Universidade Federal do ABC - UFABC

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas - CECS

Programa de Pós-Graduação em Energia

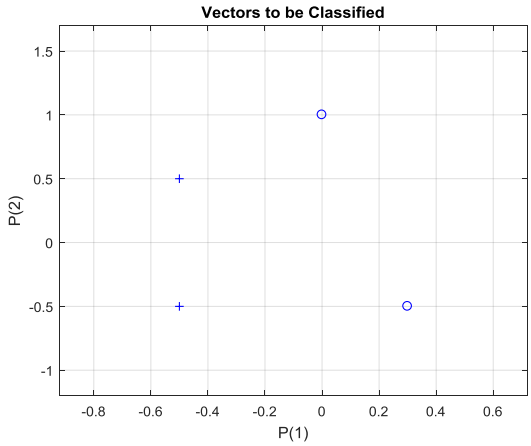
ENE-300/EEL-312: Inteligência Computacional

Exercícios: Redes Neurais Artificiais

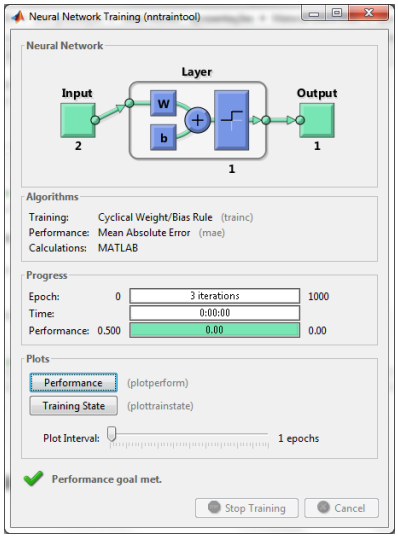
RCS

Março - 2021

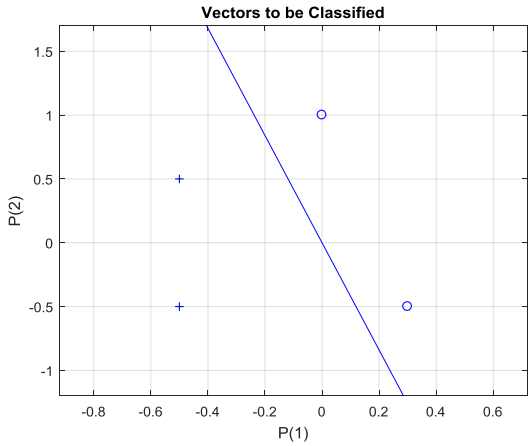
```
% Exercício 1 (neste exercício a RNA deve encontrar uma linha que divide
as classes: P1 negativo a saída = 1 = + e P1 positivo a saída = 0 = o)

P=[-0.5 -0.5 0.3 0; -0.5 0.5 -0.5 1.0] % vetor de treinamento - entrada
T=[1 1 0 0] % vetor de treinamento - saída
plotpv(P,T) % plota os vetores de entrada e saída no plano
grid
pause

% vetores para treinamento - entrada e saída

P =
   -0.5000   -0.5000    0.3000         0
   -0.5000    0.5000   -0.5000    1.0000

T =
     1     1     0     0
```



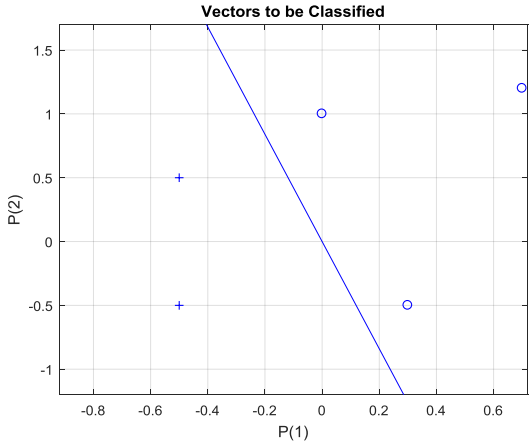

```
net = newp([-1 1;-1 1],1); %um neurônio com duas entradas (de -1 a 1)
net.trainParam.epochs = 500; %se comentar, usa valores default (não muda)
net.trainParam.goal = 1e-9; %se comentar, usa valores default (não muda)
net = train(net,P,T);%treina RNA segundo P e T
```



```
net.IW{1} %mostra pesos utilizados/encontrados
net.b{1} %mostra bias utilizado/encontrado
pause

-2.1000   -0.5000 %pesos

0 %bias

figure
plotpv(P,T) %plota os vetores de entrada e saída no plano
plotpc(net.IW{1},net.b{1}) %plota a linha de classificação da rna
grid
```



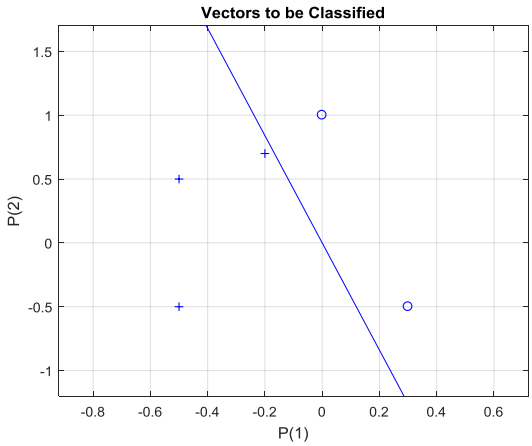

```
% testando com nova entrada

P1 = [0.7; 1.2]
pause

P1 =
    0.7000
    1.2000

a1 = sim(net,P1); %simula a rna com a nova entrada P1
plotpv(P1,a1);
hold on
plotpv(P,T)
plotpc(net.IW{1},net.b{1}); %plota resposta de P1=0 acima da linha da rna
grid
pause
```




```
% testando com nova entrada
P2 = [-0.2; 0.7]
pause
```

```
P2 =
   -0.2000
    0.7000
```

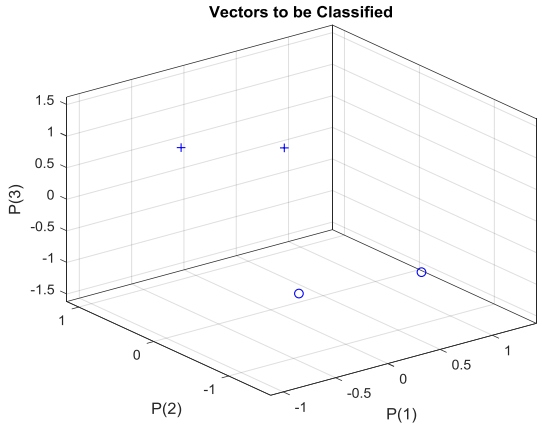
```
a2 = sim(net,P2); %simula a rna com a nova entrada P2
figure
plotpv(P2,a2);
hold on
plotpv(P,T)
plotpc(net.IW{1},net.b{1});%plota resposta de P2=1 abaixo da linha da rna
grid
```



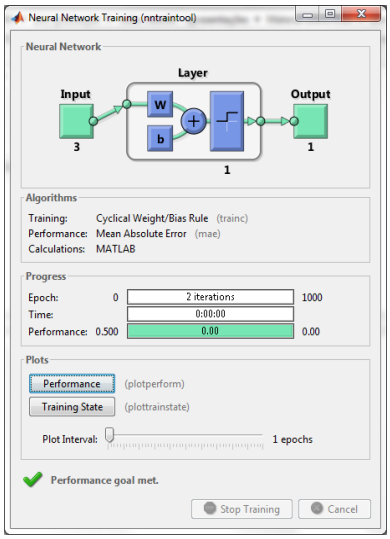

```
% neste exercício a rna aprende que quando x é negativo a saída é 1 e
está abaixo da reta
```

```
% Exercício 2 (neste exercício a rna aprende que quando a entrada 3 é
positiva a saída é 1 = + e quando a entrada 3 é negativa a saída é 0 = o)

% definindo e plotando os vetores de treinamento

P=[-0.5 0.2 -0.3 0.8; 0.5 0.1 -0.8 -0.9;
    0.9 0.8 -0.8 -0.9]
T=[1 1 0 0]
plotpv(P,T)
grid
pause
```

```
P =
   -0.5000    0.2000   -0.3000    0.8000
    0.5000    0.1000   -0.8000   -0.9000
    0.9000    0.8000   -0.8000   -0.9000


T =
     1     1     0     0
```




```
% definindo e treinando a rna (se não definiu o número máximo de
iterações ou erro, os valores default são usados)

net = newp([-1 1;-1 1;-1 1],1); %defini número de neurônios e entradas
net = train(net,P,T); %treina a rna segundo P e T
```

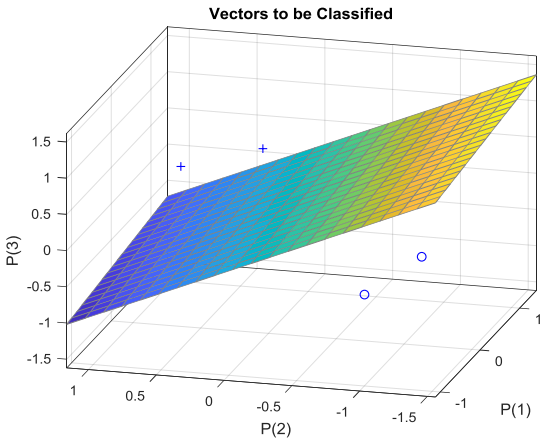



```
net.IW{1}
net.b{1}
```

```
-0.2000    1.3000    1.7000 %pesos
```

```
0 %bias
```

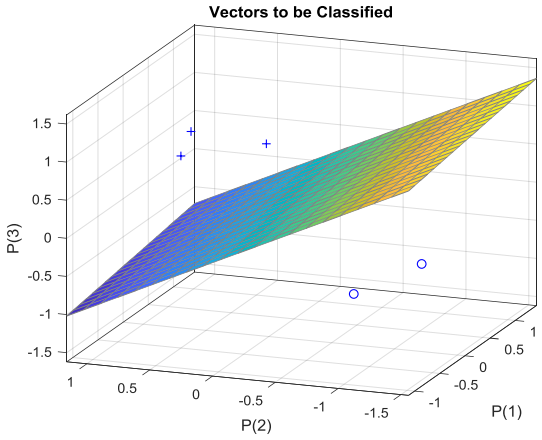
```
% plotando o plano que divide/classifica os vetores de entrada
pause
figure
plotpv(P,T)
plotpc(net.IW{1},net.b{1})
grid
```



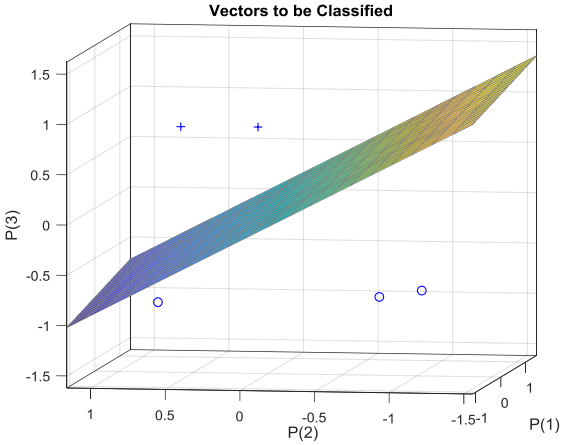

```
% testando com nova entrada P1
P1 = [0.7; 0.9; 0.6]
pause
```

```
P1 =
    0.7000
    0.9000
    0.6000
```

```
a1 = sim(net,P1)
v=[-1 1 -1 1];
plotpv(P1,a1,v); % plota a resposta de P1 acima do plano
hold on
plotpv(P,T)
plotpc(net.IW{1},net.b{1});
grid
pause
```




```
% testando com nova entrada
P2 = [-0.2; 0.7; -0.9]
pause

P2 =
   -0.2000
    0.7000
   -0.9000
a2 = sim(net,P2)
figure
plotpv(P2,a2,v); % plota a resposta de P2 abaixo do plano
hold on
plotpv(P,T)
plotpc(net.IW{1},net.b{1});
grid
```

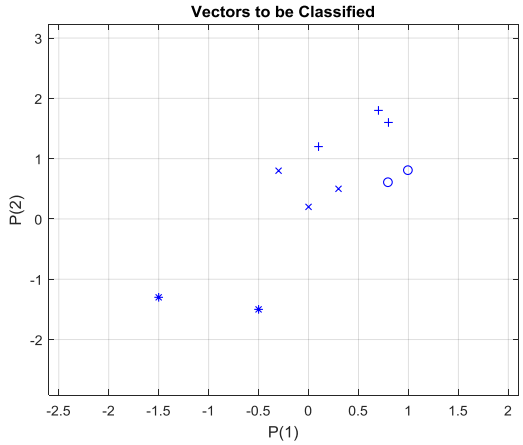



```
%A rna aprendeu que quando a entrada 3 é positiva a saída é 1 = + e
quando a entrada 3 é negativa a saída é 0 = o. Entrada 3 positiva = 1 e
entrada 3 negativa = 0
```

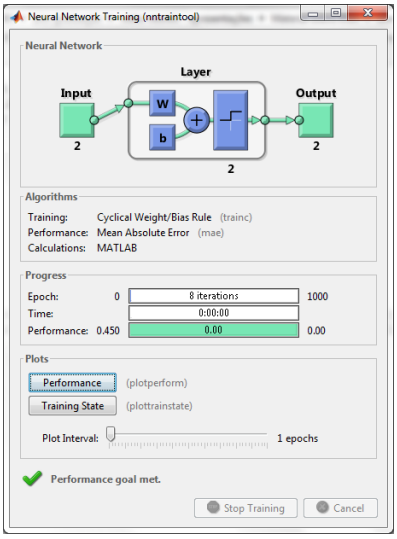
```
% Exercício 3 (neste exercício a rna deve classificar novas entradas
segundo uma das quatro classes existentes)

%definindo os vetores de treinamento
P=[0.1 0.7 0.8 0.8 1 0.3 0 -0.3 -0.5 -1.5;
   1.2 1.8 1.6 0.6 0.8 0.5 0.2 0.8 -1.5 -1.3]
T=[1 1 1 0 0 1 1 1 0 0; 0 0 0 0 0 1 1 1 1 1]
```

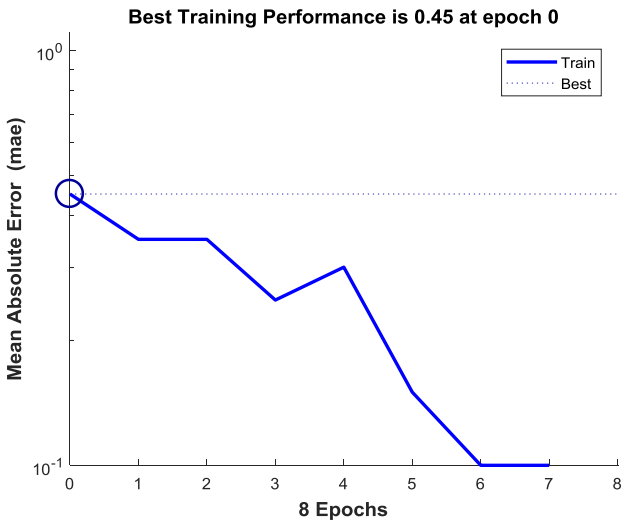
```
P =
    0.1    0.7    0.8    0.8    1.0    0.3      0   -0.3   -0.5   -1.5
    1.2    1.8    1.6    0.6    0.8    0.5    0.2    0.8   -1.5   -1.3

T =
     1     1     1     0     0     1     1     1     0     0
     0     0     0     0     0     1     1     1     1     1
```

```
%plotando os vetores de treinamento
plotpv(P,T)
grid
pause
```

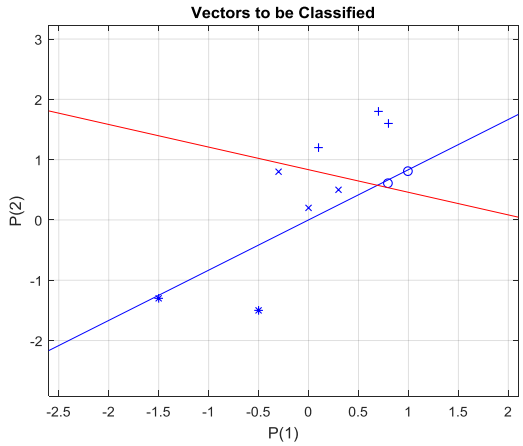



```
%criando e treinando as RNAs
net = newp([-2 2;-2 2],2); %dois neurônios com entradas entre -2 e +2
net = train(net,P,T);
```

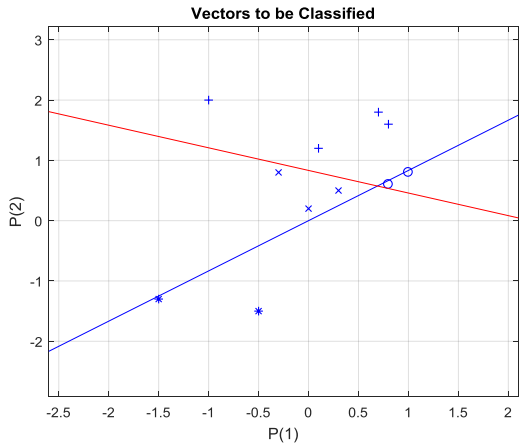




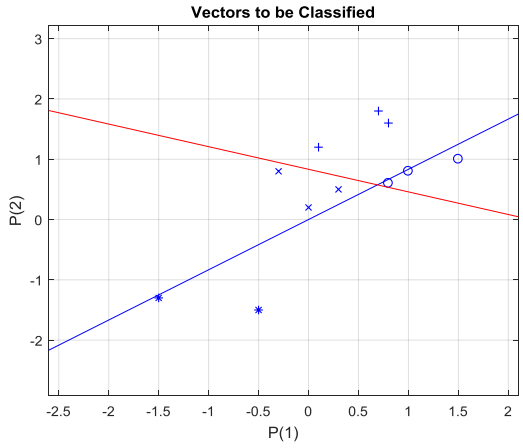
```
%pesos e bias encontrados
net.IW{1}
net.b{1}

pesos =
   -2.0000    2.4000
   -0.9000   -2.4000

bias =
     0
     2

% Plotando as retas que dividem o plano em quatro áreas
pause
figure
plotpv(P,T)
plotpc(net.IW{1},net.b{1})
grid
```




```
% testando com a nova entrada P1
P1 = [-1; 2]

P1 =
    -1
     2

pause
a1 = sim(net,P1) %simulando a rna com a nova entrada P1

a1 =
     1
     0

plotpv(P1,a1); %Plotando resposta de P1 com os vetores que possuem Y > 1
hold on
plotpv(P,T)
plotpc(net.IW{1},net.b{1});
grid
pause
```



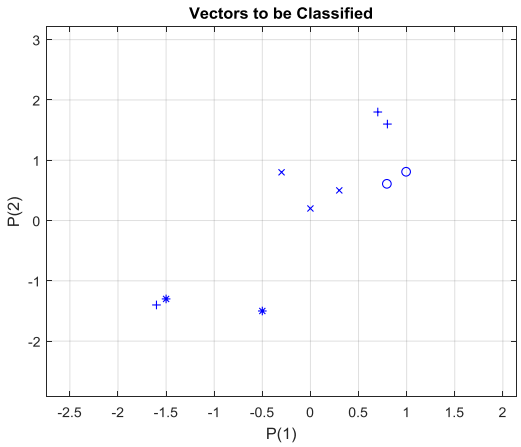

```
% testando com a nova entrada P2
P2 = [1.5; 1]

P2 =    1.5000
        1.0000

pause
a2 = sim(net,P2) %simulando a rna com a nova entrada P2

a2 =
     0
     0

figure %Plotando resposta de P2 com os vetores que possuem Y e X >0 e <1
plotpv(P2,a2);
hold on
plotpv(P,T)
plotpc(net.IW{1},net.b{1});
grid
```



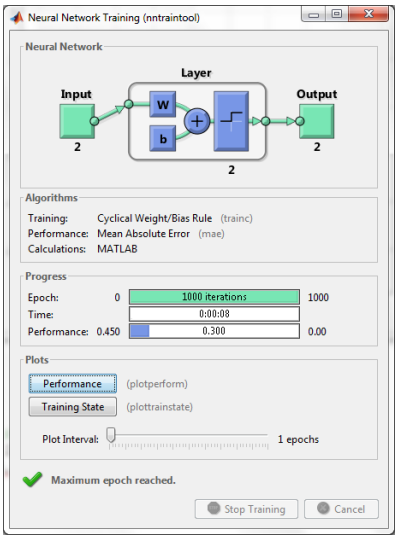

```
% Neste exercício a rna classificou novas entradas segundo uma das quatro
classes criadas durante o treinamento
```

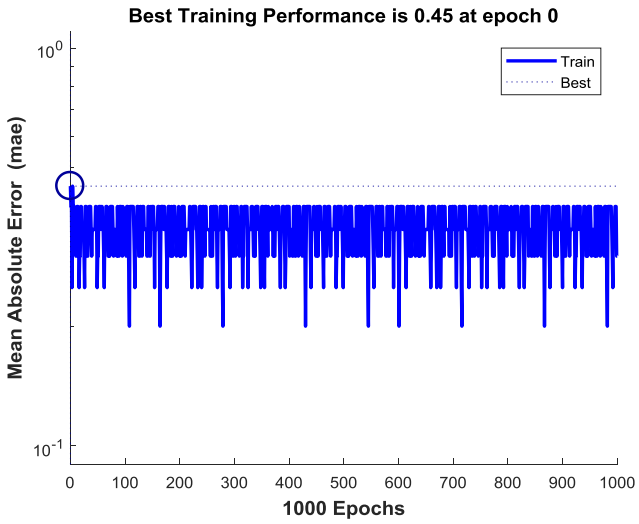
```
% Exercício 4 (este exercício é igual ao anterior, porém com uma
inconsistência no conjunto de treinamento, ou seja, o seu primeiro
elemento é -1.6 e -1.4). Ele deveria pertencer a classe 01 e não 10.

%Plotando os vetores de entrada
P=[-1.5 0.7 0.8 0.8 1 0.3 0 -0.3 -0.5 -1.5;
   -1.3 1.8 1.6 0.6 0.8 0.5 0.2 0.8 -1.5 -1.3]
T=[1 1 1 0 0 1 1 1 0 0; 0 0 0 0 0 1 1 1 1 1]
plotpv(P,T)
grid
pause
```

```
P =
   -1.6   0.7   0.8   0.8   1.0   0.3   0.0  -0.3  -0.5  -1.5
   -1.4   1.8   1.6   0.6   0.8   0.5   0.2   0.8  -1.5  -1.3
T =
     1     1     1     0     0     1     1     1     0     0
     0     0     0     0     0     1     1     1     1     1
```

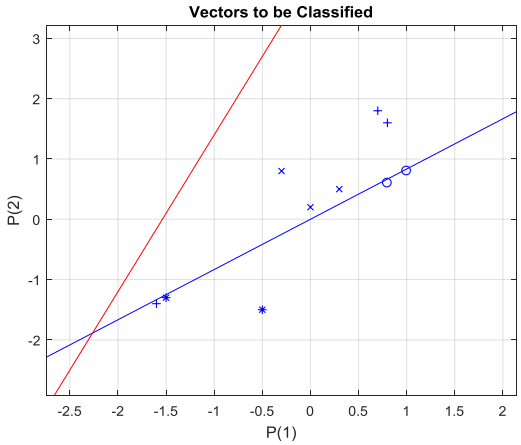



```
net = newp([-2 2;-2 2],2); % definindo e treinando a RNA
net = train(net,P,T);
net.IW{1}
net.b{1}
```

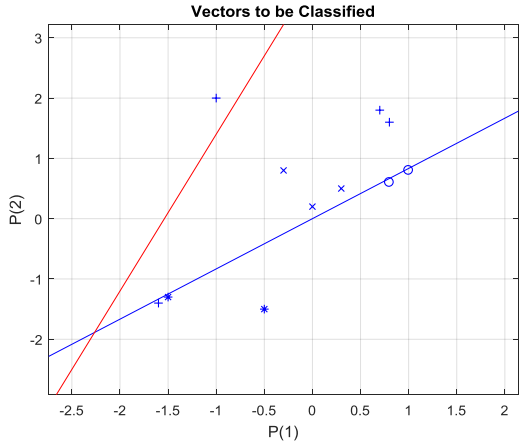




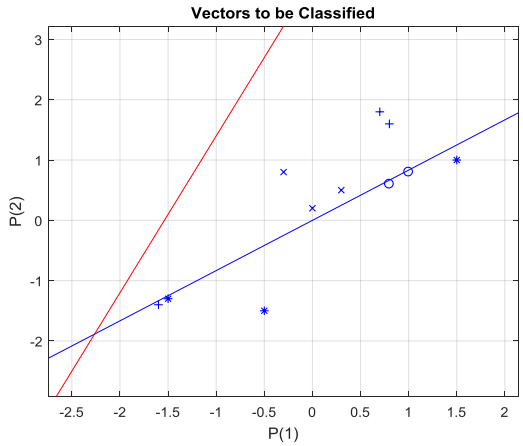
```
net.IW{1}
net.b{1}

pesos=
   -3.5000    4.2000
    1.3000   -0.5000

bias=
     0
     2

%plotando as retas que criam as áreas
pause
figure
plotpv(P,T)
plotpc(net.IW{1},net.b{1})
grid
```




```
% testando com a nova entrada P1
P1 = [-1; 2]
pause

P1 =

    -1
     2

a1 = sim(net,P1)

a1 =
     1
     0

%plotando a resposta (não há classificação)
plotpv(P1,a1);
hold on
plotpv(P,T)
plotpc(net.IW{1},net.b{1});
grid
pause
```




```
% testando com a nova entrada P2
P2 = [1.5; 1]
pause
a2 = sim(net,P2)

a2 =
     0
     1

%plotando a resposta (não há classificação)
figure
plotpv(P2,a2);
hold on
plotpv(P,T)
plotc(net.IW{1},net.b{1});
grid
```

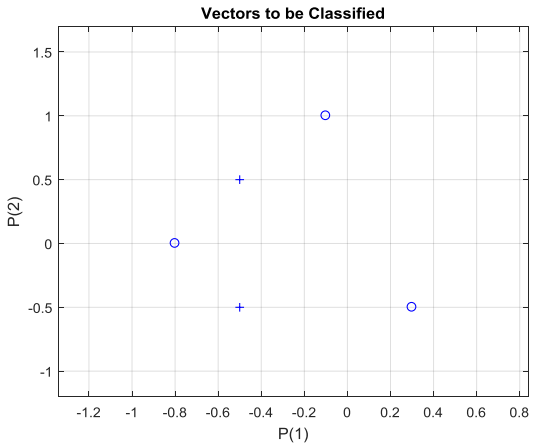



```
% A inconsistência presente no vetor de entrada não permitiu o
treinamento da RNA, independentemente do número de iterações. Portanto,
não houve classificação correta dos novos casos testados.
```

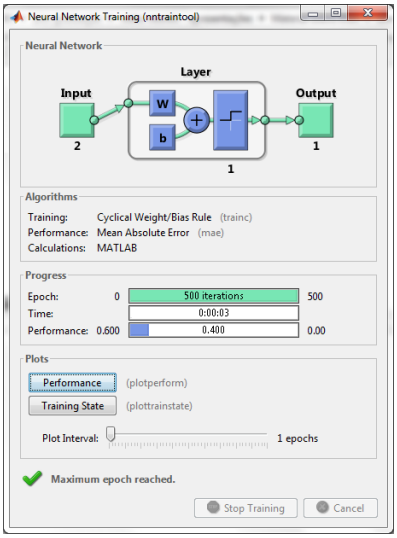
```
% Exercício 5 - Neste caso, alterar o número de iterações não tem efeito,
pois não há uma única reta que pode separar as classes em duas regiões
distintas

%apresentando os vetores de entrada

P = [ -0.5 -0.5 +0.3 -0.1 -0.8;
      -0.5 +0.5 -0.5 +1.0 +0.0 ]
T = [1 1 0 0 0]
```

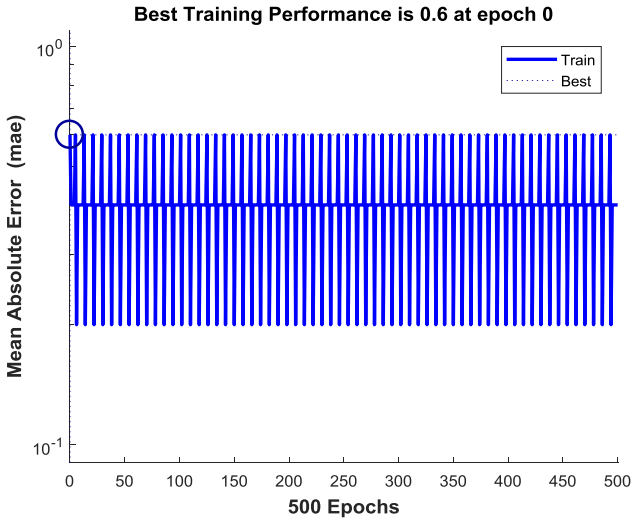
```
P =
   -0.5000   -0.5000    0.3000   -0.1000   -0.8000
   -0.5000    0.5000   -0.5000    1.0000         0

T =
     1     1     0     0     0
```

```
% plotando os vetores de entrada
plotpv(P,T)
grid
pause
```

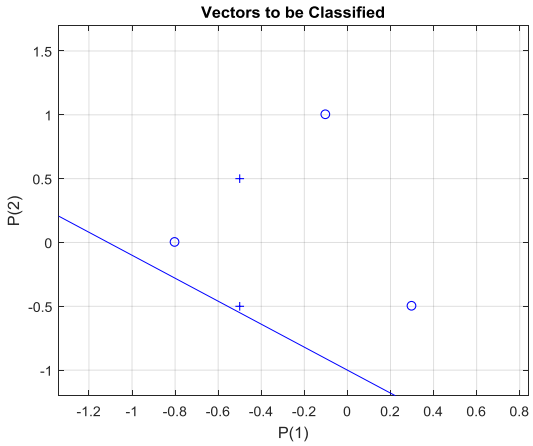



```
%definindo e treinando a RNA
net = newp([-1 1;-1 1],1);
net.trainParam.epochs = 500;
net = train(net,P,T);
```



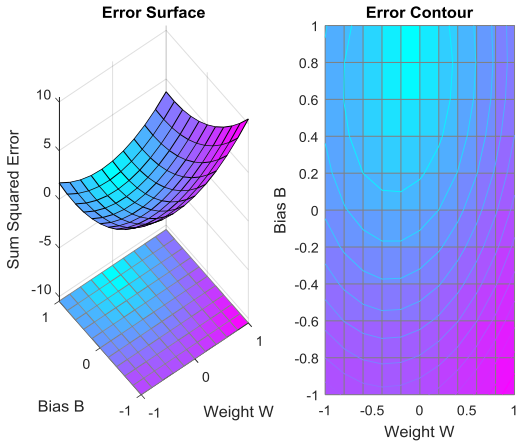



```
% parâmetros encontrados
net.IW{1}
net.b{1}
pause

pesos =    -0.9000    -1.0000

bias =     -1

%plotando a reta encontrada após o treinamento
figure
plotpv(P,T)
plotpc(net.IW{1},net.b{1})
grid
```

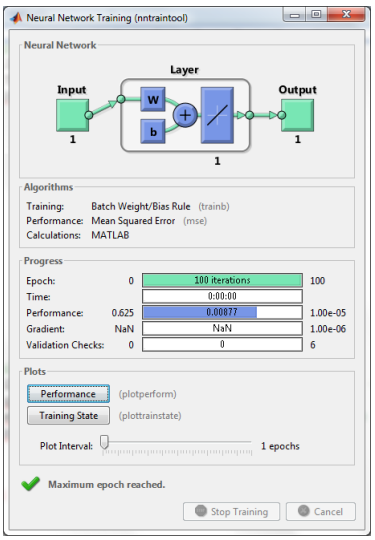


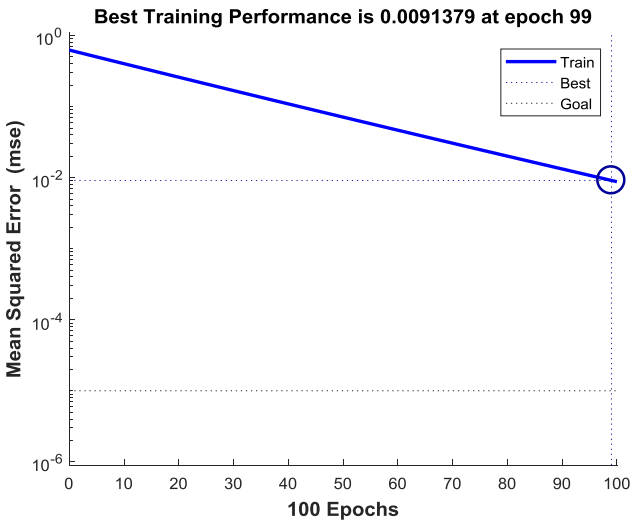

```
% alterar o numero de iterações não tem efeito
```

```
% Exercício 6 - Neste caso, a RNA deve encontrar os pesos e bias para
aproximar uma função, cujos dados são apresentados no conjunto de
treinamento. Treinando um neurônio linear p/ função y = -0.2273.x+0.7273

% Apresentando o conjunto de treinamento
P = [1.0 -1.2]
T = [0.5 1.0]

% Plotando a superfície de erro
w_range = -1:0.2:1;  b_range = -1:0.2:1;
ES = errsurf(P,T,w_range,b_range,'purelin');
plotes(w_range,b_range,ES);
```



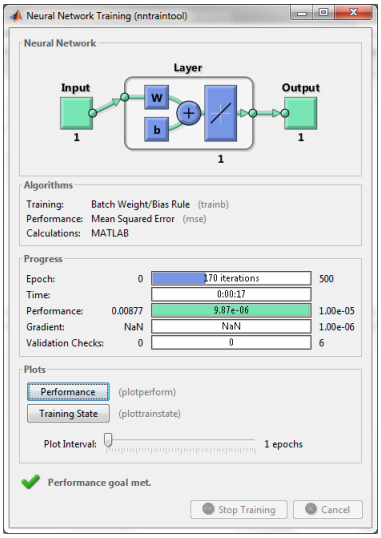

```
% Criando e treinando a RNA
net = newlin([-2 2],1); % neurônio linear com entradas de -2 a 2
net.trainParam.goal = .00001; % erro objetivo
net.trainParam.epochs = 100;  % máximo número de iterações
[net,tr] = train(net,P,T); % treinando a RNA
```

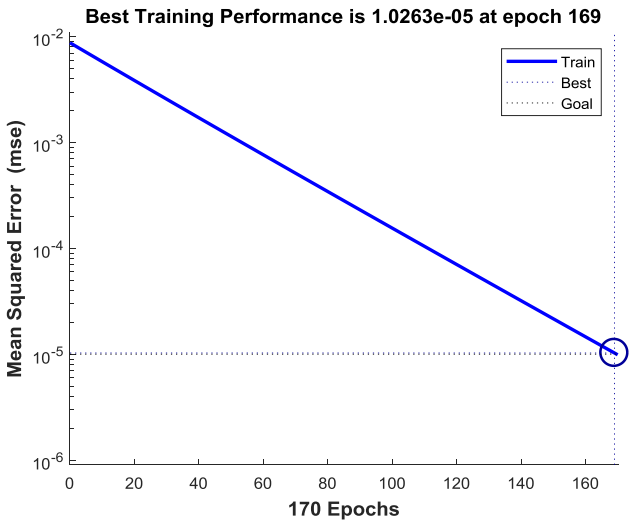





```
net.IW{1} %parâmetros encontrados com 100 iterações
net.b{1}
peso =  -0.2236
bias =  0.6341
```

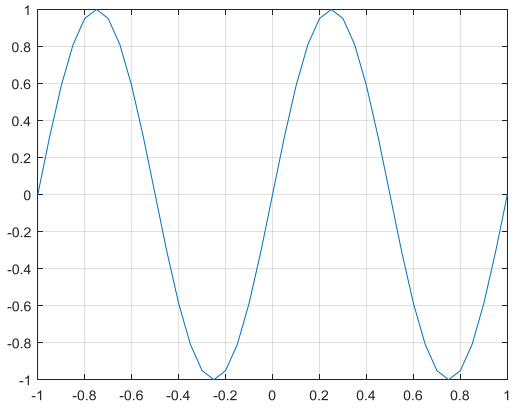
```
% Testando a RNA com o segundo ponto do vetor de treinamento
p = -1.2
a = sim(net, p)
pause

a = 0.9024 % O valor correto é 1 (ver conjunto de treinamento)

% Melhorando a resposta (aumentando de 100 para 500 iterações - reduzindo
o erro)

net.trainParam.goal = .00001;
net.trainParam.epochs = 500;
[net,tr] = train(net,P,T);
```



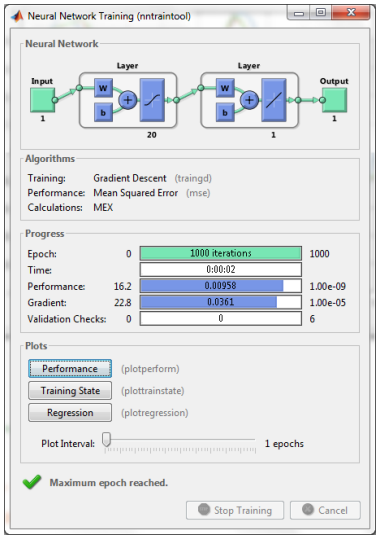


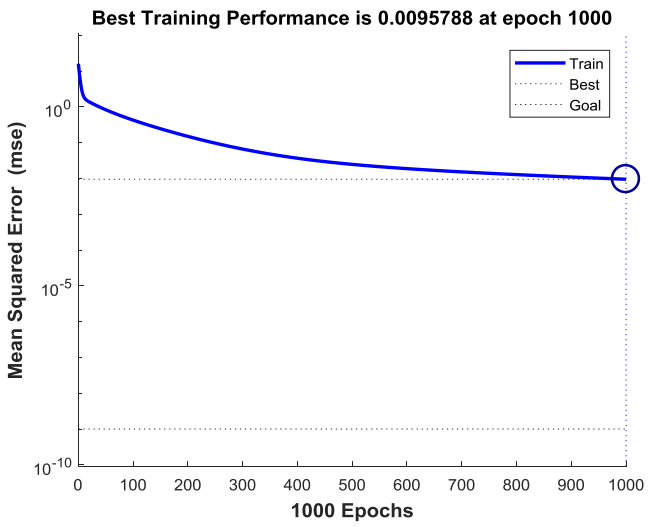

```
net.IW{1}% parâmetros encontrados com 500 iterações
net.b{1}

peso =  -0.2280 % se aproximando dos coeficientes da equação desejada
bias =   0.7242 % se aproximando dos coeficientes da equação desejada

% Novamente, testando a RNA com o segundo ponto do vetor de treinamento
p = -1.2
a = sim(net, p)
pause

a = 0.9977 %deveria ser 1 - bem próximo (bem melhor com 500 iterações)

% Testando a RNA com uma nova entrada

p = 1.8
a = sim(net, p)

a= 0.3138 % substituindo função conhecida observa-se a precisão

% equação = -0.2273.x+0.7273
% y = -0.2273 . 1.8 + 0.7273 = 0.31816
% quanto maior o número de iterações, maior a precisão
```

```
% Exercício 7 - Neste caso, a RNA deve encontrar os pesos e bias para
aproximar uma função senoidal, cujos dados são apresentados no conjunto
de treinamento.

% Define a função (os pontos) de treinamento
p = [-1:.05:1];
t = sin(2*pi*p);

% Plota a função que a RNA deve aprender
figure
plot (p,t)
grid
```

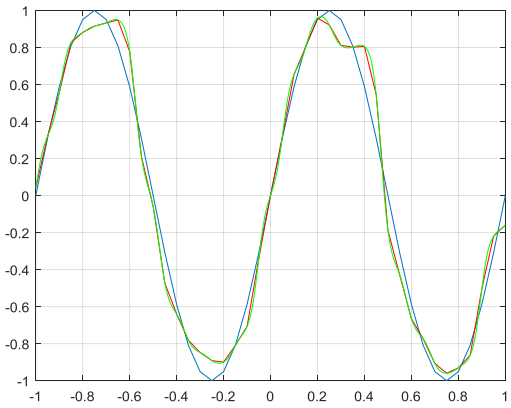



```
% definindo e treinando a RNA
net=newff(minmax(p),[20,1],{'tansig','purelin'},'traingd'); % define os
valores de entrada máximo e mínimo, números de neurônios nas camadas de
entrada e saída, funções de ativação e método de treinamento.
net.trainParam.goal = 1e-9;
net.trainParam.show = 10; % qual o intervalo para mostrar o MSE
net.trainParam.epochs = 1000;
net = init(net);
[net,tr]=train(net,p,t); % treina com base em p e t (armazena em net)
```

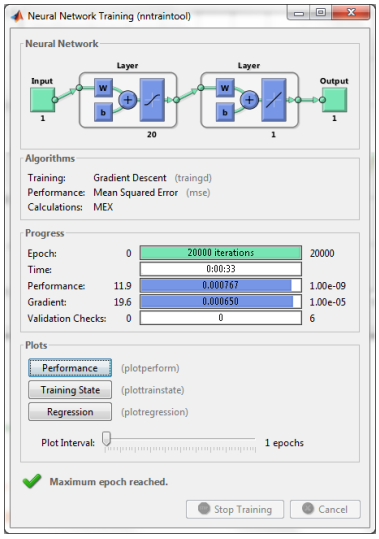


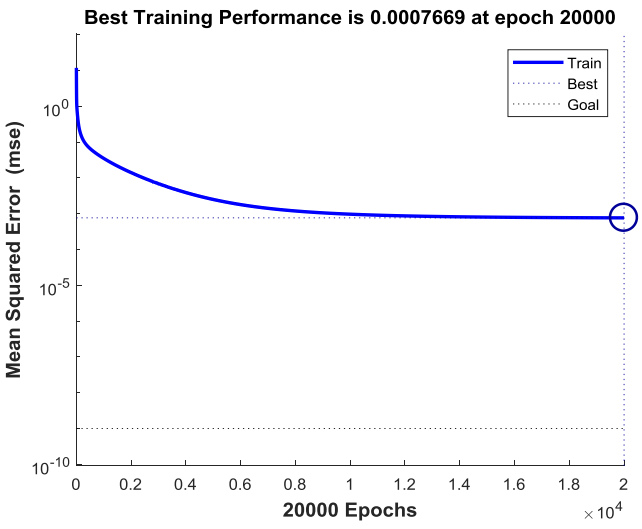


```
% Testando a RNA frente ao vetor de treinamento
figure
plot (p,t)
a1 = sim(net,p);
hold on
plot (p,a1,'r')
grid

% Testando a RNA diante de novos pontos
p1 = [-1:.005:1];
a2 = sim(net,p1);
plot (p1,a2,'g')
```

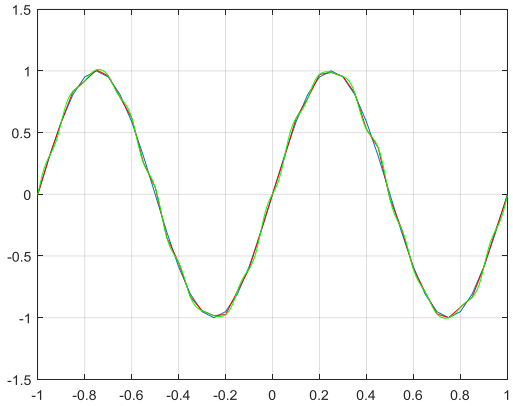



```
% Aumentando o numero de iterações de 1000 para 20000
% Desempenho no processo de treinamento
```

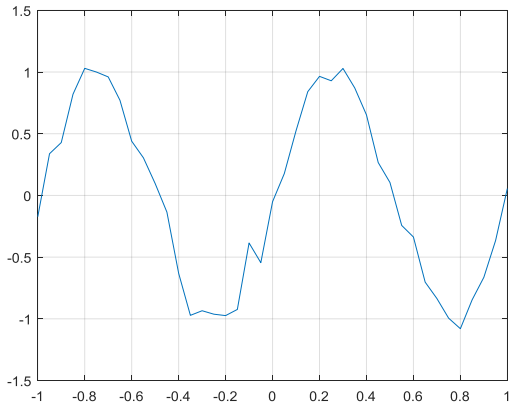




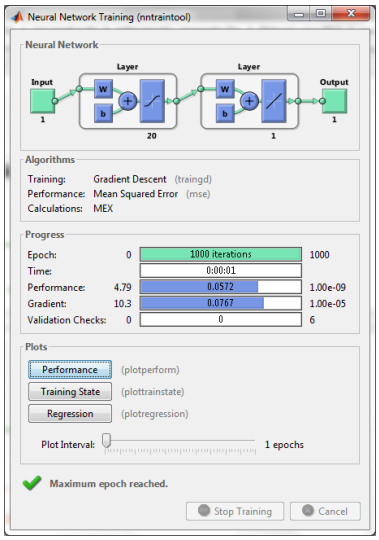
```
% Testando a RNA frente ao vetor de treinamento e novos pontos
```

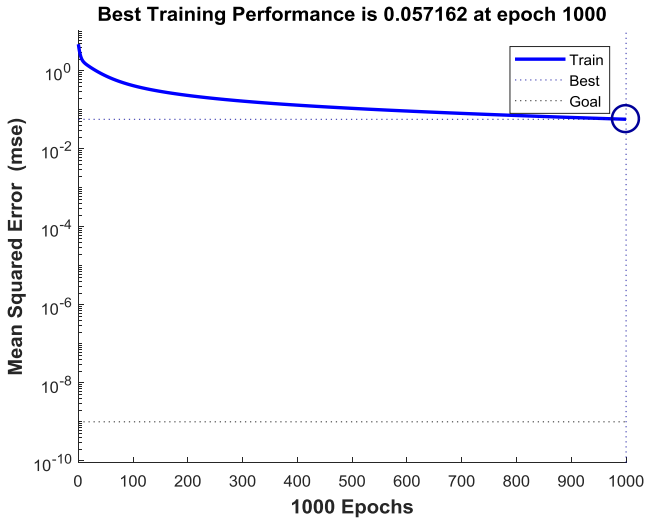



```
%============================================================
% NOVO CASO - treinando a rede com ruído no sinal

% definindo o sinal de entrada com ruído
p = [-1:.05:1];
t = sin(2*pi*p)+0.1*randn(size(p));

% Plotando o sinal de entrada
figure
plot (p,t)
grid
```

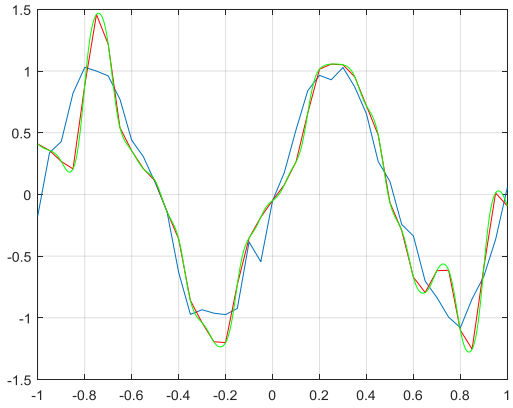



```
 % Definindo os parâmetros de treinando da RNA (comentários semelhantes ao
caso anterior - sem ruído)

net=newff(minmax(p),[20,1],{'tansig','purelin'},'traingd');
net.trainParam.goal = 1e-9;
net.trainParam.show = 10;
net.trainParam.epochs = 1000;
net = init(net);
[net,tr]=train(net,p,t);
```

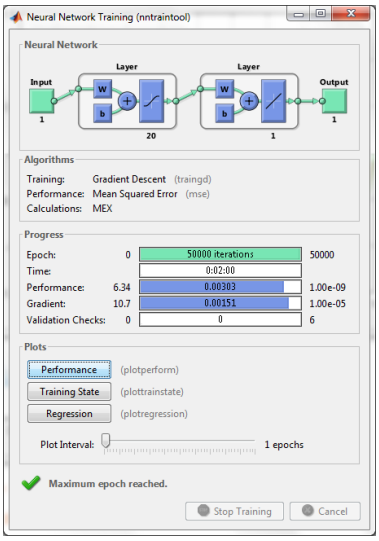


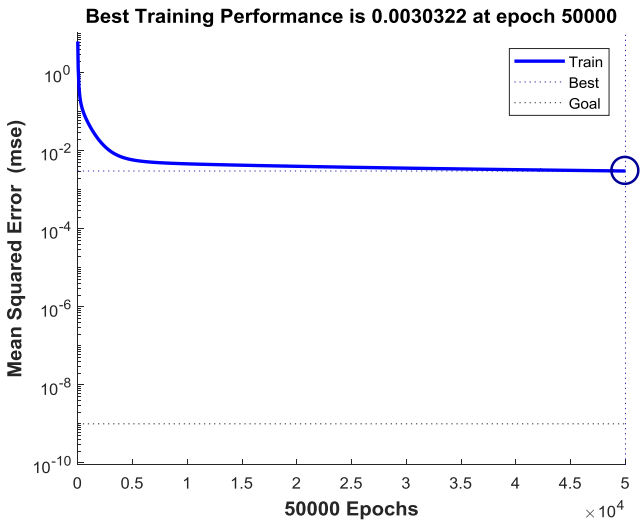



```
% Testando a RNA frente ao vetor de treinamento
a1 = sim(net,p);
figure
plot (p,t)
hold on
plot (p,a1,'r')
grid

% Testando a RNA diante de novos pontos
p1 = [-1:.0005:1];
a2 = sim(net,p1);
plot (p1,a2,'g')
```

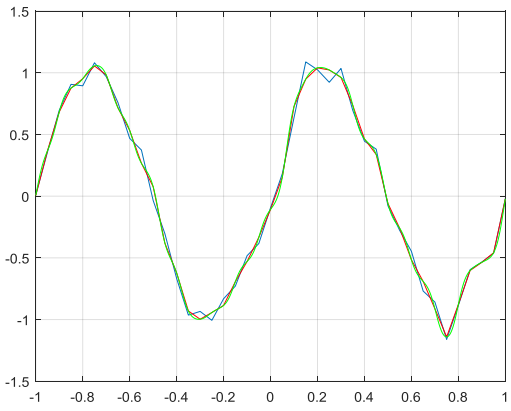


```
% Aumentando o numero de iterações de 1000 para 50000
% Desempenho no processo de treinamento
```



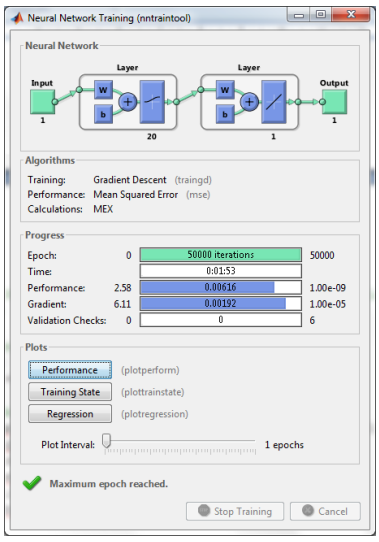


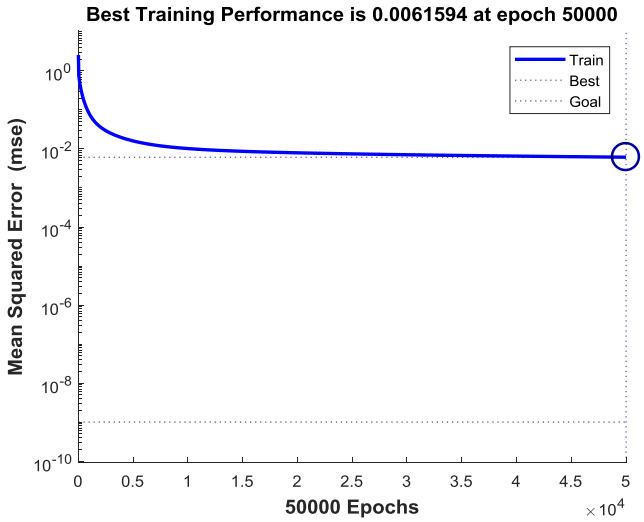

```
% Testando a RNA frente ao vetor de treinamento e novos pontos
```

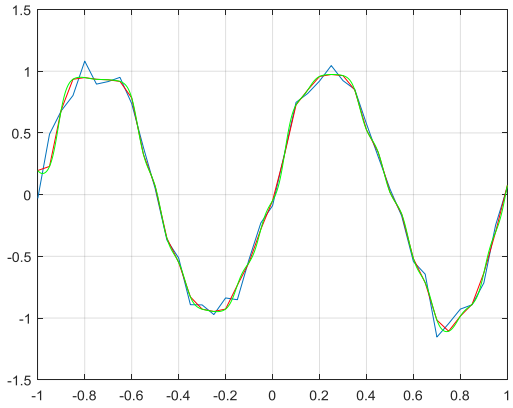


```
%=================================================================

% Tentativa de RNA com hardlim (neste caso o matlab não inicia o processo
de treinamento)

% treinando a rede com ruído no sinal
p = [-1:.05:1];
t = sin(2*pi*p)+0.1*randn(size(p));

% treinamento da rede
net=newff(minmax(p),[20,1],{'tansig','hardlim'},'traingd');
net.trainParam.goal = 1e-9;
net.trainParam.show = 10;
net.trainParam.epochs = 5000;
net = init(net);
[net,tr]=train(net,p,t);
```

```
%=====================================================================

% Tentativa de RNA com logsig (observa-se um desempenho um pouco inferior
ao tansig para 50000 iterações, porém uma tentativa com maior número de
iterações poderia resolver -???- é necessário testar)

% treinando a rede com ruído no sinal
p = [-1:.05:1];
t = sin(2*pi*p)+0.1*randn(size(p));

% treinamento da rede
net=newff(minmax(p),[20,1],{'logsig','purelin'},'traingd');
net.trainParam.goal = 1e-9;
net.trainParam.show = 10;
net.trainParam.epochs = 50000;
net = init(net);
[net,tr]=train(net,p,t);

% Testando a RNA frente ao vetor de treinamento
a1 = sim(net,p);
figure
plot (p,t)
hold on
plot (p,a1,'r')
grid

% Testando a RNA diante de novos pontos
p1 = [-1:.0005:1];
a2 = sim(net,p1);
plot (p1,a2,'g')
```






1.5
1
0.5
0
-0.5
-1
-1.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1