Neurônio de McCulloch-Pitts

Erick Amorim Fernandes, 86031
ELT 451 – Inteligência Computacional
Departamento de Engenharia Elétrica, Universidade Federal de Viçosa, Viçosa - MG
erick.fernandes@ufv.br

Resumo – A prática consiste na reprodução do neurônio de McCulloch-Pitts e na utilização desse neurônio para emular algumas funções logicas. Por fim, serão feitas análises de crédito baseadas no treinamento do neurônio estudado.

I. INTRODUÇÃO

As Redes Neurais Artificiais (RNA's) se baseiam em técnicas computacionais inspiradas na estrutura neural de organismos inteligentes através de modelagem matemática. Uma grande rede neural artificial pode ter centenas ou milhares de unidades de processamento enquanto um cérebro de um mamífero pode ter muitos bilhões de neurônios.[1]

O primeiro modelo de redes neurais artificiais foi proposto por Warren S. McCulloch e Walter Pitts em um artigo publicado em 1943: "A logical calculus of the ideas imanente in nervous activity" Bulletin of Mathematical Biophysics.[2]

Os autores fizeram uma analogia entre células nervosas vivas e sua modelagem matemática, em um trabalho publicado sobre "neurônios formais", simulando o comportamento do neurônio natural, no qual o neurônio possuía apenas uma saída e o processamento consistia num modelo de um neurônio biológico, conhecido como neurônio de McCulloch-Pitts.

Observe o esquema na Figura 1:

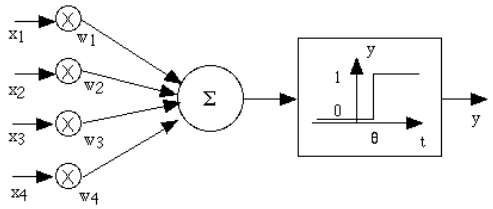

Figura 1 – Modelagem de um neurônio.

II. OBJETIVOS

Essa prática tem como objetivo, o entendimento do modelo de neurônio de *McCulloch-Pitts*, implementação de funções, tais como *"sum"* e *"hardlim"* e treinar um neurônio para análise de crédito.

III. MATERIAIS E MÉTODOS

Utilizou-se o *software Matlab* para o desenvolvimento da prática. Primeiramente, foi desenvolvida uma função para criar um neurônio de McCulloch-Pitts.

Essa função recebe os vetores de entradas, pesos e o potencial de ação (bias) e a partir desses valores retorna a saída calculada. A função utilizada nessa implementação foi "hardlim".

Em sequência, utilizando o neurônio anterior, foi feita uma modificado nos pesos e no bias de tal forma que o mesmo fosse capaz de emular as funções logicas NAND, OR e XOR.

Por fim, utilizando um modelo de tomada de decisão, feito pelo um especialista em análise de crédito, foi criado um neurônio para realizar algumas classificações e em sequência realizar uma análise de crédito para 3 (três) colegas de turma.

Visando diminuir a carga computacional e melhorar a precisão do treinamento, os dados foram normalizados seguindo a regra: valor a ser normalizado menos valor mínimo divido pela diferença entre o máximo e o mínimo de seu grupo.

IV. RESULTADOS E ANÁLISES

Inicialmente foi implementada função referente ao neurônio que apresentasse como saída valores binários em função de três parâmetros: vetor de entradas, vetor de pesos e bias.

Foi utilizada a função *hardlim* que recebia como parâmetros o somatório dos pesos associados à respectiva entrada. Definiu-se o bias como - 1.

Posteriormente foi implementado um algoritmo o qual modificava o valor referente aos pesos, w_i, em função de valores aleatórios considerando, de forma que fossem determinadas as saídas referentes às funções NAND, OR e XOR de duas entradas, sendo obtida, ainda, a reta de separação da função lógica representada.

Na Fig. 2 encontra-se representada a saída referente à função lógica NAND. Os pesos encontrados foram $w = [-0{,}9\ -0{,}9]$ e bias igual a 1. Em relação à porta OR, Fig. 3, os pesos encontrados foram $wi = [0{,}9\ 0{,}9]\ e\ bias\ igual\ a\ -0{,}8$. Para a lógica XOR, Fig. 4, não foi possível obter os pesos associados à saída requerida, uma vez que a saída não é linearmente separável.

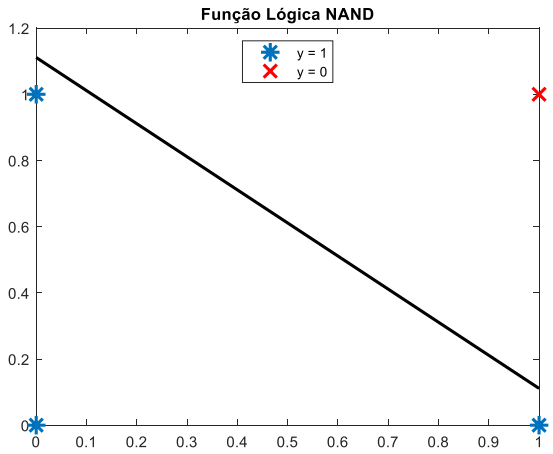

Figura 2. Saída porta lógica NAND

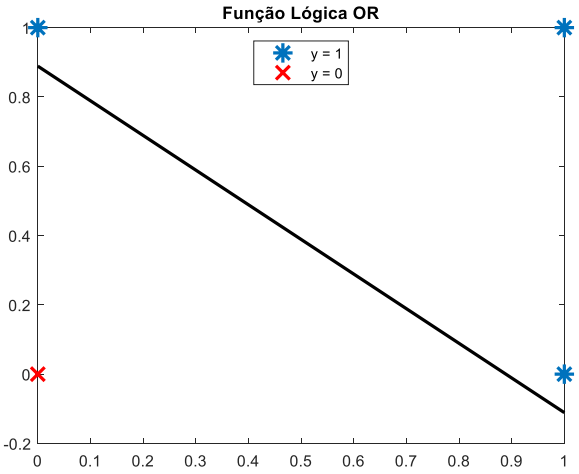

Figura 3- Saída porta lógica OR.

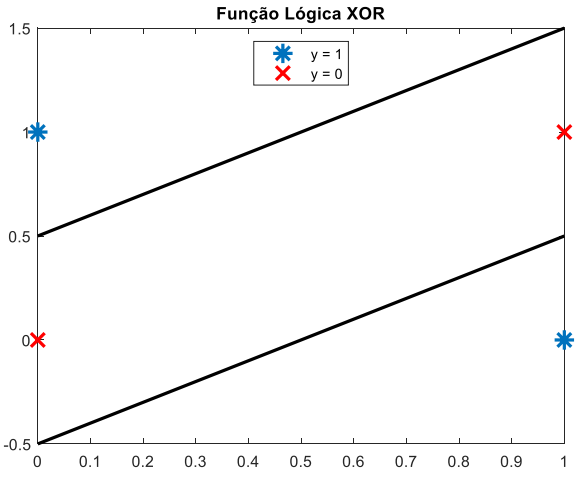

Figura 4- Saída hipotética para porta lógica XOR.

Por fim foi utilizado um banco de dados referente a seis (6) indivíduos, cada um com 7 características, sendo requerida como aprovado e não aprovado. Para tanto, foi utilizado o mesmo neurônio definido anteriormente considerando 7 entradas. Para cada entrada foram definidos patamares binários respectivos à classificação utilizada. Por exemplo, para a classe *sexo*, a subclasse '*masculino'* foi associada a '1' e '*feminino'* a '0'. Posteriormente foi realizado o processo de treinamento segundo n iterações admitindo que os pesos se modificavam segundo valores aleatórios, para obtenção dos pesos ideais a serem obtidos para o neurônio. Para verificar se os pesos obtidos atendiam à condição necessária foi utilizada a função *isequal*.

Por fim, os pesos obtidos foram, $w_i = [-0{,}5890\ 0{,}0929\ 0{,}2751\ -0{,}5103\ 1{,}9138\ -1{,}4054\ 1{,}2448]$.

Para o banco de dados de treinamento foi usado a seguinte entrada:

Tabela 1 - Entradas para análise de crédito.

Indivíduo	A	B	C
Idade	18	30	22
Sexo	0	1	0
Casa	1	1	0
carro	0	1	0
Casado	0	1	1
Filhos	0	3	1
Renda	1100	500	1200

E as saídas após o processamento do neurônio de McCulloch-Pitts foram: Individuo A e C aprovados, enquanto B teve seu pedido negado.

V. CONCLUSÃO

A partir dos resultados obtidos foi possível observar a aplicabilidade das redes neurais (RNA) bem como facilidade quanto à implementação da estrutura utilizada. É notório destacar que é necessário estabelecer critérios que reduzam o tempo de estimação em relação à obtenção dos pesos obtidos no processo de estimação. Foi possível perceber, também, que para casos não linearmente separáveis faz-se necessário o uso de aprimoramento do modelo, que no caso da função lógica XOR poderia ser resolvido ao se utilizar duas retas para classificação.

VI. REFERÊNCIAS BIBLIOGRÁFICAS

[1] http://www.icmc.usp.br/pessoas/andre/research/neural/

[2]HTTP://ACROQUE.BLOGSPOT.COM.BR/2008/10/O-MODELO-DE-MCCULLOCH-E-PITTS-PARTE-1.HTML

[3]Getting Started with MATLAB.

```
%% Erick Amorim Fernandes 86301
%  Função matemática para representação de um neurônio
baseado no Modelo
%  de McCulloch-Pitts

% x -> vetor de entrada
% w -> vetor de pesos

function y = neuronio(x,w,bias)

soma = 0;
[A,B] = size(x);

for j = 1:A

    for i = 1:B
        soma = soma + x(j,i)*w(i);
    end

    y(j) = hardlim(soma+bias);
    y = y';
    soma = 0;
end


%  Função matemática para representação de um neurônio
baseado no Modelo
%  de McCulloch-Pitts

% x -> vetor de entrada
% w -> vetor de pesos

function y = neuronio(x,w,bias)

soma = 0;
[A,B] = size(x);

for j = 1:A

    for i = 1:B
        soma = soma + x(j,i)*w(i);
    end

    y(j) = hardlim(soma+bias);
    y = y';
    soma = 0;
end
```

```
%  Programa para utilização da função "neurônio" criada para
prática 2 de
%  disciplina de IC, assim como sua representação em plano
cartesiano.

%% Inicialização
close all
clear all
clc

%% NAND

%% Utilização do neurônio

Ent = [[0 0];   % Entrada de dados
       [0 1];
       [1 0];
       [1 1]];

w = [-0.9 -0.9]; % Pesos

bias = 1;

y = neuronio(Ent,w,bias);

%% Parte responsável pela plotagem

Xr = Ent(:,1);
Yr = Ent(:,2);
r = -((w(1)/w(2)).*Xr) - (bias/w(2));

figure()
plot(Xr(1:3),Yr(1:3),'*','linewidth',2,'MarkerSize',12)
hold on
plot(Xr(4),Yr(4),'rX','linewidth',2,'MarkerSize',12)
plot(Xr,r,'k','linewidth',2)
title('Função Lógica NAND')
legend('y = 1','y = 0','Location','North')

%% OR

%% Utilização do neurônio

Ent = [[0 0];   % Entrada de dados
       [0 1];
       [1 0];
       [1 1]];
```

```
w = [.9 .9]; % Pesos

bias = -.8;

y = neuronio(Ent,w,bias);

%% Parte responsável pela plotagem

Xr = Ent(:,1);
Yr = Ent(:,2);
r = -((w(1)/w(2)).*Xr) - (bias/w(2));

figure()
plot(Xr(2:4),Yr(2:4),'*','linewidth',2,'MarkerSize',12)
hold on
plot(Xr(1),Yr(1),'rX','linewidth',2,'MarkerSize',12)
plot(Xr,r,'k','linewidth',2)
title('Função Lógica OR')
legend('y = 1','y = 0','Location','North')

%% XOR

%% Representação gráfica

Ent = [[0 0];   % Entrada de dados
       [0 1];
       [1 0];
       [1 1]];

Xr = Ent(:,1);
Yr = Ent(:,2);


figure()
plot(Xr(2:3),Yr(2:3),'*','linewidth',2,'MarkerSize',12)
hold on
plot(Xr(1),Yr(1),'rX','linewidth',2,'MarkerSize',12)
plot(Xr(4),Yr(4),'rX','linewidth',2,'MarkerSize',12)
plot([0,1],[0.5,1.5],'k','linewidth',2)
plot([0,1],[-0.5,0.5],'k','linewidth',2)
title('Função Lógica XOR')
legend('y = 1','y = 0','Location','North')
```

```
%  Este programa tem por objetivo aprender um padrão de
analise crédito
%  através de um banco de dados inicial e depois reproduzir
sua tomada de
%  decisão em 3 casos hipotéticos

%% Inicialização

clear all
close all
clc

%% Dados para treinamento
%  Masculino será considerado como 1, Feminino como 0
%        Idade Sexo Casa Carro Casado Filhos  Renda
Resultado

Dados = [  18   1    0     1     0      0     1200        0;
%treinamento
           19   1    1     1     1      1      700        1;
%treinamento
           25   0    0     0     1      2      800        1;
%treinamento
           40   1    1     0     1      4      800        0;
%treinamento
           21   1    0     0     0      0     1100        1;
%treinamento
           22   0    1     1     1      2      500        0;
%treinamento
           18   0    1     0     0      0     1100        0;
%aplicação
           30   1    1     1     1      3      500        0;
%aplicação
           22   0    0     0     1      1     1200        0];
%aplicação
%            25   0    0     0     1      2      800
1]; %aplicação


%% Normalizando

dds_normalizado = [normalizar(Dados(:,1))   , Dados(:,2:5) ,
...
                   normalizar(Dados(:,6:7)) , Dados(:,8)];

%% Treinamento
```

```
%  Encontrar os pesos que satisfazem nossos dados de entrada
tomando o bias = -1;

bias = -1;
w = ones(1,7); % Criação do vetor de pesos
y = ones(6,1); % Vetor de resposta
flag = 0;
a=-2;
b=2;

while flag == 0

    w = a+(b-a)*rand(1,7); % Intervalo para busca de soluções
    y = neuronio(dds_normalizado(1:6,1:7) ,w ,bias);
    flag = isequal(y,dds_normalizado(1:6,8));

end


%% Aplicando ao neurônio

[A,L] = size(Dados); % Responsável por pegar o ponto final
dos dados de aplicação

y2 = neuronio(dds_normalizado(7:A,1:7) ,w ,bias)
```