Neurônio de McCulloch-Pitts

Denise Cristina Henrique de Freitas

ELT 460 – Inteligência Computacional

Departamento de Engenharia Elétrica, Universidade Federal de Viçosa, Viçosa - MG

E-mail: denise.henrique@ufv.br

*Resumo - Este relatório aborda o clássico modelo de neurônio artificial proposto por McCulloch e Pitts, que consiste em um dos conceitos básicos para iniciar os estudos referentes as redes neurais. Para auxiliar o desenvolvimento deste trabalho foram empregadas ferramentas do Matlab, possibilitando a implementação do modelo, bem como aplicação em exemplos.*

1. INTRODUÇÃO

Uma Rede Neural Artificial (RNA) consiste em uma estrutura de processamento, com possibilidades de implementação em dispositivos eletrônicos, cuja composição se dá através de unidades de processamento simples que apresentam um comportamento específico de entrada/saída, o qual é determinado pela função de transferência, pelas conexões com outras unidades e por meio de entradas externas.[1]

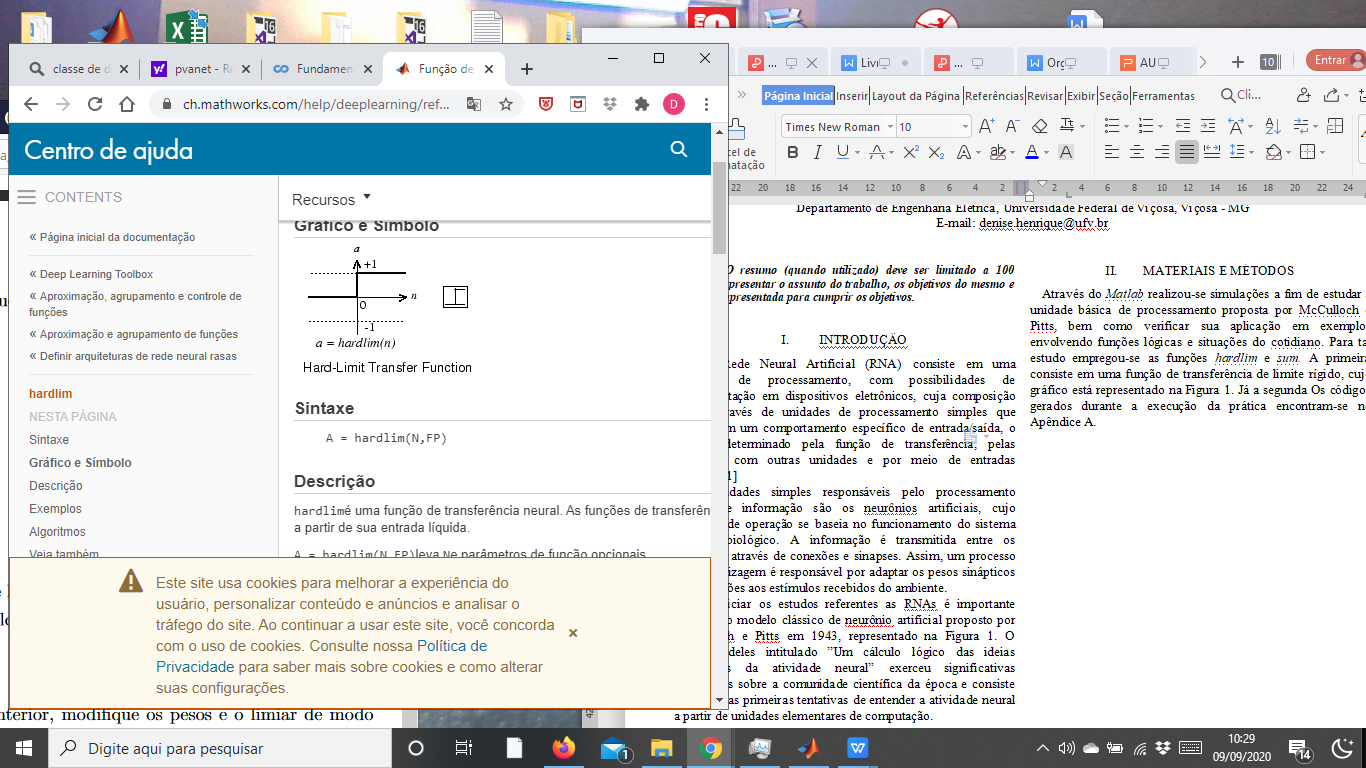
As unidades simples responsáveis pelo processamento básico de informação são os neurônios artificiais, cujo princípio de operação se baseia no funcionamento do sistema nervoso biológico. A informação é transmitida entre os neurônios através de conexões e sinapses. Assim, um processo de aprendizagem é responsável por adaptar os pesos sinápticos das conexões aos estímulos recebidos do ambiente[2].

Para iniciar os estudos referentes as RNAs é importante conhecer o modelo clássico de neurônio artificial proposto por McCulloch e Pitts em 1943. O trabalho deles intitulado ”Um cálculo lógico das ideias intrínsecas da atividade neural” exerceu significativas influências sobre a comunidade científica da época e consiste em uma das primeiras tentativas de entender a atividade neural a partir de unidades elementares de computação[1].

Assim, tendo em vista a importância das Redes Neurais em distintos campos de pesquisa da atualidade e a necessidade de estudar os elementos mais básicos para posterior compreensão do seu funcionamento, o presente trabalho tem por objetivo entender o modelo de neurônio proposto por McCulloch e Pitts, implementá-lo através do *Matlab* e, por fim, analisar algumas aplicações que envolva tal conceito.

1. MATERIAIS E MÉTODOS

Através do *Matlab* realizou-se simulações a fim de estudar a unidade básica de processamento proposta por McCulloch e Pitts, bem como verificar sua aplicação em exemplos envolvendo funções lógicas e situações do cotidiano. Para tal estudo empregou-se a função *hardlim,* queconsiste em uma função de transferência de limite rígido, cujo gráfico está representado na Figura 1. Os códigos gerados durante a execução da prática encontram-se no Apêndice A.

1. 
2. Figura 1 - Neurônio de McCulloch e Pitts [3]
3. Implementação do neurônio

No modelo de neurônio artificial de McCulloch e Pitts, representado na figura 2, a unidade básica de processamento possui um limiar (threshold) fixo θ, e recebe algumas entradas (x1,x2,...xm), que são ponderadas através de pesos sinápticos. Além disso, a função de ativação é reponsável pela determinação do estado da saída[4].

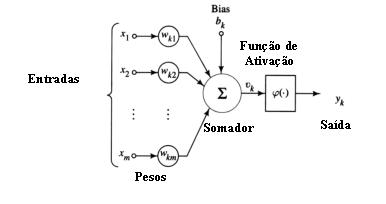


Figura 2 - Neurônio de McCulloch e Pitts[4]

Matematicamente, a saída do neurônio pode ser expressa por

 (1)

Assim, para implementação no *Matlab* criou-se uma função, através do comando *function* que recebe três parâmetros, sendo eles o vetor de entradas, o vetor pesos e o valor bias, realiza o somatório das entradas ponderadas e aplica a função de ativação (comando *hardlim*), retornando ao usuário o valor de y. Afim de verificar o comportamento da função criada, realizou-se dois testes, considerando quatro entradas, nos quais os vetores de entrada e de peso foram gerados aleatoriamente e foi aplicado um limiar θ = -1.

1. Funções Lógicas

Afim de verificar a aplicação da função criada, que foi descrita no tópico anterior, empregou-se o neurônio implementado para emular as funções lógicas NAND, OR e XOR de dois bits. As tabelas-verdade de cada uma dessas funções encontram-se ilustradas nas tabelas 1, 2 e 3, respectivamente.

Tabela 1 - Lógica NAND Tabela 2 - Lógica OR

|  |  |  |
| --- | --- | --- |
| X1 | X2 | yd |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| X1 | X2 | yd |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Tabela 3 - Lógica XOR

|  |  |  |
| --- | --- | --- |
| X1 | X2 | yd |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Nesse caso, o vetor de entradas empregado para emular cada uma das funções lógicas indicadas acima, corresponde ao conjunto de possibilidades de valores que as duas variáveis binárias podem assumir, utilizados para construção da tabela-verdade. Tendo em vista, a simplicidade do modelo, resultante da aplicação de apenas duas entradas, a determinação do vetor de pesos necessário para a obtenção das saídas desejadas, pode ser obtido sem muitos esforços.

É importante destacar que o neurônio de McCulloch e Pitts apresenta um bom desempenho apenas para determinação de padrões relacionados a funções linearmente separáveis. Uma função classificada desta forma permite que plotando os resultados em um gráfico que relacione as variáveis de entrada seja possível separar através de uma reta os padrões distintos, nesse caso 0 e 1. A reta que divide o espaço de características em dois, equivale ao limiar do teste da tomada de decisão do perceptron, ou seja, ela segue a expressão[5]

w1x1+ w2x2+ θ = 0 (2)

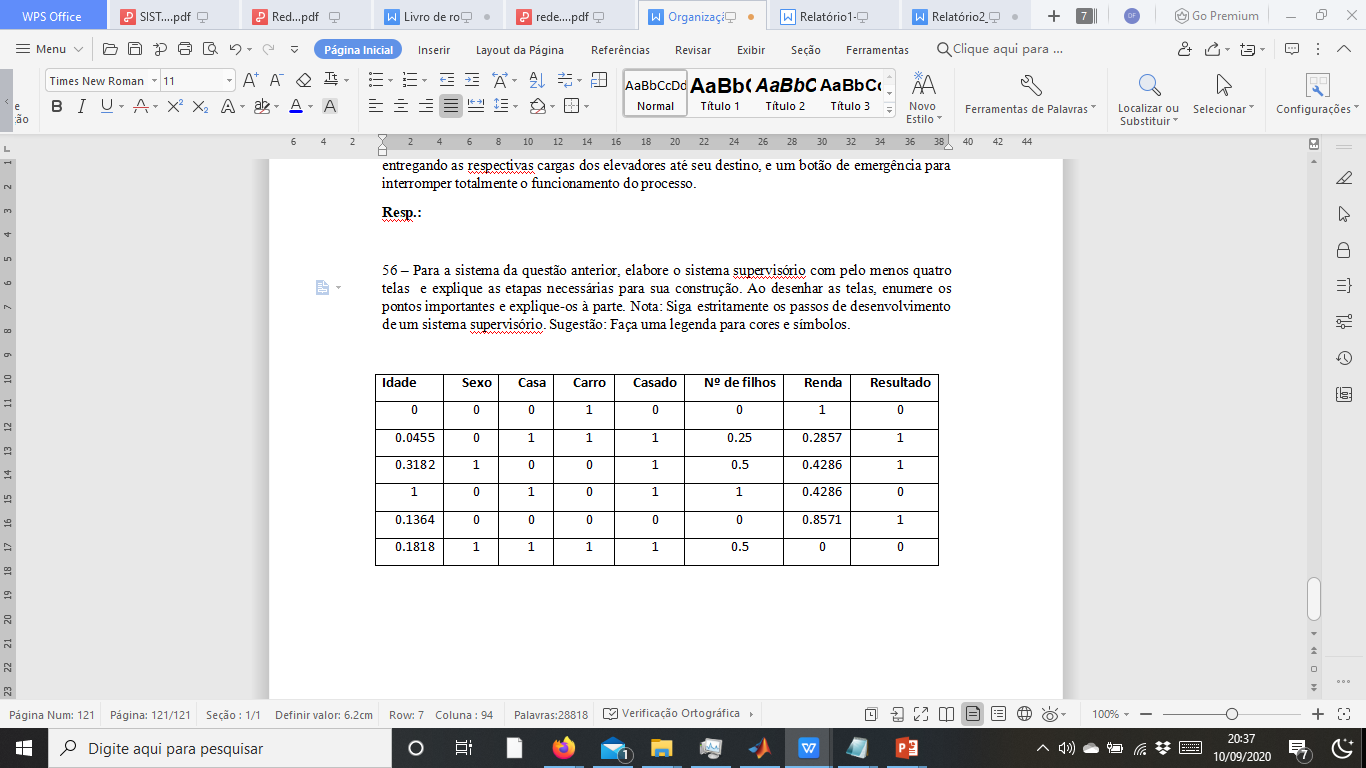
Dadas as funções acima, apenas a lógica XOR não consiste em uma função linearmente separável, consequentemente espera-se que o modelo adeque-se bem aos outros dois casos.

1. Análise de crédito

Na análise de crédito um especialista avalia o potencial de retorno de um indivíduo, bem como os riscos inerentes à concessão. Assim, é possível identificar os clientes que futuramente poderão acarretar uma situação de risco de caixa à organização. Posto isso, na última etapa deste trabalho é realizado um ajuste no neurônio artificial, de modo que ele seja capaz de simular as decisões de um especialista em análise de crédito.

Uma vez que as variáveis analisadas são de ordens consideravelmente diferentes, realizou-se a normalização dos dados, afim de trabalhar em um mesmo intervalo de valores. Além disso, foi atribuído o valor zero a entrada referente ao sexo masculino e um ao feminino. Os valores obtidos após tais procedimentos encontram-se dispostos na tabela 4.

Tabela 4 - Análise de crédito com dados normalizados



Para determinação dos valores dos pesos realizou-se algumas iterações até que o programa identificasse um vetor de pesos cuja multiplicação pelas respectivas entradas resultasse nos valores desejados de saídas. Os valores foram gerados aleatoriamente com distribuição uniforme no intervalo de [-10,10].

A aplicação do neurônio ajustado foi realizada para análise de crédito de três clientes distintos, representados pelos seguinte conjunto de dados x1=[18, F,1,0,0,0,1100], x2=[30, M,1,1,1,3,500] e x3=[20,F,0,0,1,1,1200].

Nesse caso, tendo em vista que o conjunto de dados de entrada possui mais de duas variáveis, a existência de um vetor de pesos que pondere as entradas adequadamente é condicionada a existência de um hiperplano separador de dimensão n-1 (onde n é o número de variáveis de entrada), cuja equação é dada por[5]

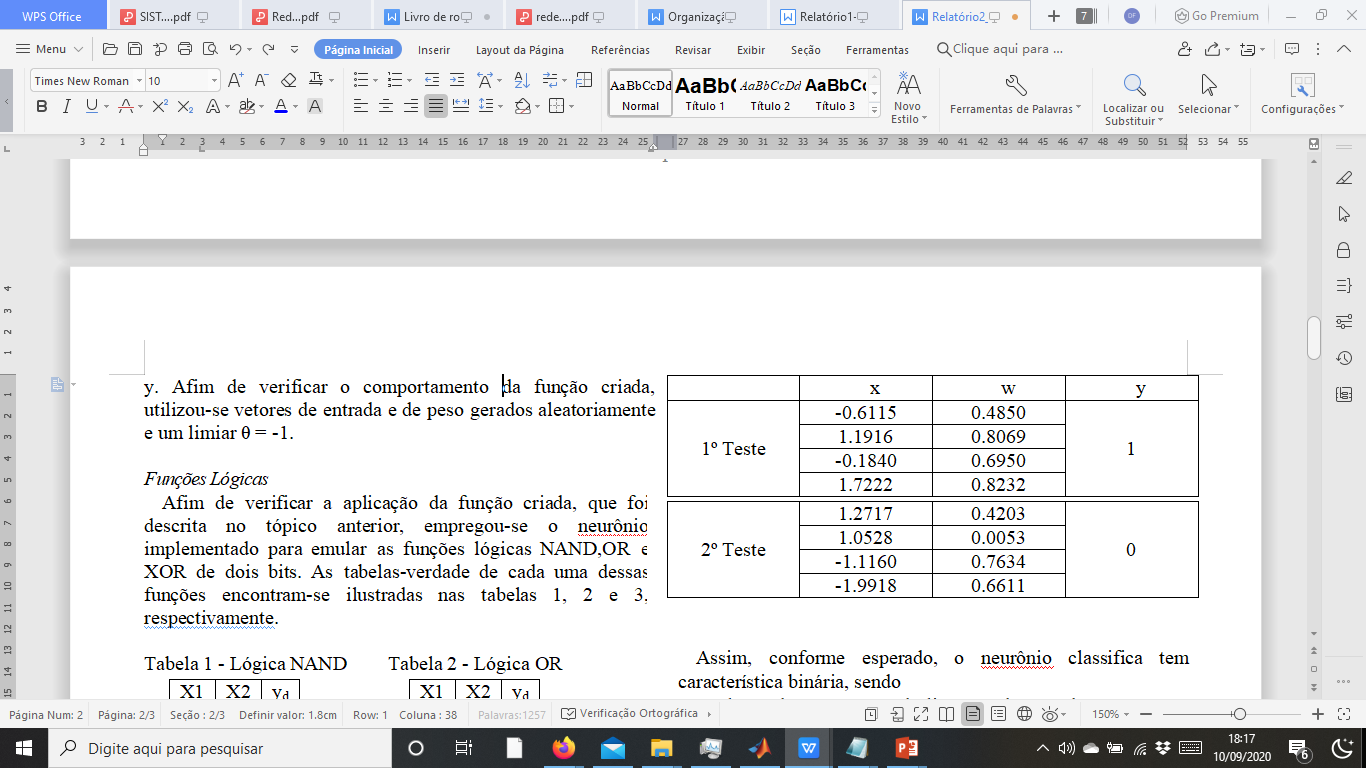
f(x) = **wx** + θ = 0, (3)

sendo w é o vetor de pesos e θ um escalar que representa o limiar.

1. RESULTADOS

Os valores obtidos para a saída do neurônio artificial, bem como os vetores de entrada e de peso gerados aleatoriamente para cada um dos testes realizados com o neurônio modelado na primeira etapa, encontram-se dispostos na tabela 5.

Tabela 5 - Dados dos testes com o neurônio artificial



Conforme esperado, a saída apresenta um comportamento binário, sendo a função de ativação, um degrau unitário. Assim, como no primeiro conjunto de dados o somatório das entradas ponderadas acrescido de θ resulta em um valor superior a zero, o neurônio indica em sua saída o valor um, já para o segundo conjunto de dados temos a situação oposta, resultando, pois, em zero.

Em relação a segunda etapa da prática, as retas de separação para as funções NAND e OR podem ser analisadas nas figuras 3 e 4, respectivamente. Para obtenção da saída desejada na função lógica NAND utilizou-se o vetor de pesos w=[-0.5 -0.8] e θ = 1, já para a função OR, o vetor de pesos empregado foi w=[0.5 0.8] e θ = -0.3.

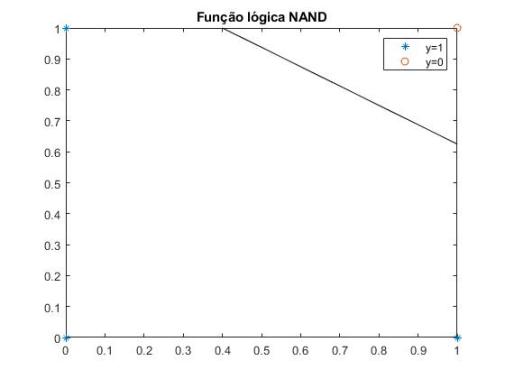


Figura 3 - Reta de separação para função NAND

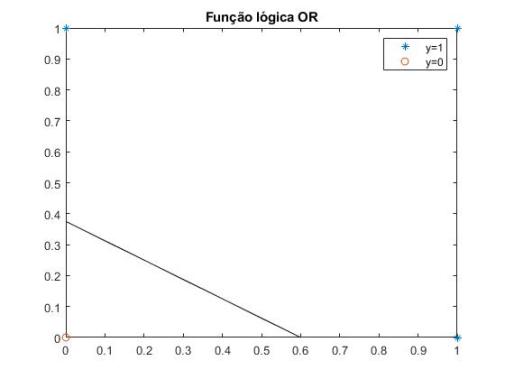


Figura 4 - Reta de separação para função OR

De acordo com o apresentado anteriormente em relação a função lógica XOR, não é possível determinar uma reta de separação para os valores de saída, consequentemente, o modelo de McCulloch e Pitts não é efetivo para classificação de padrões nesse caso. Assim, não foi possível determinar um conjunto de pesos, para ponderar as entradas, de modo a obter os valores de saídas desejadas.

Por fim, em relação a análise de créditos, o vetor de pesos obtido após as iterações foi

w = [-6.2912, -0.2709, 3.5630, -7.0192, 5.6795, -6.4774,

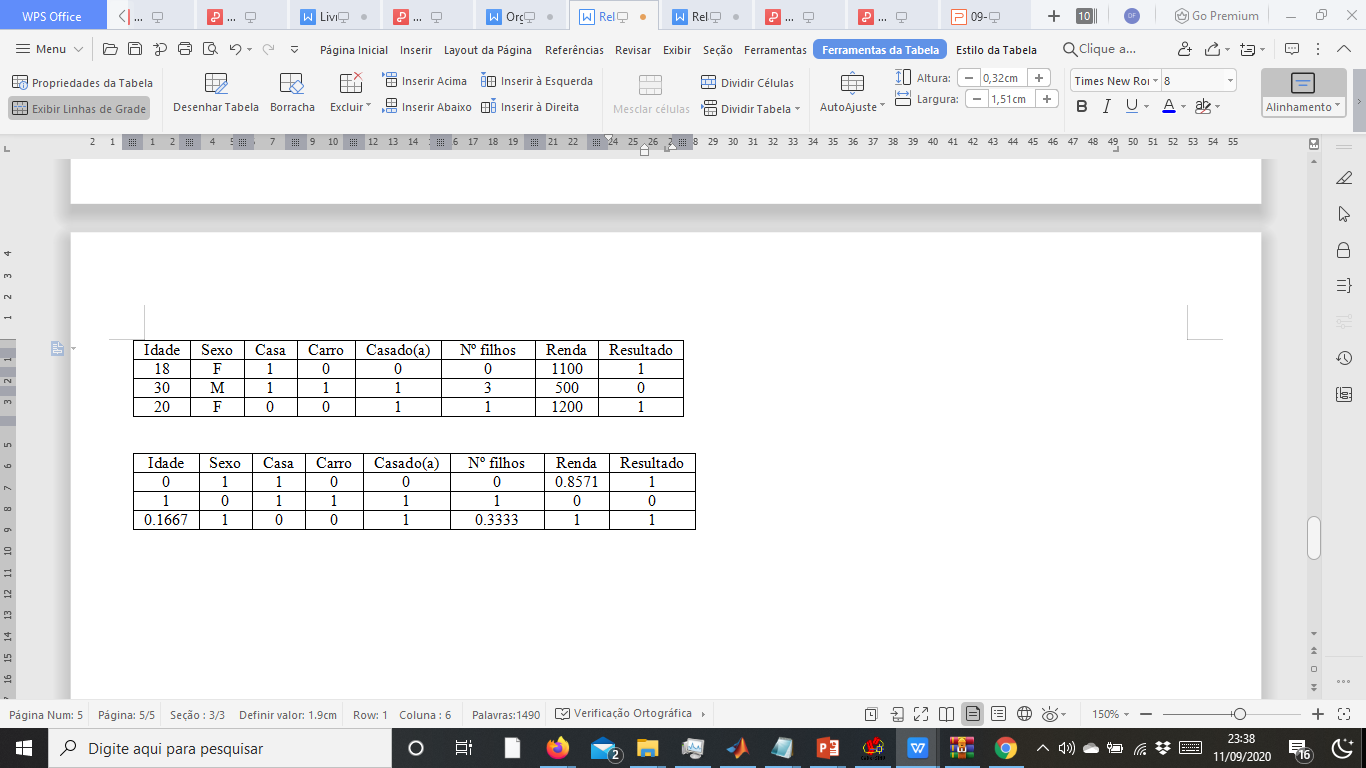
7.9545]

Portanto, uma vez que foi possível obter o vetor w, tal que as entradas são corretamente ponderadas atingindo as saídas

desejadas para todos os conjuntos de dados apresentados na tabela 4, é possível afirmar que existe, não uma reta, mas sim um hiperplano de dimensão n-1 que separa essas classes no hiperespaço.

O resultado da aplicação da análise para os três clientes diferentes, encontram-se dispostos na tabela 6. Assim como na fase de ajuste do neurônio os dados de entrada foram normalizados.

Tabela 6 - Dados da aplicação do neurônio ajustado



1. CONSIDERAÇÕES FINAIS

Desse modo, a execução desta prática possibilitou uma melhor compreensão dos conceitos envolvidos no modelo de neurônio artificial de McCulloch e Pitts, que consiste em uma das bases para o desenvolvimento das redes neurais, largamente empregadas na atualidade. Assim, através dos recursos do *Matlab* foi possível simular e verificar o comportamento desta unidade básica de processamento, analisando seu comportamento satisfatório para classes linearmente separáveis. A terceira etapa do trabalho evidencia a aplicabilidade de tais conceitos em problemas cotidianos, enfatizando a importância do estudo de tal conteúdo. Posto isso, os objetivos do trabalho foram alcançados com êxito.

Referências Bibliográficas

[1] ZUBEN, Fernando Von; CASTRO, Leandro N. de. Redes Neurais Artificiais. In: PROJETO de Redes Neurais Artificiais. DCA/FEE: Unicamp, 2012. cap. 1, p. 1-22.

[2] Data Science Academy. Deep Learning Book, 2019. Disponível em: <http://www.deeplearningbook.com.br/>. Acesso em: 12 setembro. 2020.

[3] MATHWORKS. Help Center. Disponível em: https://www.mathworks.com/help/deeplearning/ref/hardlim.html. Acesso em: 12 set. 2020.

[4] ZAMBIASI, Saulo Popov. Redes Neurais:O Neurônio Artificial. ed. Faculdade BARDDAL: 2008.Disponível em: https://www.gsigma.ufsc.br/~popov/aulas/rna/neuronio\_artificial/index.html. Acesso em: 12 set. 2020.

[5] SILVA, Jacson Rodrigues.. Modelo McCulloch e Pitts. CCA UFES:, 2014. Disponível em: http://jeiks.net/wp-content/uploads/2014/08/RNA-Slides\_04.pdf. Acesso em: 12 set. 2020.

Apêndice A

%Neurônio artificial McCulloch e Pitts

function y = neuronio(x,w,bias)

soma=0;

[l,c]=size(x);%Armazena o número de linhas e colunas da matriz de entrada nas variáveis l e c

for k= 1:c

for i=1:l

soma=soma+(x(i)\*w(i)); %Somatório das entradas ponderadas

end

u=soma+bias;

y=hardlim(u);%Aplica a função de ativação para determinar o estado da saída

soma=0;

end

% NAND

x=[[0 0];[0 1];[1 0];[1 1]];

w=[-0.5 -0.8];

bias=1;

y=neuronio(x,w,bias);

x1=x(:,1);

x2=x(:,2);

r=-((w(1)/w(2)).\*x1)- (bias/w(2));

plot(x1(1:3),x2(1:3),'\*')

hold on

plot(x1(4),x2(4),'o')

plot(x1,r,'black')

title('Função lógica NAND')

axis([0 1 0 1])

legend('y=1','y=0')

% OR

x=[[0 0];[0 1];[1 0];[1 1]];

w=[0.5 0.8];

bias=-0.3;

y=neuronio(x,w,bias);

x1=x(:,1);

x2=x(:,2);

r=-((w(1)/w(2)).\*x1)- (bias/w(2));

plot(x1(2:4),x2(2:4),'\*')

hold on

plot(x1(1),x2(1),'o')

plot(x1,r,'black')

title('Função lógica OR')

axis([0 1 0 1])

legend('y=1','y=0')

% Ajuste do neurônio para análise de crédito

Idade=[18 19 25 40 21 22];

Idaden=(Idade-min(Idade).\*ones(1,6))/(max(Idade)-min(Idade));%Vetor idade normalizado

filhos=[0 1 2 4 0 2];

filhosn=(filhos-min(filhos).\*ones(1,6))/(max(filhos)-min(filhos));%Vetor nº de filhos normalizado

renda=[1200 700 800 800 1100 500];

rendan=(renda-min(renda).\*ones(1,6))/(max(renda)-min(renda));%Vetor renda normalizado

x=[[Idaden(1) 0 0 1 0 filhosn(1) rendan(1)];[Idaden(2) 0 1 1 1 filhosn(2) rendan(2)];[Idaden(3) 1 0 0 1 filhosn(3) rendan(3)];[Idaden(4) 0 1 0 1 filhosn(4) rendan(4)];[Idaden(5) 0 0 0 0 filhosn(5) rendan(5)];[Idaden(6) 1 1 1 1 filhosn(6) rendan(6)]]%vetor das entradas

bias=-1;

yd=[0 1 1 0 1 0];%Vetor de saídas desejadas

y=[5 5 5 5 5 5];%Vetor com valores de saída diferente das desejadas

[l,c]=size(x);

soma=0;

a=-10;%limite inferior para o vetor de pesos

b=10;%limite inferior para o vetor de pesos

s=0;

cont=0;

g=true;

while g

w=a+(b-a)\*rand(7,1);%Gera um novo vetor de pesos a casa iteração com valores no intervalo [a,b]

for i=1:l

for j=1:c

soma=soma+(x(i,j).\*w(j));

end

u=soma+bias;

if(u<0)

y(i)=0;

else

y(i)=1;

end

soma=0;

end

for k=1:length(y)

if y(k)==yd(k)

s=s+1;

end

end

if s==6 %Verifica se o todos os elementos de y são equivalentes a yd

g=false;

end

s=0;

cont=cont+1;

End

w %Exibe o vetor de pesos

%% Aplicação do neurônio ajustado

w=[-6.29117388166985;-0.270856365571673;3.56301785547054;-7.01916239602372;5.67948771071601;-6.47738630133723;7.95451595134727];

Idade=[18 30 20];

Idaden=(Idade-min(Idade).\*ones(1,3))/(max(Idade)-min(Idade))%Vetor idade normalizado

filhos=[0 3 1];

filhosn=(filhos-min(filhos).\*ones(1,3))/(max(filhos)-min(filhos))%Vetor nº de filhos normalizado

renda=[1100 500 1200];

rendan=(renda-min(renda).\*ones(1,3))/(max(renda)-min(renda))%Vetor renda normalizado

x=[[Idaden(1) 1 1 0 0 filhosn(1) rendan(1)];[Idaden(2) 0 1 1 1 filhosn(2) rendan(2)];[Idaden(3) 1 0 0 1 filhosn(3) rendan(3)]]%vetor das entradas

bias=-1;

soma=0;

[linhas,colunas]=size(x);

for l=1:linhas

for c=1:colunas

soma=soma+x(l,c).\*w(c);

end

u=soma+bias;

if u<0

y(l)=0;

else

y(l)=1;

end

end

y %Exibe o vetor de saída