Redes Neurais no MATLAB 6.1

Redes Neurais no MATLAB

- Duas formas de utilização:
  - Linhas de comando, e m-files
  - Interface gráfica (NNTool)

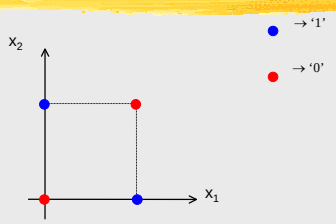
Redes Neurais no MATLAB

- Duas formas de utilização:
  - Linhas de comando, e m-files
  - Interface gráfica (NNTool)

Passos para a Criação de uma RN

- Definir os padrões
- Inicializar a rede
- Definir os parâmetros de treinamento
- Treinar a rede
- Testar a rede

O Problema do OU Exclusivo



X_1	X_2	valor
0	0	0
0	1	1
1	0	1
1	1	0

Passos para a Criação de uma RN

- Definir os padrões
- Inicializar a rede
- Definir os parâmetros de treinamento
- Treinar a rede
- Testar a rede

Definindo os Padrões

X_1	X_2	valor
0	0	0
0	1	1
1	0	1
1	1	0

Vetor de entrada: $P = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \rightarrow P = [0 \quad 0 \quad 1 \quad 1; \quad 0 \quad 1 \quad 0 \quad 1]$

Vetor de saída: $T = [0 \quad 1 \quad 1 \quad 0]$

Passos para a Criação de uma RN

- Definir os padrões
- Inicializar a rede
- Definir os parâmetros de treinamento
- Treinar a rede
- Testar a rede

Inicializando a Rede Neural

Redes Feed-forward: **Função "newff"**

```
net = newff(   [min(P')' max(P')'],
               [N_hidden 1],
               {'tansig' 'logsig'},
               'traingd');
```

(limites dos padrões de entrada)
(número de neurônios de cada camada)
(função de ativação de cada camada)
(algoritmo de treinamento)

Funções de Ativação

purelin	Linear
logsig	Sigmóide
tansig	Tangente hiperbólica
satlin(s)	Linear com saturação

Algoritmos de Treinamento

traingd	*Gradient descent backpropagation*
traingdm	*Gradient descent backpropagation* com momentum
traingda	*Gradient descent backpropagation* com taxa adaptativa
traingdx	*Gradient descent backpropagation* com momentum e taxa adaptativa
trainlm	*Levenberg-Marquardt backpropagation* (default)
trainrp	*Resilient backpropagation* (Rprop)

Passos para a Criação de uma RN

- Definir os padrões
- Inicializar a rede
- **Definir os parâmetros de treinamento**
- Treinar a rede
- Testar a rede

Definindo parâmetros de treinamento

```
net.trainParam.epochs = 100;          Número de epochs
net.trainParam.goal = 1e-8;           Erro final desejado
net.trainParam.lr = 0.01;             Taxa de aprendizado
net.trainParam.show = 25;             Atualização da tela (epochs)

net.trainParam.mc = 0.9;              Taxa de momentum

net.trainParam.lr_inc = 1.05;         Taxa de incremento da l.r.
net.trainParam.lr_dec = 0.7;          Taxa de decremento da l.r.
net.trainParam.max_perf_inc = 1.04;   Incremento máximo do erro
```

Passos para a Criação de uma RN

- Definir os padrões
- Inicializar a rede
- Definir os parâmetros de treinamento
- Treinar a rede
- Testar a rede

Treinando a Rede Neural

```
net = train(net, P, T);
```

Passos para a Criação de uma RN

- Definir os padrões
- Inicializar a rede
- Definir os parâmetros de treinamento
- Treinar a rede
- Testar a rede

Testando a Rede Neural

```
C = sim(net, P);
```

M-file desenvolvida para o XOR

xor1.m

Validação Cruzada

- Dividir os padrões disponíveis em três conjuntos:
  - treinamento (70%): matrizes Ptrain, Ttrain
  - validação (20%): matrizes Pvalid, Tvalid
  - teste (10%): matrizes Ptest, Ttest

Validação Cruzada

```
% Inicializa a rede neural
net = newff([min(P')' max(P')'],[10 1],{'tansig' 'logsig'},'traingd');
net.trainParam.goal = 1e-8;

% Treina a rede iterativamente, de 5 em 5 epochs,
% até o total de 100 epochs, calculando os erros
Nepoch   = 5;
NN       = 20;
mape_min = 1e38;

for i = 1:NN,
  net.trainParam.epochs = Nepoch;
  net = train(net, Ptrain, Ttrain);

  Ctrain = sim(net, Ptrain);
  Cvalid = sim(net, Pvalid);

  % Calcula os erros MAPE para os padrões de treinamento e validação
  mape_train(i) = 100*mean(abs((Ttrain-Ctrain)./Ttrain))
  mape_valid(i) = 100*mean(abs((Tvalid-Cvalid)./Tvalid))
```

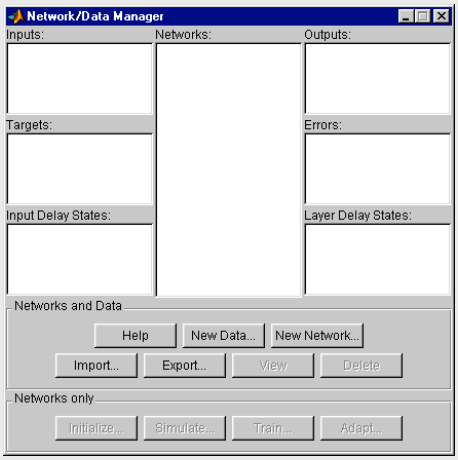
Validação Cruzada

```
  % encontra o número de epochs ótimo
  if (mape_valid(i) < mape_min)
    mape_min = mape_valid(i);
    net_opt = net;
    Noptim = Nepoch * i;
  end
end

% Melhor rede:
net = net_opt;

% Testa a rede com os 3 conjuntos de padrões
Ctrain = sim(net, Ptrain);
Cvalid = sim(net, Pvalid);
Ctest  = sim(net, Ptest);
```

Redes Neurais no MATLAB

- Duas formas de utilização:
  - Linhas de comando, e m-files
  - Interface gráfica (NNTool)

Interface Gráfica NNTool



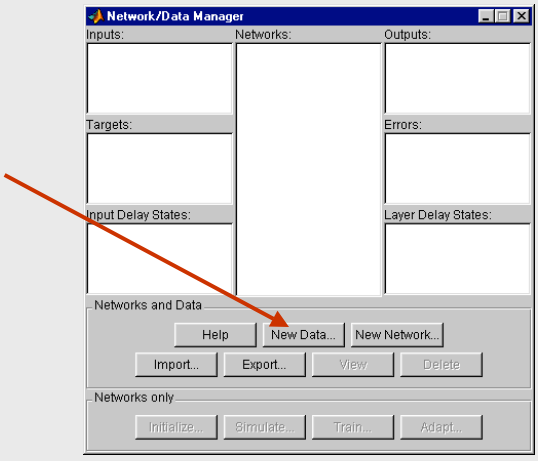
Passos para a Criação de uma RN

- Definir os padrões
- Criar a rede
- Inicializar a rede
- Definir os parâmetros de treinamento
- Treinar a rede
- Testar a rede

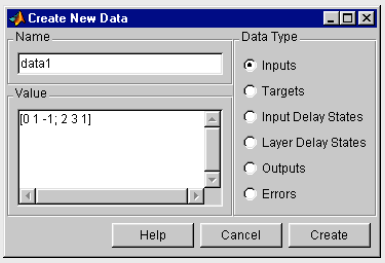
Passos para a Criação de uma RN

- Definir os padrões
- Criar a rede
- Inicializar a rede
- Definir os parâmetros de treinamento
- Treinar a rede
- Testar a rede

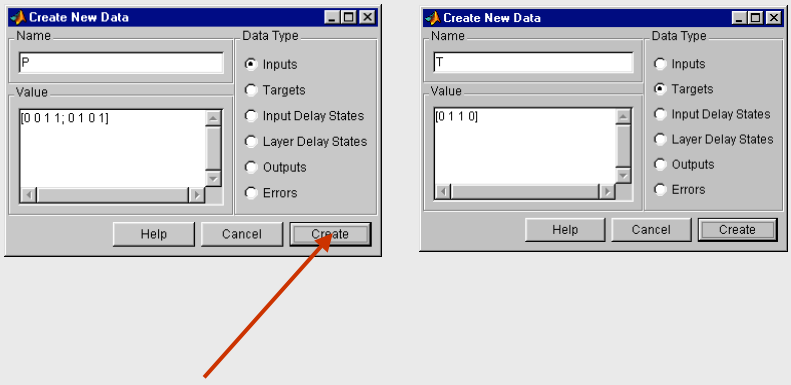
Definindo os Padrões



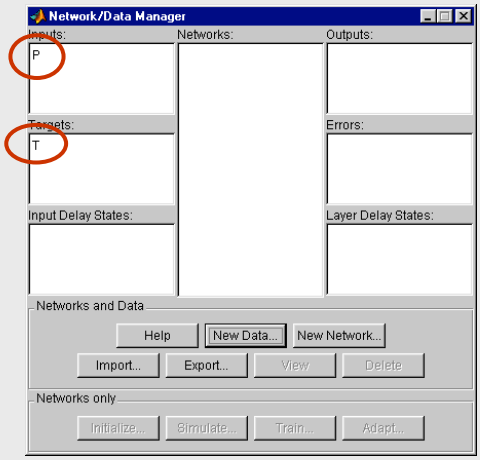
Definindo os Padrões



Definindo os Padrões



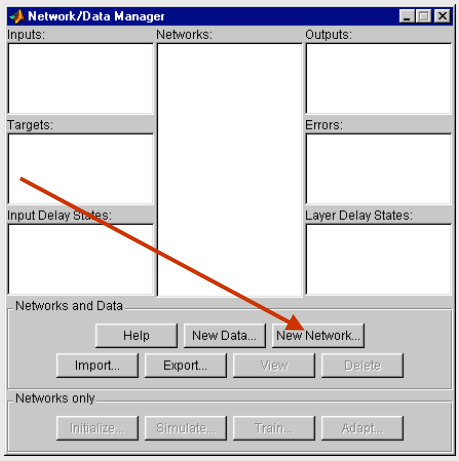
Definindo os Padrões



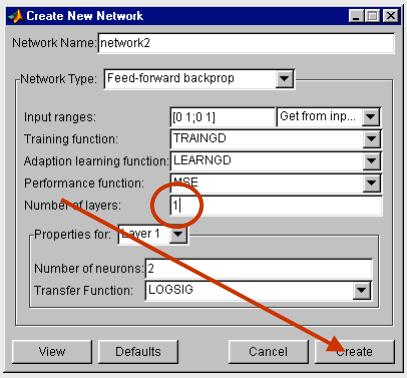
Passos para a Criação de uma RN

- Definir os padrões
- Criar a rede
- Inicializar a rede
- Definir os parâmetros de treinamento
- Treinar a rede
- Testar a rede

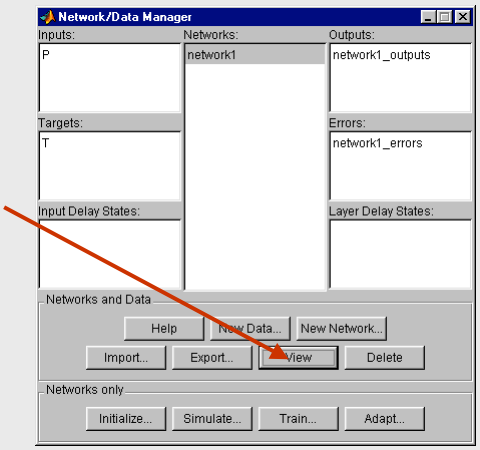
Criando a Rede Neural



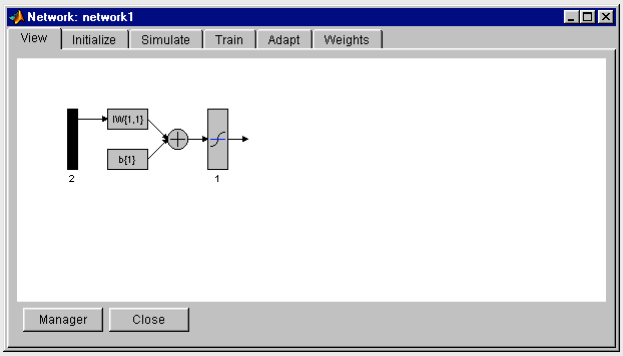
Criando a Rede Neural



Visualizando a Rede Neural



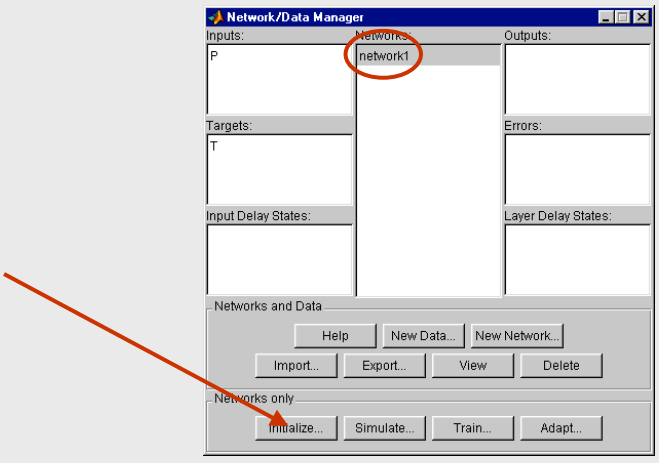
Visualizando a Rede Neural



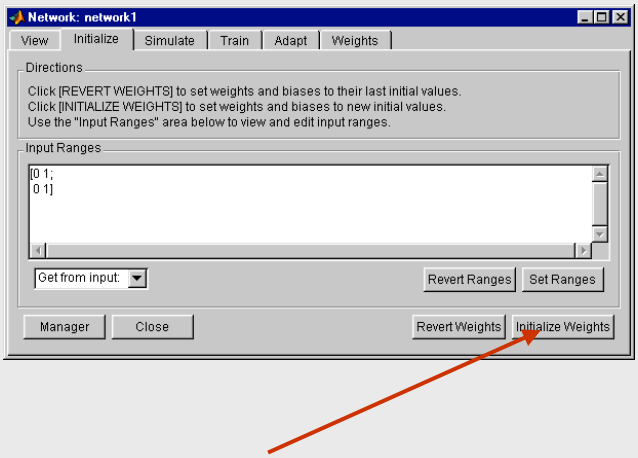
Passos para a Criação de uma RN

- Definir os padrões
- Criar a rede
- **Inicializar a rede**
- Definir os parâmetros de treinamento
- Treinar a rede
- Testar a rede

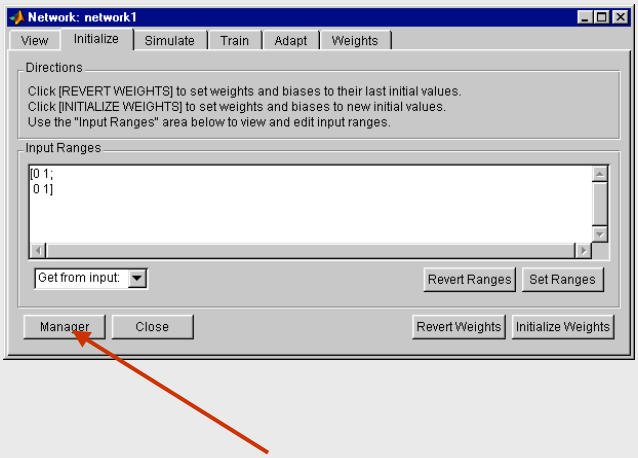
Inicializando a Rede Neural



Inicializando a Rede Neural



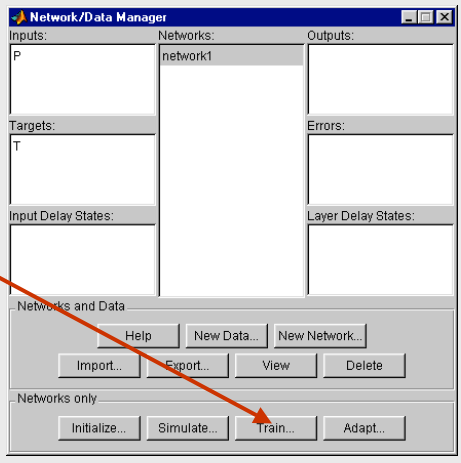
Inicializando a Rede Neural



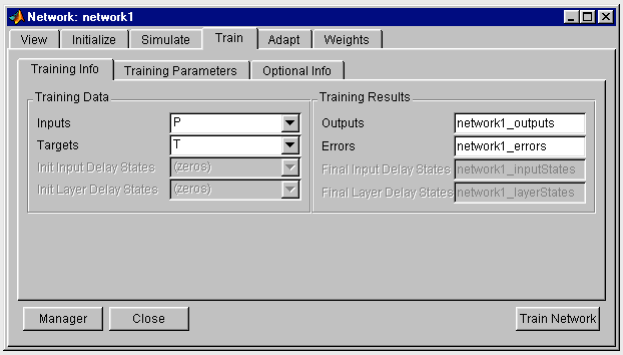
Passos para a Criação de uma RN

- Definir os padrões
- Criar a rede
- Inicializar a rede
- **Definir os parâmetros de treinamento**
- Treinar a rede
- Testar a rede

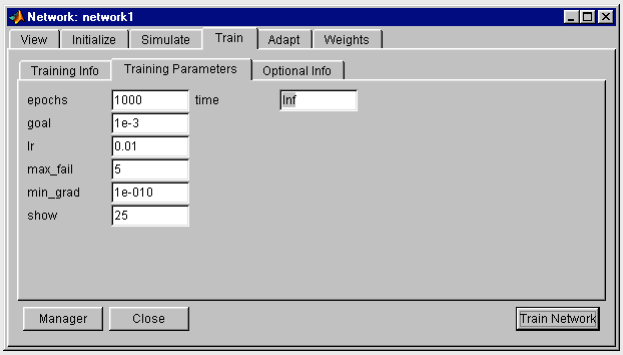
Definindo parâmetros de treinamento



Definindo parâmetros de treinamento



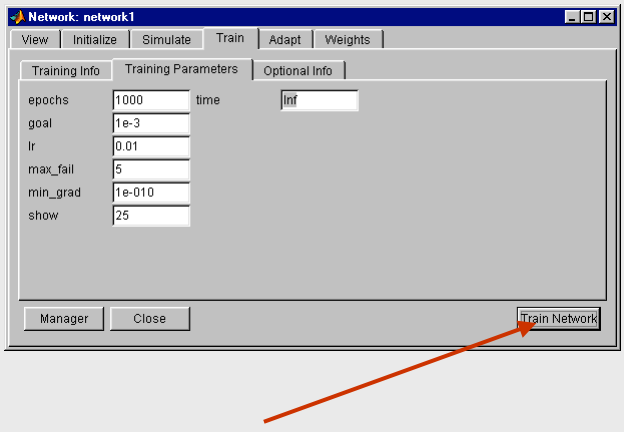
Definindo parâmetros de treinamento



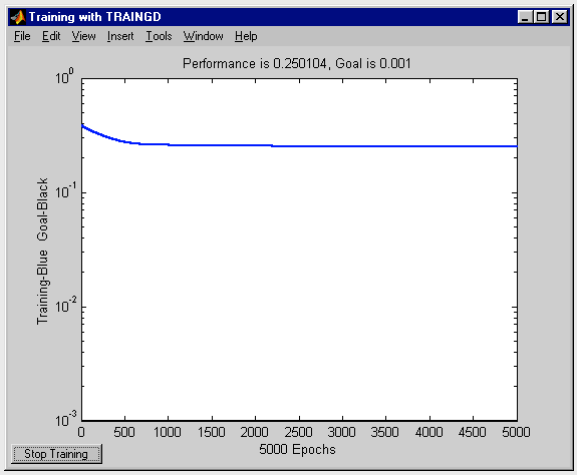
Passos para a Criação de uma RN

- Definir os padrões
- Criar a rede
- Inicializar a rede
- Definir os parâmetros de treinamento
- **Treinar a rede**
- Testar a rede

Treinando a Rede Neural



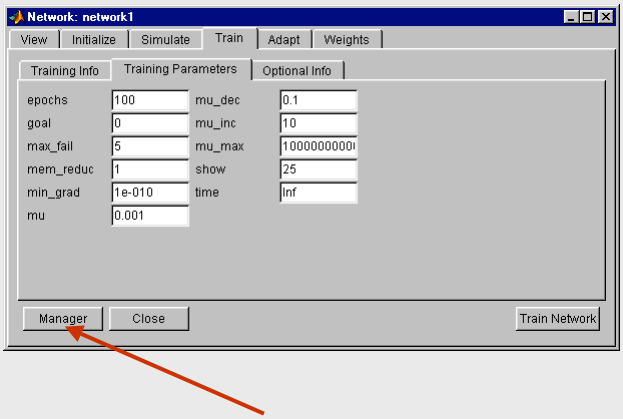
Treinando a Rede Neural



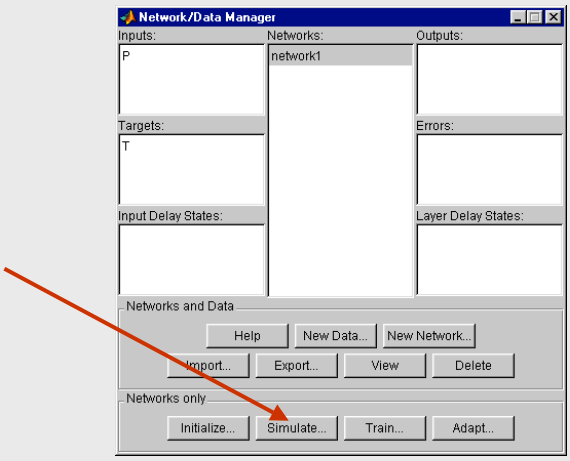
Passos para a Criação de uma RN

- Definir os padrões
- Criar a rede
- Inicializar a rede
- Definir os parâmetros de treinamento
- Treinar a rede
- Testar a rede

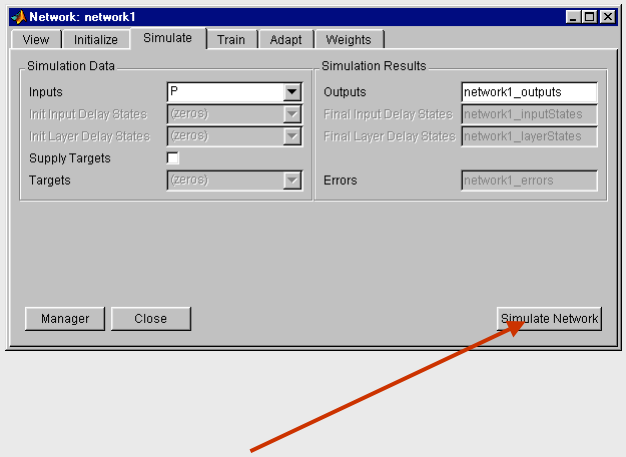
Testando a Rede Neural



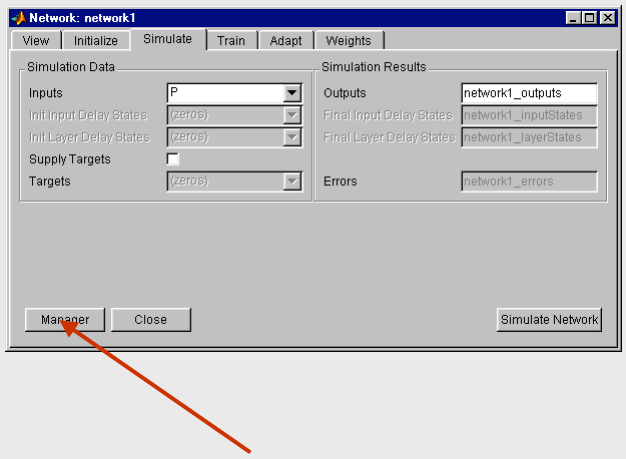
Testando a Rede Neural



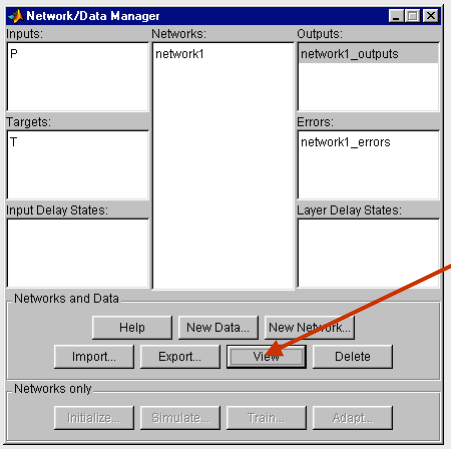
Testando a Rede Neural



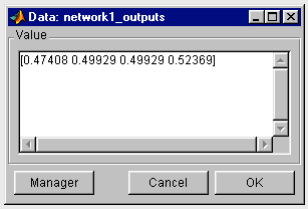
Testando a Rede Neural



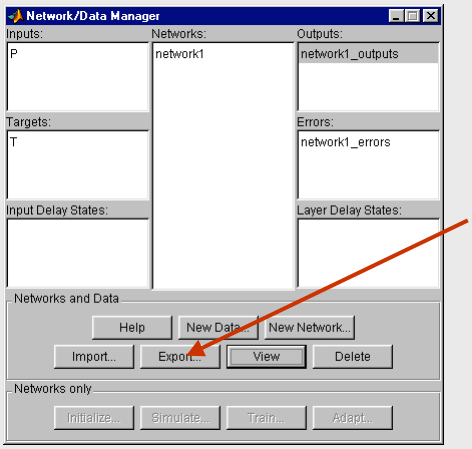
Testando a Rede Neural



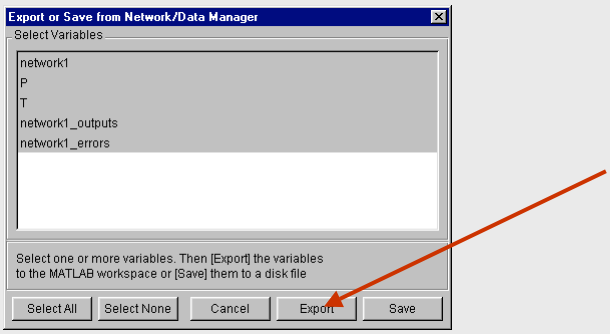
Testando a Rede Neural



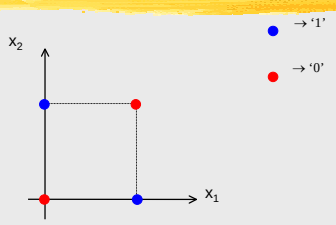
Exportando os Dados



Exportando os Dados

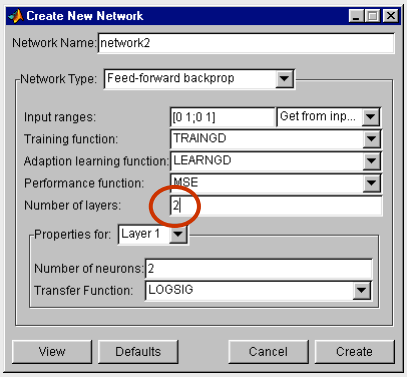


O Problema do OU Exclusivo

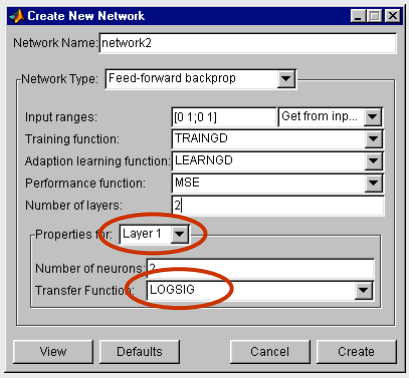


X_1	X_2	valor
0	0	0
0	1	1
1	0	1
1	1	0

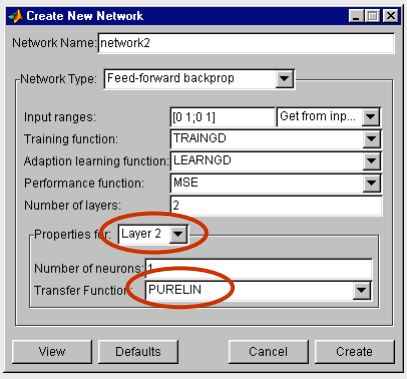
Rede Neural com Camada Escondida



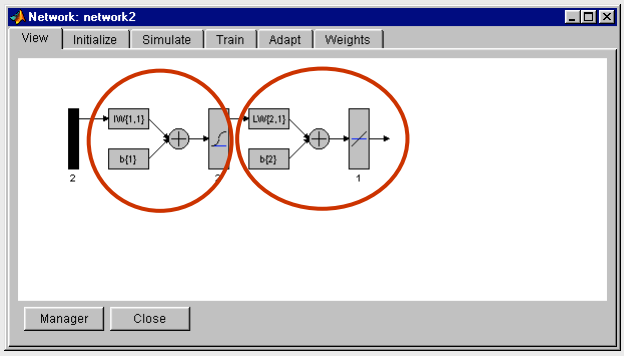
Rede Neural com Camada Escondida



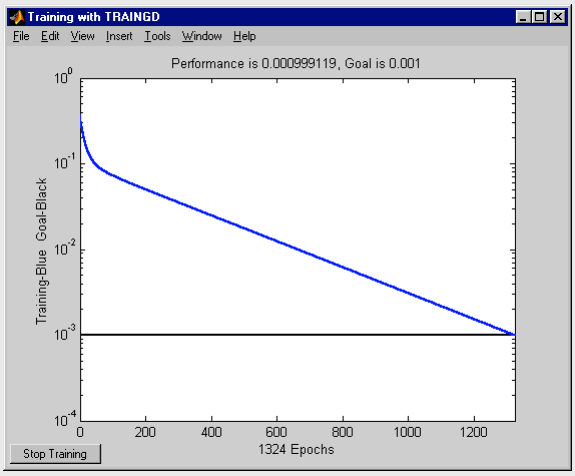
Rede Neural com Camada Escondida



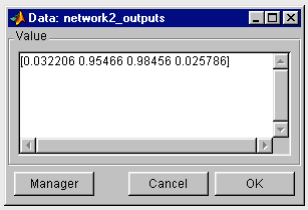
Rede Neural com Camada Escondida



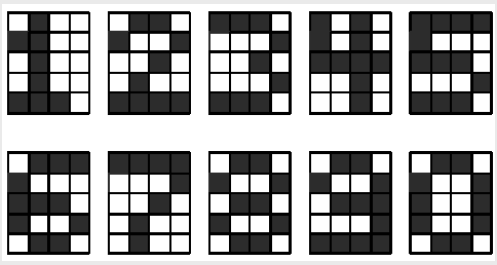
Rede Neural com Camada Escondida



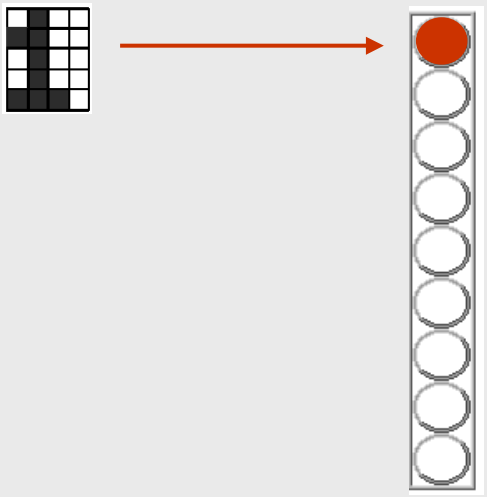
Rede Neural com Camada Escondida



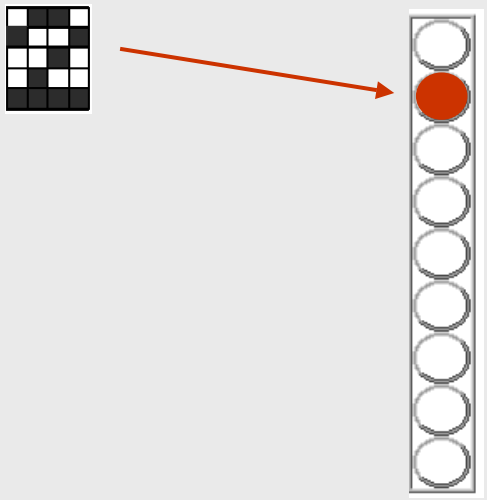
Reconhecimento de Dígitos



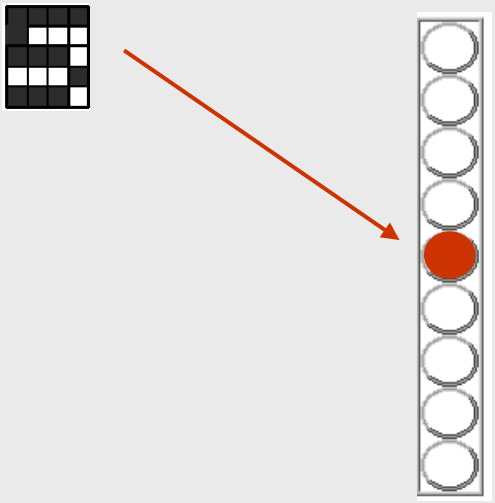
Reconhecimento de Dígitos



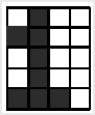
Reconhecimento de Dígitos



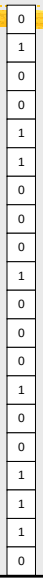
Reconhecimento de Dígitos



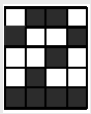
Definição dos Padrões de Entrada



0	1	0	0
1	1	0	0
0	1	0	0
0	1	0	0
1	1	1	0



Definição dos Padrões de Entrada



0	1	1	0
1	0	0	1
0	0	1	0
0	1	0	0
1	1	1	1

0
1
1
0
1
0
0
1
0
0
1
0
0
1
0
0
1
1
1
1

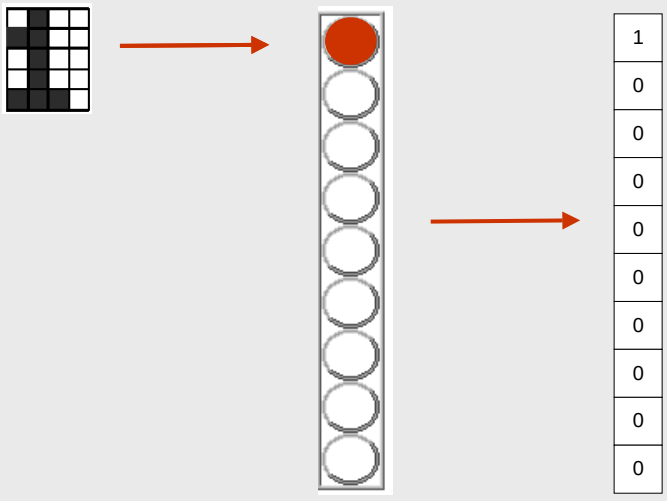
Definição dos Padrões de Entrada

- Cada dígito (padrão): **20 bits**
- Número de padrões: **10 dígitos**
- Representação: **Matriz 20 x 10**
  - Cada coluna representa um dígito
  - Cada linha representa um bit
  - Cada bit está associado com um neurônio de entrada

Definição dos Padrões de Entrada

digitos.txt

Definição dos Padrões de Saída

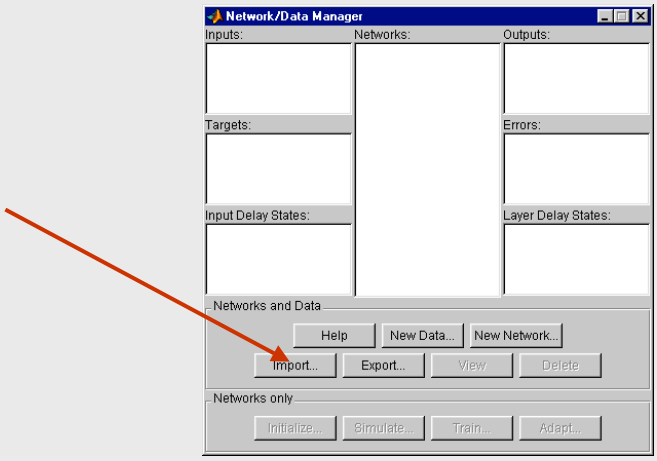


Definição dos Padrões de Saída

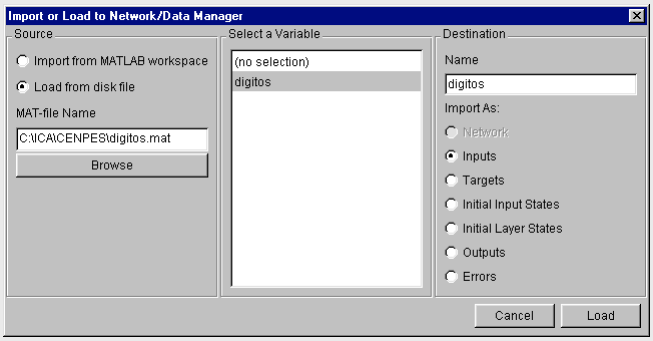
Conversão dos Arquivos

```
>> load digitos.txt
>> save digitos
>> load saidas.txt
>> save saidas
```

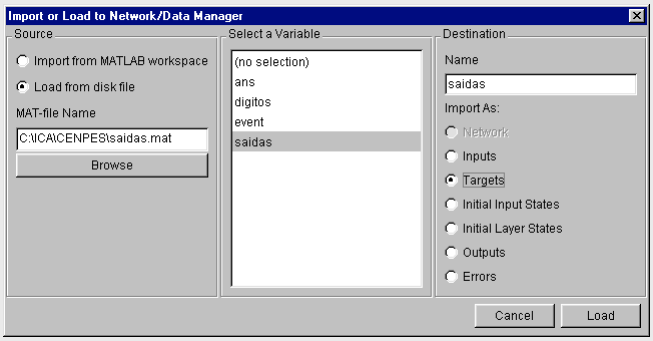
Importação dos Dados p/ NNTool



Importação dos Dados p/ NNTool



Importação dos Dados p/ NNTool



Definição das Redes

- Network name: network15
- Tipo: feed-forward backprop
- Input Ranges: Get from input
- Training Function: TRAINGDM
- Number of layers: 2
  - Layer 1: 15 neurons TANSIG
  - Layer 2: 10 neurons PURELIN

Definição das Redes

- Network name: network25
- Tipo: feed-forward backprop
- Input Ranges: Get from input
- Training Function: TRAINGDM
- Number of layers: 2
  - Layer 1: 25 neurons TANSIG
  - Layer 2: 10 neurons PURELIN

Definição das Redes

- Network name: network35
- Tipo: feed-forward backprop
- Input Ranges: Get from input
- Training Function: TRAINGDM
- Number of layers: 2
  - Layer 1: 35 neurons TANSIG
  - Layer 2: 10 neurons PURELIN

Treinamento das Redes

- Epochs: 10000
- Goal (MSE): 0.5e-3
- Learning Rate (lr): 0.1
- Momentum: 0.0

Treinamento das Redes

- Epochs: 10000
- Goal (MSE): 0.5e-3
- Learning Rate (lr): 0.4
- Momentum: 0.0

Treinamento das Redes

- Epochs: 10000
- Goal (MSE): 0.5e-3
- Learning Rate (lr): 0.9
- Momentum: 0.0

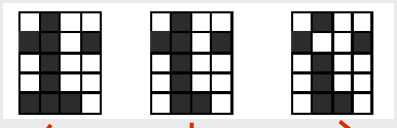
Treinamento das Redes

- Epochs: 10000
- Goal (MSE): 0.5e-3
- Learning Rate (lr): 0.1
- Momentum: 0.4

Treinamento das Redes

- Epochs: 10000
- Goal (MSE): 0.5e-3
- Learning Rate (lr): 0.9
- Momentum: 0.4

Teste das Redes



1 bit errado 2 bits errados 3 bits errados