores de saida desejados [8,2]
yd=[0 0; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 1 1];

% Valores da matriz w [4,2]
% w = -0.5 + rand(4,2);
wteste = [-0.4139    0.3238;
          0.0905    0.0015;
          0.1767   -0.1842;
          0.2691    0.2395];

% Saída: Multiplicar uma matriz x[8,4]por w[4,2] gerando y[8,2]
y= (x*wteste) > 0;

e=yd-y;
count=0;
while norm(e)~=0
   % Atualização de w= [4,8]*e[8,2] + w[4,2]
   wteste = wteste + 0.1*x'*e;
   % Novo y
   y= (x*wteste) > 0;
   % Atualizção do erro
   e=yd-y;
   count=count+1;
end
```

Saída:

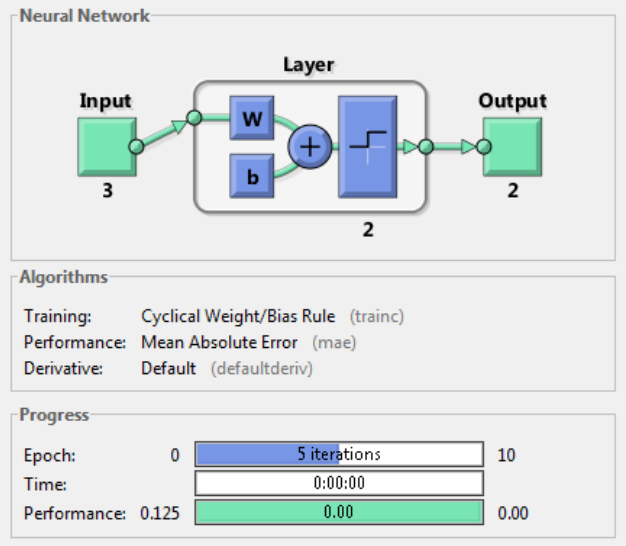
```
>> redeneuralAlgoritmo

y =

     0     0
     0     1
     0     1
     0     1
     0     1
     0     1
     0     1
     1     1


count =

     8
```

Número de épocas = 8

2. Implementação da função ‘E’ e ’OU’ usando a *‘toolbox’* de RNA do Matlab.

```
function [ resp ] = redeneuralToolBox( )

    x = [0 0 0; 0 0 1; 0 1 0; 0 1 1; 1 0 0; 1 0 1; 1 1 0; 1 1 1]';
    yd = [0 0; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 1 1]';
    wb= [-0.4139 0.3238]';
    w = [ 0.0905    0.0015;
          0.1767   -0.1842;
          0.2691    0.2395]';
    % Criar a rede
    net = newp(x,yd);

    % Definir taxa de aprendizado
    net.trainParam.lr = 0.1;

    % Pesos da rede
    net.b{1,1}=wb;
    net.IW{1,1}=w;

    % Parâmetros de treinamento
    net.trainParam.epochs=10;
    net.trainParam.goal=0;

    % Treinar a rede
    net = train(net,x,yd);

    % Simular a rede
    y = sim(net, x)';
```

Saída do treinamento:



Saída da simulação:

```
y =

     0     0
     0     1
     0     1
     0     1
     0     1
     0     1
     0     1
     1     1
```

Número de épocas: 5

3. Discussão dos resultados encontrados

Inicializando-se a matriz de pesos com valores aleatórios entre [-0.5,0.5], o número de épocas do algoritmo é diferente do número de épocas da toolbox, devido a inicialização aleatória dos pesos ser diferente a cada execução.

Ao fixar a matriz dos pesos em ambos os scripts, utilizar a mesma taxa de aprendizado e mesmos valores de bias e erro, pode-se observar que em algumas execuções o número de épocas do script é igual ao número de épocas da toolbox, porém isto acontece poucas vezes, em geral, o número de épocas é diferente, mas bem próximo.

Este comportamento pode ser observado na tabela abaixo. Os valores de **saída** de bias e pesos do script (algoritmo) e da toolbox são diferentes, porém são próximos. Isto no leva a pensar, que há uma pequena diferença entre a implementação da regra delta feita pela toolbox e o script feito a mão.

	Algoritmo Redes Neurais	**Toolbox Redes Neurais**
Bias Inicial	[-0.4139 0.3238]	[-0.4139 0.3238]
Bias Final	[-0.5139 -0.0762]	[-2.4139 -0.6762]
Pesos Iniciais	[0.0905 0.0015; 0.1767 -0.1842; 0.2691 0.2395]	[0.0905 0.0015; 0.1767 -0.1842; 0.2691 0.2395]
Pesos Finais	[0.1905 0.1015; 0.1767 0.1158; 0.1691 0.2395]	[2.0905 1.0015; 0.1767 0.8158; 0.2691 1.2395]

Ao analisar passo a passo o código através da opção ‘debug’, pode-se observar que da segunda iteração em diante a matriz de peso gerada pela toolbox sofre algumas alterações (pequenas), que não são as mesmas realizadas na matriz de peso do script feito a mão. Para realizar estas alterações de pesos há uma estrutura complexa implementada na toolbox, que dificulta a obtenção de uma relação exata desta diferença na regra delta, até mesmo através do ‘debug’.