

Toolbox de Redes Neurais

Matrix-Laboratory **MATLAB**

Sumário

.Introdução ao Matlab

.Toolbox de Redes Neurais no MATLAB

- NNTool;

- Linha de comando;

.Estudo de Casos

- Análise de Crédito Bancário (Classificação);

- Sensação Térmica (Previsão).

Introdução ao ambiente

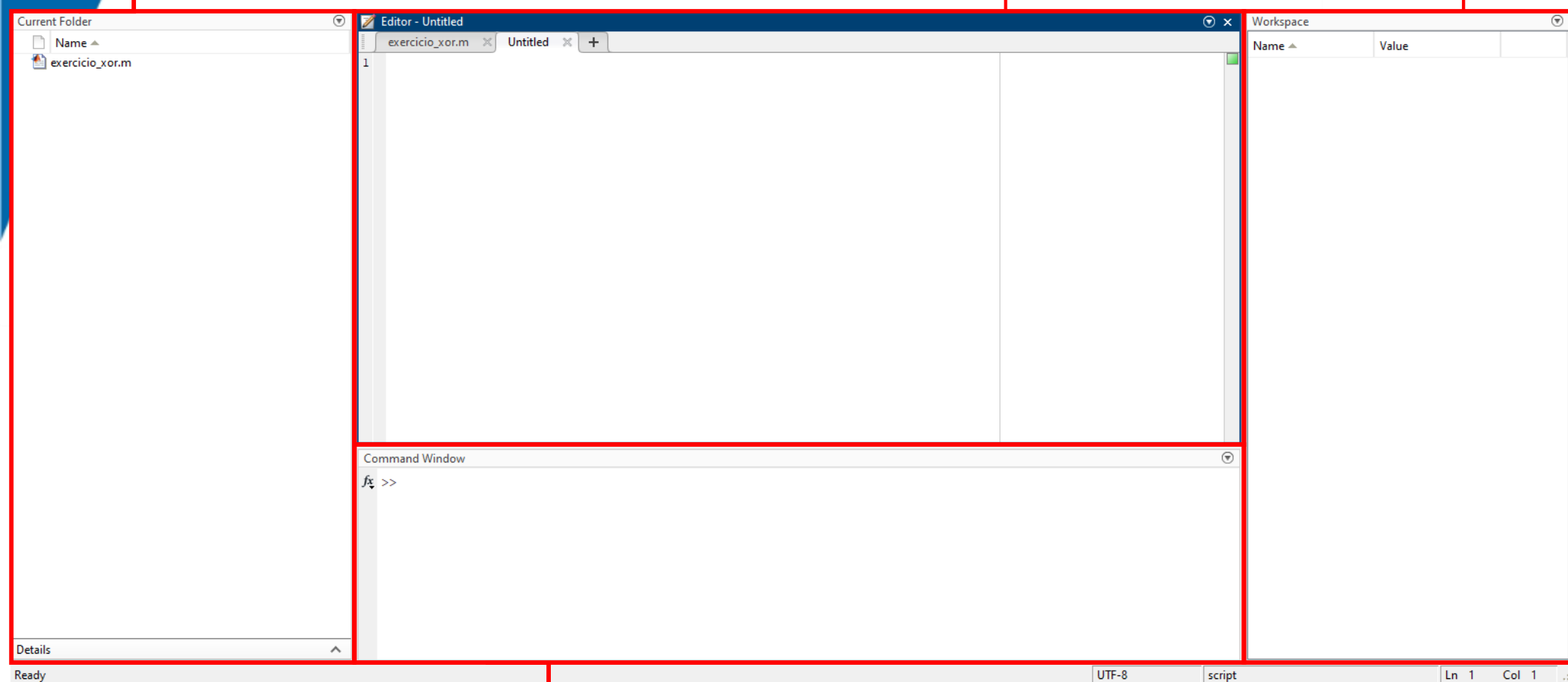
- **MATLAB - MATrix LABoratory**
- Programação baseada em Matrizes;
- Vetores e escalares são tratados como matrizes ($1 \times N$, $N \times 1$, $N \times N$).

Introdução ao Matlab

Diretório atual

Editor de script

Workspace



Janela de comando

Exemplo: Quadrado Mágico

Command Window

```
>> help magic
magic Magic square.
magic(N) is an N-by-N matrix constructed from the integers
1 through N^2 with equal row, column, and diagonal sums.
Produces valid magic squares for all N > 0 except N = 2.
```

[Documentation for magic](#)

```
>> n = magic(5)
```

n =

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

```
>> sum(n)
```

ans =

65	65	65	65	65
----	----	----	----	----

```
>> sum(n,2)
```

ans =

65
65
65
65
65



Matriz $N \times N$ construída a partir de inteiros de cuja soma das colunas são iguais às somas das linhas.

Definindo uma matriz

- Elementos de uma linha são separados por **espaços** ou **vírgula**;
- O final de cada linha é indicado por um **ponto-e-vírgula**;
- A lista de elementos é delimitada por **colchetes []**.

```
Command Window
>> M = [5 2 1; 3 3 10; 4 0 8; 7 5 3; 2 6 0]

M =

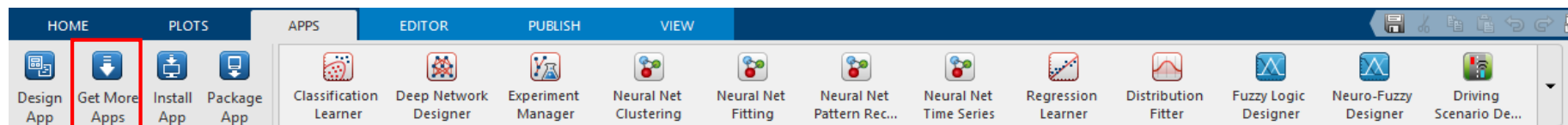
     5     2     1
     3     3    10
     4     0     8
     7     5     3
     2     6     0
```

Toolbox de Redes Neurais

- .Definição do problema;**
- .Inicialização da rede;**
- .Parâmetros de treinamento;**
- .Treinamento da rede;**
- .Teste da rede.**

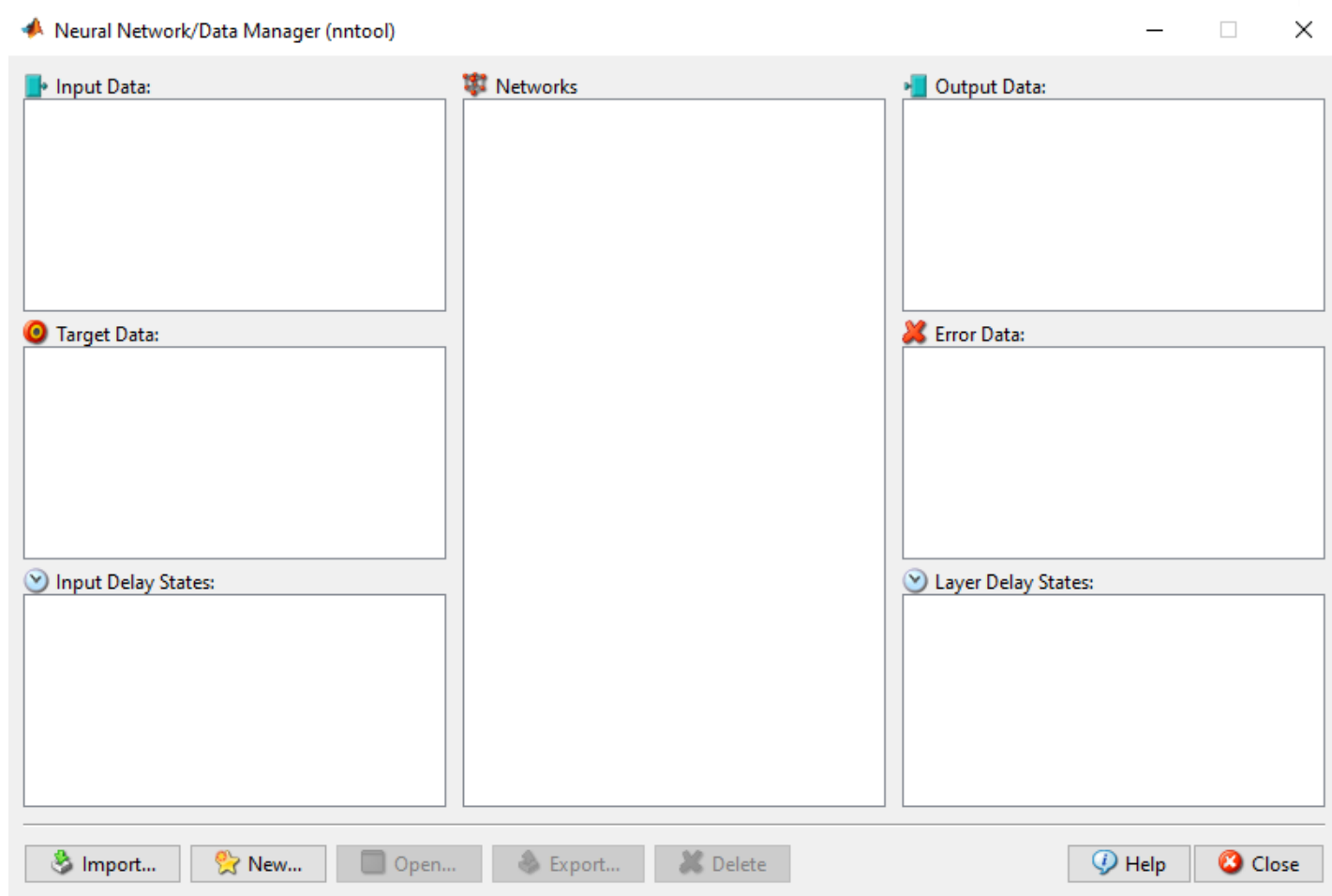
Toolbox de Redes Neurais

- O NNTool é um Toolbox do MATLAB para implementação de Redes Neurais.
- A ferramenta é parte do pacote 'Deep Learning Toolbox', que pode ser instalado diretamente pelo MATLAB



Toolbox de Redes Neurais

Tela inicial do NNTool



Toolbox de Redes Neurais

Exemplo: Problema do OU Exclusivo (XOR)

I1	I2	O
0	0	0
0	1	1
1	0	1
1	1	0

Tabela verdade da porta XOR

Command Window

```
>> P = [0 1 0 1; 0 0 1 1]
```

```
P =
```

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

```
>> T = [0 1 1 0];
```

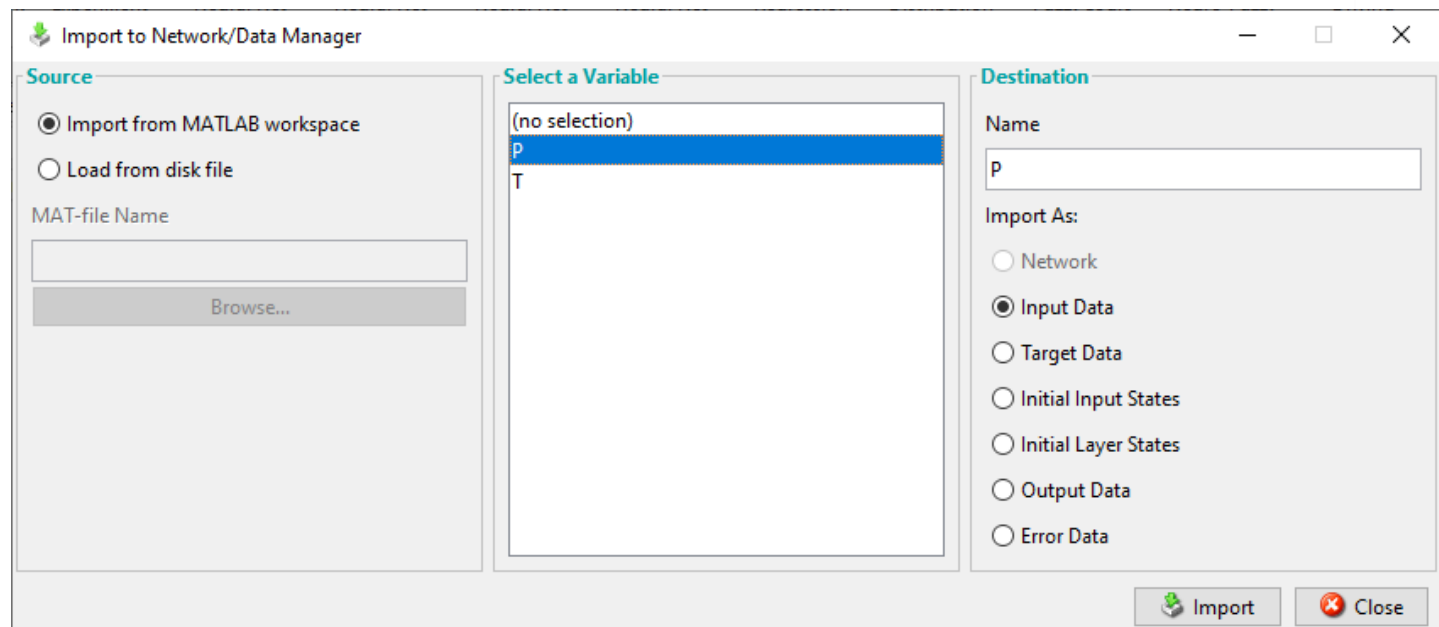
Lembrete

linhas – atributos

colunas - registros

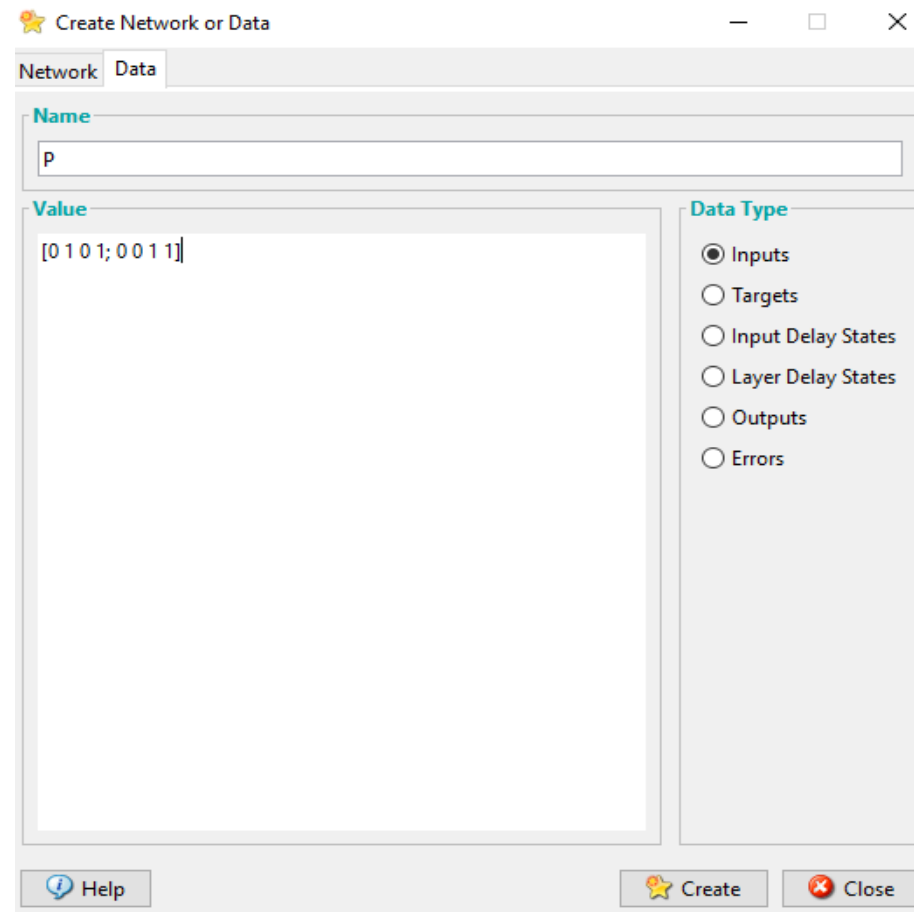
Toolbox de Redes Neurais

Os registros do *dataset* podem ser importados diretamente do *Workspace* ou adicionados manualmente.



Toolbox de Redes Neurais

Para inserção manual dos dados, deve ser escrita a matriz com os dados



Toolbox de Redes Neurais

Criação da Rede Neural

Create Network or Data

Network Data

Name: net

Network Properties

Network Type: Feed-forward backprop

Input data: P

Target data: T

Training function: TRAINLM

Adaption learning function: LEARNGDM

Performance function: MSE

Number of layers: 2

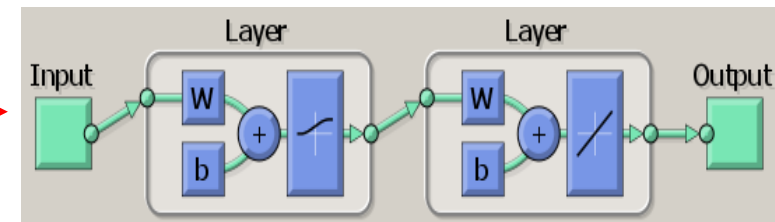
Properties for: Layer 1

Number of neurons: 10

Transfer Function: TANSIG

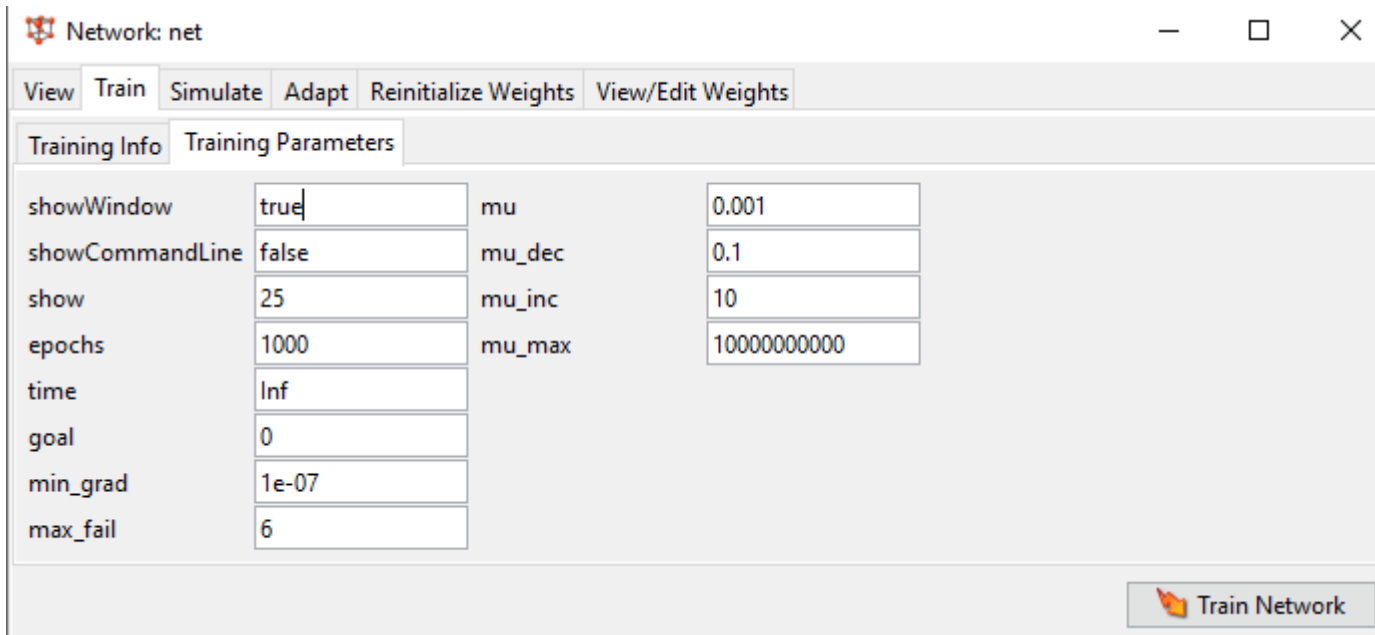
View Restore Defaults

Help Create Close



Toolbox de Redes Neurais

Selecionando o modelo na tela inicial, é possível modificar os parâmetros.



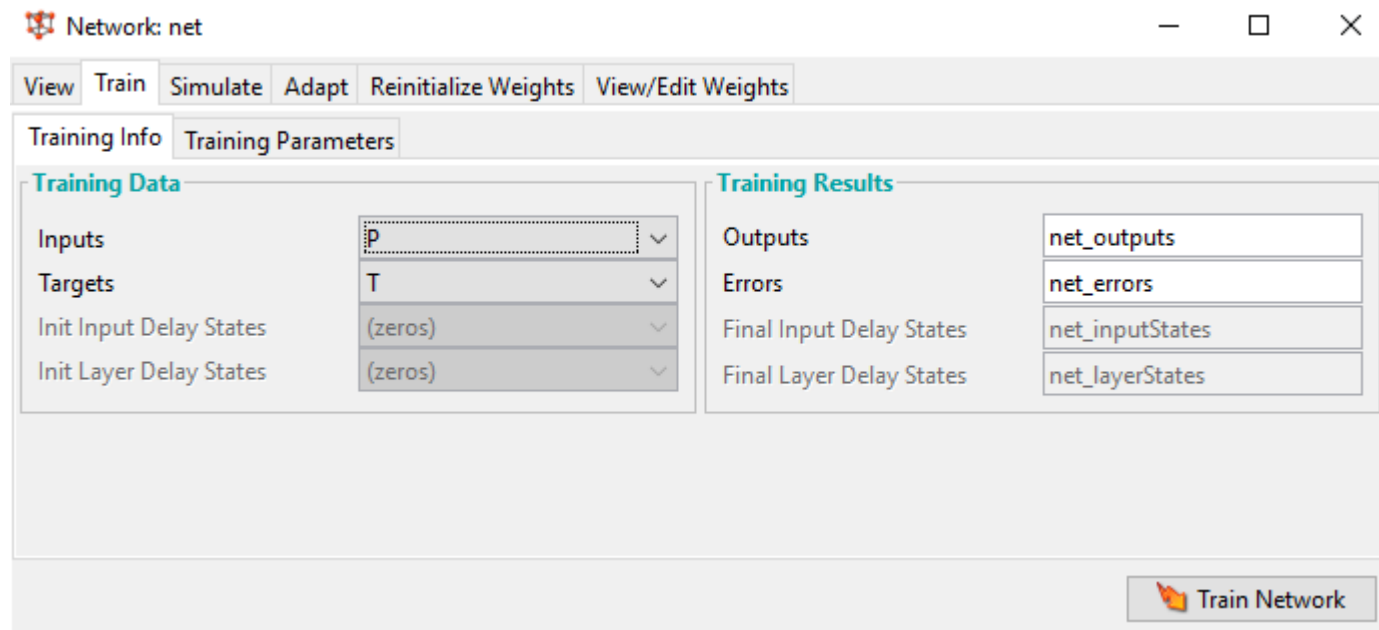
The image shows the 'Network: net' window in the MATLAB Neural Network Toolbox. The 'Training Parameters' tab is active, displaying a list of parameters and their values. The parameters are arranged in two columns. The first column includes 'showWindow', 'showCommandLine', 'show', 'epochs', 'time', 'goal', 'min_grad', and 'max_fail'. The second column includes 'mu', 'mu_dec', 'mu_inc', and 'mu_max'. Each parameter has a corresponding input field. The 'Train Network' button is located at the bottom right of the window.

Parameter	Value
showWindow	true
showCommandLine	false
show	25
epochs	1000
time	Inf
goal	0
min_grad	1e-07
max_fail	6
mu	0.001
mu_dec	0.1
mu_inc	10
mu_max	10000000000

Train Network

Toolbox de Redes Neurais

Selecionando o modelo na tela inicial, é possível modificar os parâmetros.



Toolbox de Redes Neurais

• Além da Interface gráfica, também é possível criar um modelo de Rede Neural usando Linha de Comando.

```
net = feedforwardnet(n_hidden); %Inicialização da Rede

net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';

%Opcoes para função de treinamento da rede
%traingd - Gradient descent Backpropagation
%traingdm - Gradient descent backpropagation with momentum
%traingda - Gradient descent Backpropagation with adaptive ratio
%traingdx - Gradient descent backpropagation with momentum and adaptive ratio
net.trainFcn = 'traingd';

%Alguns parametros para modificar o modelo
%Para ver todos os parâmetros que permitem em uma função de treinamento:
%help traingd (ou qualquer função de treinamento que queira)
net.trainParam.epochs = 100; %Épocas para o treinamento do modelo
net.trainParam.goal = 0; %Erro objetivo do modelo
net.trainParam.lr = 0.1; %Taxa de aprendizado (default: 0.01)
net.trainParam.mc = 0.9; %Constante de momento (default: 0.9)
net.trainParam.show = 25; %Parametro para mostrar a distância entre épocas para mostrar o resultado.
net.trainParam.showWindow = true; %Mostrar a tela de treinamento
```


Toolbox de Redes Neurais

```
n_hidden = 2; %Quantidade de neurônios na camada escondida

net = feedforwardnet(n_hidden); %Inicialização da Rede

net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';

%Opcoes para função de treinamento da rede
net.trainFcn = 'traingd';

%Alguns parametros para modificar o modelo
%Para ver todos os parâmetros que permitem em uma função de treinamento:
%help traingd (ou qualquer função de treinamento que queira)
net.trainParam.epochs = 100; %Épocas para o treinamento do modelo
net.trainParam.goal = 0; %Erro objetivo do modelo
net.trainParam.lr = 0.1; %Taxa de aprendizado (default: 0.01)
net.trainParam.mc = 0.9; %Constante de momento (default: 0.9)
net.trainParam.show = 25; %Parametro de épocas para mostrar o resultado.
net.trainParam.showWindow = true; %Mostrar a tela de treinamento

%Divisao da base de dados, caso precise dividir entre treinamento,
%validação e teste.

%Possiveis divisoes: dividerand, divideind, divideblock
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 1; %Proporcao para conjunto de treinamento
net.divideParam.valRatio = 0.0; %Proporcao para conjunto de validacao
net.divideParam.testRatio = 0.0; %Porporcao para conjunto de teste
```

Define as camadas escondidas

- 1 camada oculta:
`feedforwardnet(2)`
- 2 camadas ocultas:
`feedforwardnet([3 2])`

Toolbox de Redes Neurais

```
n_hidden = 2; %Quantidade de neurônios na camada escondida

net = feedforwardnet(n_hidden); %Inicialização da Rede

net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';

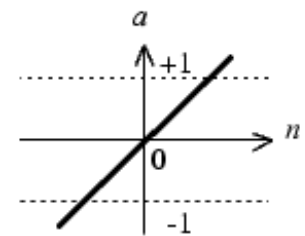
%Opcoes para função de treinamento da rede
net.trainFcn = 'traingd';

%Alguns parametros para modificar o modelo
%Para ver todos os parâmetros que permitem em uma função de treinamento:
%help traingd (ou qualquer função de treinamento que queira)
net.trainParam.epochs = 100; %Épocas para o treinamento do modelo
net.trainParam.goal = 0; %Erro objetivo do modelo
net.trainParam.lr = 0.1; %Taxa de aprendizado (default: 0.01)
net.trainParam.mc = 0.9; %Constante de momento (default: 0.9)
net.trainParam.show = 25; %Parametro de épocas para mostrar o resultado.
net.trainParam.showWindow = true; %Mostrar a tela de treinamento

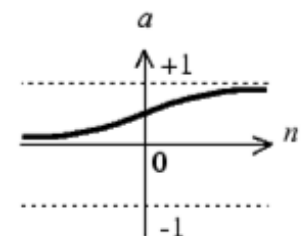
%Divisao da base de dados, caso precise dividir entre treinamento,
%validação e teste.

%Possiveis divisoes: dividerand, divideind, divideblock
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 1; %Proporcao para conjunto de treinamento
net.divideParam.valRatio = 0.0; %Proporcao para conjunto de validacao
net.divideParam.testRatio = 0.0; %Proporcao para conjunto de teste
```

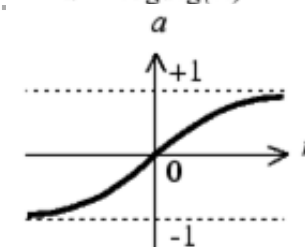
Funções de ativação



$$a = \text{purelin}(n)$$



$$a = \text{logsig}(n)$$



$$a = \text{tansig}(n)$$

Toolbox de Redes Neurais

```
n_hidden = 2; %Quantidade de neurônios na camada escondida

net = feedforwardnet(n_hidden); %Inicialização da Rede

net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';

%Opcoes para função de treinamento da rede
net.trainFcn = 'traingd';

%Alguns parametros para modificar o modelo
%Para ver todos os parâmetros que permitem em uma função de treinamento:
%help traingd (ou qualquer função de treinamento que queira)
net.trainParam.epochs = 100; %Épocas para o treinamento do modelo
net.trainParam.goal = 0; %Erro objetivo do modelo
net.trainParam.lr = 0.1; %Taxa de aprendizado (default: 0.01)
net.trainParam.mc = 0.9; %Constante de momento (default: 0.9)
net.trainParam.show = 25; %Parametro de épocas para mostrar o resultado.
net.trainParam.showWindow = true; %Mostrar a tela de treinamento

%Divisao da base de dados, caso precise dividir entre treinamento,
%validação e teste.

%Possiveis divisoes: dividerand, divideind, divideblock
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 1; %Proporcao para conjunto de treinamento
net.divideParam.valRatio = 0.0; %Proporcao para conjunto de validacao
net.divideParam.testRatio = 0.0; %Porporcao para conjunto de teste
```

Algoritmos de treinamento

- **traingd** - Gradient descent backpropagation;
- **traingdm** - Gradient descent backpropagation with momentum;
- **traingda** - Gradient descent backpropagation with adaptive ratio;

Toolbox de Redes Neurais

```
n_hidden = 2; %Quantidade de neurônios na camada escondida

net = feedforwardnet(n_hidden); %Inicialização da Rede

net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';

%Opcoes para função de treinamento da rede
net.trainFcn = 'traingd';

%Alguns parametros para modificar o modelo
%Para ver todos os parâmetros que permitem em uma função de treinamento:
%help traingd (ou qualquer função de treinamento que queira)
net.trainParam.epochs = 100; %Épocas para o treinamento do modelo
net.trainParam.goal = 0; %Erro objetivo do modelo
net.trainParam.lr = 0.1; %Taxa de aprendizado (default: 0.01)
net.trainParam.mc = 0.9; %Constante de momento (default: 0.9)
net.trainParam.show = 25; %Parametro de épocas para mostrar o resultado.
net.trainParam.showWindow = true; %Mostrar a tela de treinamento

%Divisao da base de dados, caso precise dividir entre treinamento,
%validação e teste.

%Possiveis divisoes: dividerand, divideind, divideblock
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 1; %Proporcao para conjunto de treinamento
net.divideParam.valRatio = 0.0; %Proporcao para conjunto de validacao
net.divideParam.testRatio = 0.0; %Porporcao para conjunto de teste
```

Parâmetros do modelo

- Depende da função escolhida para o treinamento da rede.
- Para observar os parâmetros que podem ser modificados, é recomendável usar o comando help do MATLAB com a função de treinamento.

Exemplo: 'help traingdm'

Toolbox de Redes Neurais

```
n_hidden = 2; %Quantidade de neurônios na camada escondida

net = feedforwardnet(n_hidden); %Inicialização da Rede

net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';

%Opcoes para função de treinamento da rede
net.trainFcn = 'traingd';

%Alguns parametros para modificar o modelo
%Para ver todos os parâmetros que permitem em uma função de treinamento:
%help traingd (ou qualquer função de treinamento que queira)
net.trainParam.epochs = 100; %Épocas para o treinamento do modelo
net.trainParam.goal = 0; %Erro objetivo do modelo
net.trainParam.lr = 0.1; %Taxa de aprendizado (default: 0.01)
net.trainParam.mc = 0.9; %Constante de momento (default: 0.9)
net.trainParam.show = 25; %Parametro de épocas para mostrar o resultado.
net.trainParam.showWindow = true; %Mostrar a tela de treinamento

%Divisao da base de dados, caso precise dividir entre treinamento,
%validação e teste.

%Possiveis divisoes: dividerand, divideind, divideblock
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 1; %Proporcao para conjunto de treinamento
net.divideParam.valRatio = 0.0; %Proporcao para conjunto de validacao
net.divideParam.testRatio = 0.0; %Porporcao para conjunto de teste
```

Divisão de conjuntos

- **Indices definidos;**

```
net.divideFcn = 'divideind';
net.divideParam.trainInd = indTreino;
net.divideParam.valInd = indValidacao;
net.divideParam.testInd = indTeste;
```

- **Indices aleatórios;**

```
net.divideFcn = 'dividerand';
net.divideParam.trainRatio: 0.6000;
net.divideParam.valRatio: 0.2000;
net.divideParam.testRatio: 0.2000;
```

- **Blocos de indices;**

```
net.divideFcn = 'divideblock';
net.divideParam.trainRatio: 0.6000;
net.divideParam.valRatio: 0.2000;
net.divideParam.testRatio: 0.2000;
```

Toolbox de Redes Neurais

```
%Para ver as funções de performance: help nnperformance
net.performFcn = 'mse';

%Pode ser usado para configurar previamente a rede baseado em In/Out
net = configure(net,P,T);

view(net)

%Treinamento da rede
[net, tr] = train(net,P,T);

%Simulação da rede
C = sim(net, P);
%Equivalente ao comando acima
O = net(P);

figure
plotperform(tr)
figure
plotconfusion(T,O)
```

Métricas de desempenho

- **sse** – Sum of Squared Error;
- **sae** – Sum of Absolute Error;
- **mae** – Mean Absolute Error;
- **mse** – Mean Squared Error;

Toolbox de Redes Neurais

```
%Para ver as funções de performance: help nnperformance
net.performFcn = 'mse';

%Pode ser usado para configurar previamente a rede baseado em In/Out
net = configure(net,P,T);

view(net)

%Treinamento da rede
[net, tr] = train(net,P,T);

%Simulação da rede
C = sim(net, P);
%Equivalente ao comando acima
O = net(P);

figure
plotperform(tr)
figure
plotconfusion(T,O)
```

**Visualização do modelo
criado**

Toolbox de Redes Neurais

```
%Para ver as funções de performance: help nnperformance
net.performFcn = 'mse';

%Pode ser usado para configurar previamente a rede baseado em In/Out
net = configure(net,P,T);

view(net)

%Treinamento da rede
[net, tr] = train(net,P,T);

%Simulação da rede
C = sim(net, P);
%Equivalente ao comando acima
O = net(P);

figure
plotperform(tr)
figure
plotconfusion(T,O)
```

Treinamento da Rede Neural

Toolbox de Redes Neurais

```
%Para ver as funções de performance: help nnperformance
net.performFcn = 'mse';

%Pode ser usado para configurar previamente a rede baseado em In/Out
net = configure(net,P,T);

view(net)

%Treinamento da rede
[net, tr] = train(net,P,T);

%Simulação da rede
C = sim(net, P);
%Equivalente ao comando acima
O = net(P);

figure
plotperform(tr)
figure
plotconfusion(T,O)
```

Utilização da Rede Neural treinada

- Pode ser executada através do comando 'sim' ou 'net'.

Toolbox de Redes Neurais

```
%Para ver as funções de performance: help nnperformance
net.performFcn = 'mse';

%Pode ser usado para configurar previamente a rede baseado em In/Out
net = configure(net,P,T);

view(net)

%Treinamento da rede
[net, tr] = train(net,P,T);

%Simulação da rede
C = sim(net, P);
%Equivalente ao comando acima
O = net(P);

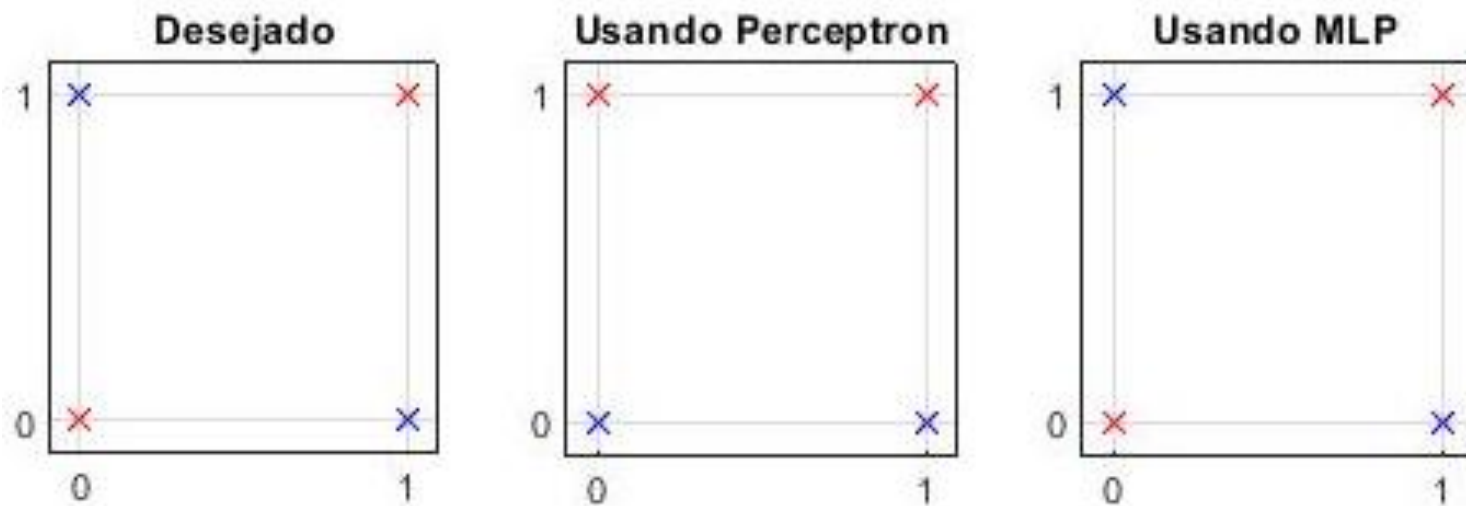
figure
plotperform(tr)
figure
plotconfusion(T,O)
```

Visualização dos resultados

- **plotperform** – gráfico do desempenho em função das épocas.
- **plotconfusion** – matriz de confusão dos resultados.

Toolbox de Redes Neurais

Visualização do resultado usando Perceptron e Multilayer Perceptron







Estudo de Caso

.Análise de Crédito Bancário

-Base de Dados: contém informações sobre 1500 clientes: 715 pagadores; 785 não pagadores.

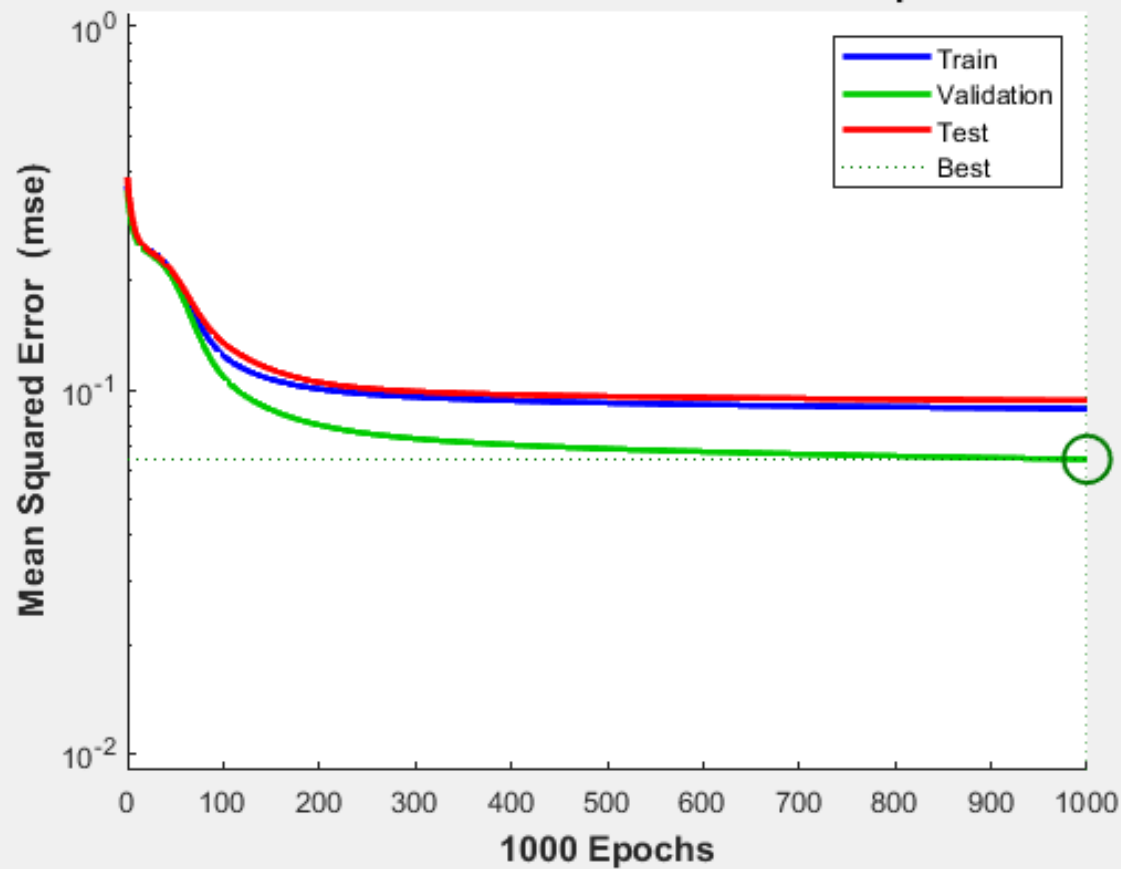
```
AnaliseCred = importdata('Classificacao/treino01.txt');  
dados = AnaliseCred.data;
```

```
P = dados(1:end,1:11)';  
T = dados(1:end,12)';
```

Workspace	
Name ▲	Value
 AnaliseCred	1x1 struct
 dados	1500x12 double
 P	11x1500 double
 T	1x1500 double

Estudo de Caso

Best Validation Performance is 0.064497 at epoch 1000



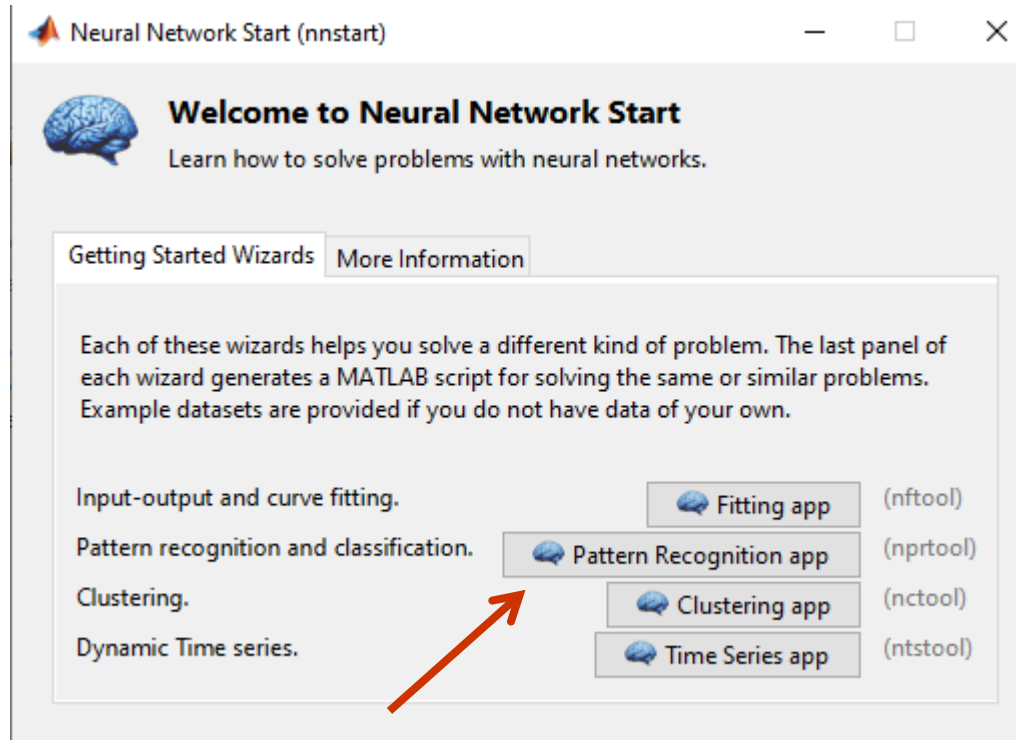
Confusion Matrix

Output Class	Target Class			
	0	1		
0	761 50.7%	124 8.3%	86.0%	14.0%
1	24 1.6%	591 39.4%	96.1%	3.9%
	0	1	96.9%	3.1%
			82.7%	17.3%
			90.1%	9.9%

Estudo de Caso


.Análise de Crédito Bancário

-NNstart



Estudo de Caso

•Análise de Crédito Bancário


**Select Data**
What inputs and targets define your pattern recognition problem?

Get Data from Workspace
Input data to present to the network.
Inputs: p ...
Target data defining desired network output.
Targets: T ...
Samples are: ☒ Matrix columns ☐ Matrix rows

Summary
Inputs 'P' is a 11x1500 matrix, representing static data: 1500 samples of 11 elements.
Targets 'T' is a 1x1500 matrix, representing static data: 1500 samples of 1 element.

Estudos de Caso


.Análise de Crédito Bancário






Validation and Test Data


Set aside some samples for validation and testing.


Select Percentages


 Randomly divide up the 1500 samples:


 Training:	70%	1050 samples
 Validation:	15% ▾	225 samples
 Testing:	15% ▾	225 samples

Explanation

 Three Kinds of Samples:


 Training:
These are presented to the network during training, and the network is adjusted according to its error.

 Validation:
These are used to measure network generalization, and to halt training when generalization stops improving.

 Testing:
These have no effect on training and so provide an independent measure of network performance during and after training.

Estudo de Caso

.Análise de Crédito Bancário




Train Network

Train the network to classify the inputs according to the targets.







Train Network

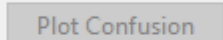
Train using scaled conjugate gradient backpropagation. (trainscg)

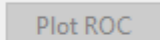
 Train

Training automatically stops when generalization stops improving, as indicated by an increase in the cross-entropy error of the validation samples.


Results

	 Samples	 CE	 %E
 Training:	1050	-	-
 Validation:	225	-	-
 Testing:	225	-	-


 Plot Confusion

 Plot ROC


Notes



Training multiple times will generate different results due to different initial conditions and sampling.



Minimizing Cross-Entropy results in good classification. Lower values are better. Zero means no error.




Percent Error indicates the fraction of samples which are misclassified. A value of 0 means no misclassifications, 100 indicates maximum misclassifications.

Estudo de Caso

•Análise de Crédito Bancário

Neural Network Training (nntraintool)

Neural Network



Algorithms

Data Division: Random (dividerand)
 Training: Scaled Conjugate Gradient (trainscg)
 Performance: Cross-Entropy (crossentropy)
 Calculations: MEX

Progress

Epoch:	0	34 iterations	1000
Time:		0:00:00	
Performance:	2.20	0.258	0.00
Gradient:	1.61	0.0299	1.00e-06
Validation Checks:	0	6	6

Plots

- ☒ Performance (plotperform)
- ☐ Training State (plottrainstate)
- ☐ Error Histogram (ploterrhist)
- ☐ Confusion (plotconfusion)
- ☐ Receiver Operating Characteristic (plotroc)

Plot Interval: 1 epochs

Opening Performance Plot

Stop Training Cancel

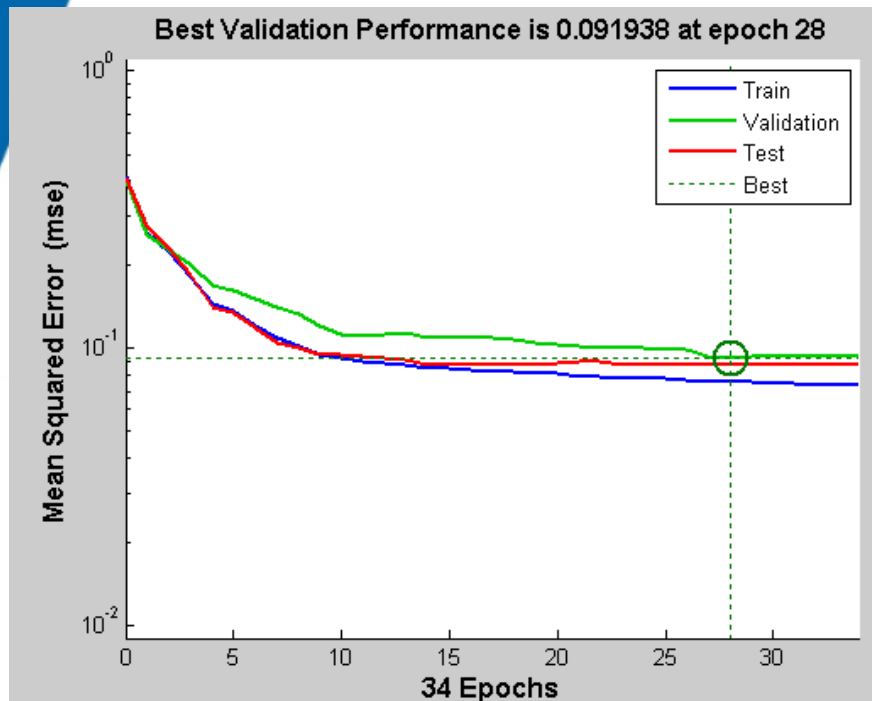
Results

	Samples	CE	%E
Training:	1050	5.94677e-1	9.33333e-0
Validation:	225	1.15662e-0	10.22222e-0
Testing:	225	1.19009e-0	11.55555e-0

Plot Confusion Plot ROC

Estudo de Caso

•Análise de Crédito Bancário



Training Confusion Matrix

Output Class \ Target Class	0	1	
0	531 50.6%	77 7.3%	87.3% 12.7%
1	23 2.2%	419 39.9%	94.8% 5.2%
	95.8% 4.2%	84.5% 15.5%	90.5% 9.5%

Validation Confusion Matrix

Output Class \ Target Class	0	1	
0	116 51.6%	22 9.8%	84.1% 15.9%
1	4 1.8%	83 36.9%	95.4% 4.6%
	96.7% 3.3%	79.0% 21.0%	88.4% 11.6%

Test Confusion Matrix

Output Class \ Target Class	0	1	
0	108 48.0%	19 8.4%	85.0% 15.0%
1	3 1.3%	95 42.2%	96.9% 3.1%
	97.3% 2.7%	83.3% 16.7%	90.2% 9.8%

All Confusion Matrix

Output Class \ Target Class	0	1	
0	755 50.3%	118 7.9%	86.5% 13.5%
1	30 2.0%	597 39.8%	95.2% 4.8%
	96.2% 3.8%	83.5% 16.5%	90.1% 9.9%