```
%% 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)
```