# Neural Network And MATLAB

*Navid Mahmoudian*

*Masood Alizadeh Arjmand*

*Ablfazl Mohammadi*

# Neural Network

- **عناوين مباحث:**
  - **شبکه Perceptron**
    - ساختن ، مقدار دهي اوليه و آموزش شبکه در GUI
    - ساختن ، مقدار دهي اوليه و آموزش شبکه با استفاده از توابع
  - **شبکه Feed-Forward**
    - ساختن ، مقدار دهي اوليه و آموزش شبکه در GUI
    - ساختن ، مقدار دهي اوليه و آموزش شبکه با استفاده از توابع
  - **شبکه RBF**
    - ساختن ، مقدار دهي اوليه و آموزش شبکه در GUI
    - ساختن ، مقدار دهي اوليه و آموزش شبکه با استفاده از توابع
  - **شبيه سازي شبکه**

# Neural Network

Perceptron GUI

# Perceptron Neural Network

- **معرفي nntool:**
- پنجره مربوطه براي كار با GUI ،پنجره Network/Data Manager نام دارد.
- اين پنجره به صورت مستقل از command-line workspace كار مي كند.
- مي توانيم نتايج قسمت GUI را به workspace منتقل كنيم يا نتايج workspace را در داخل GUI قرار دهيم .
- قسمت GUI را با يک مثال كه پياده سازي تابع AND است ادامه مي دهيم.
- p بردار ورودي است.
- t بردار خروجي (هدف) است.

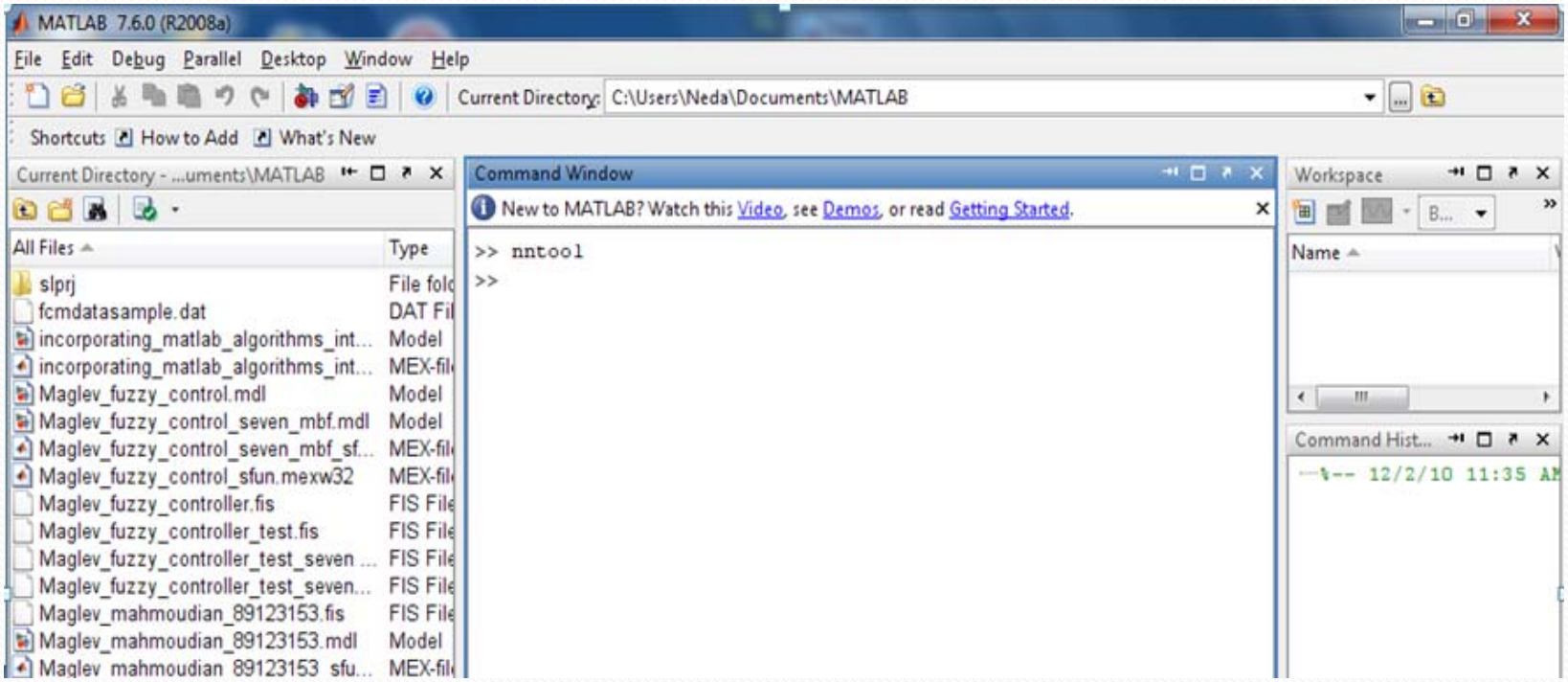
| | j = 1 | J = 2 | J = 3 | J = 4 |
|---|---|---|---|---|
| $P_{1j}$ | 0 | 0 | 1 | 1 |
| $P_{2j}$ | 0 | 1 | 0 | 1 |
| $t_{1j}$ | 0 | 0 | 0 | 1 |

$$p = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

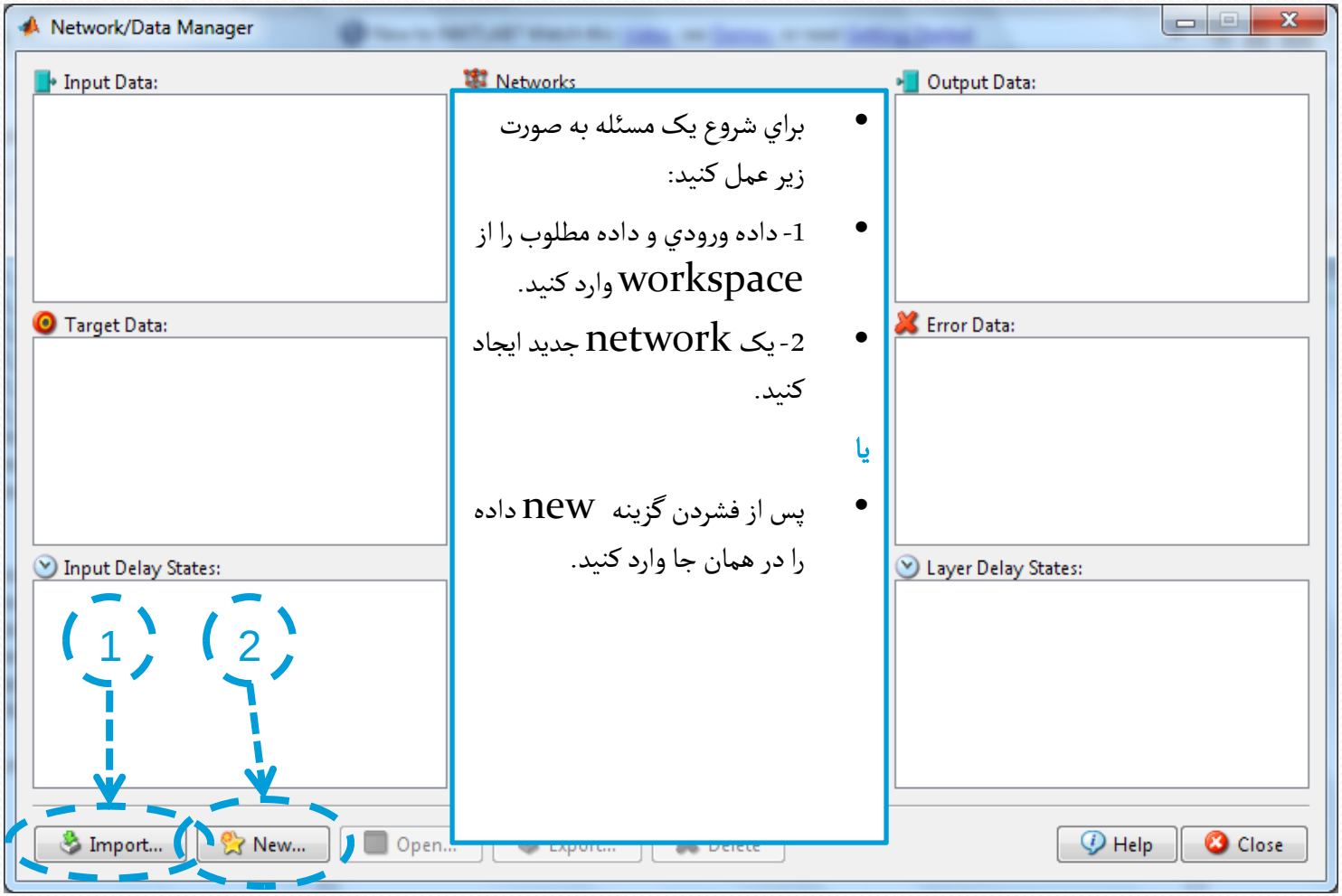
$$t = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$

# Perceptron Neural Network

- **ورودي و هدف:**
  - در command window با تايپ nntool پنجره Network/Data Manager را باز كنيد.



# Perceptron Neural Network

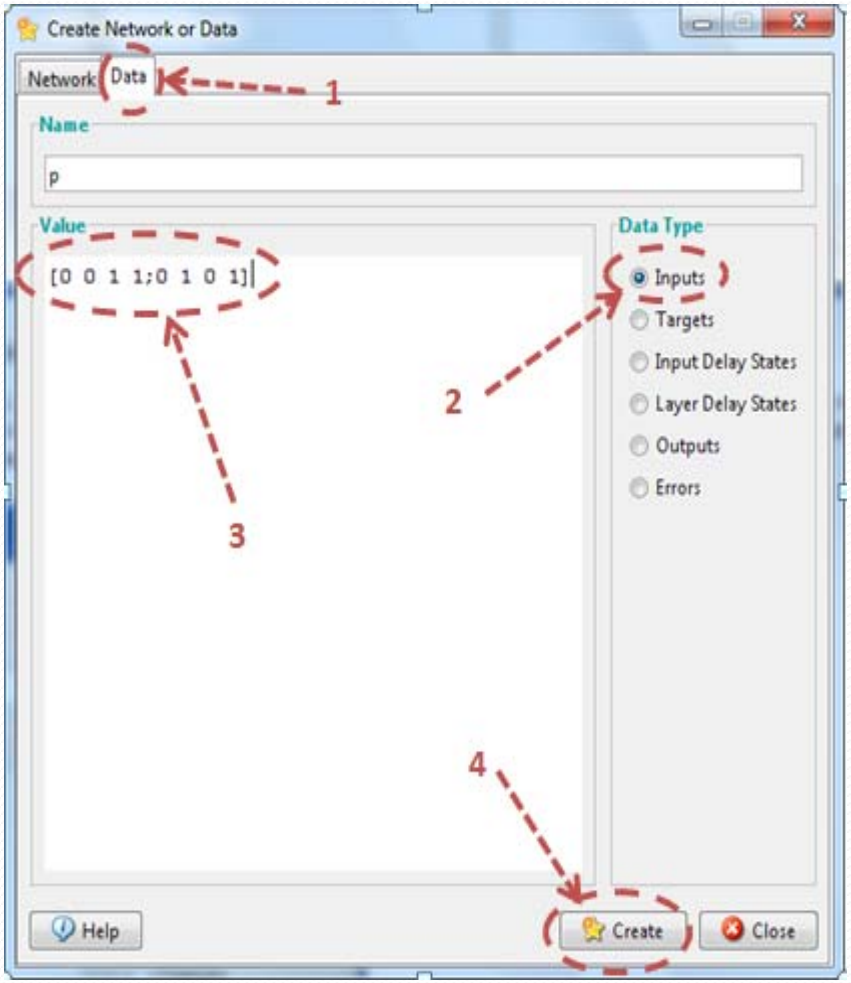



- براي شروع يک مسئله به صورت زير عمل کنيد:
- 1- داده ورودي و داده مطلوب را از workspace وارد کنيد.
- 2- يک network جديد ايجاد کنيد.

يا

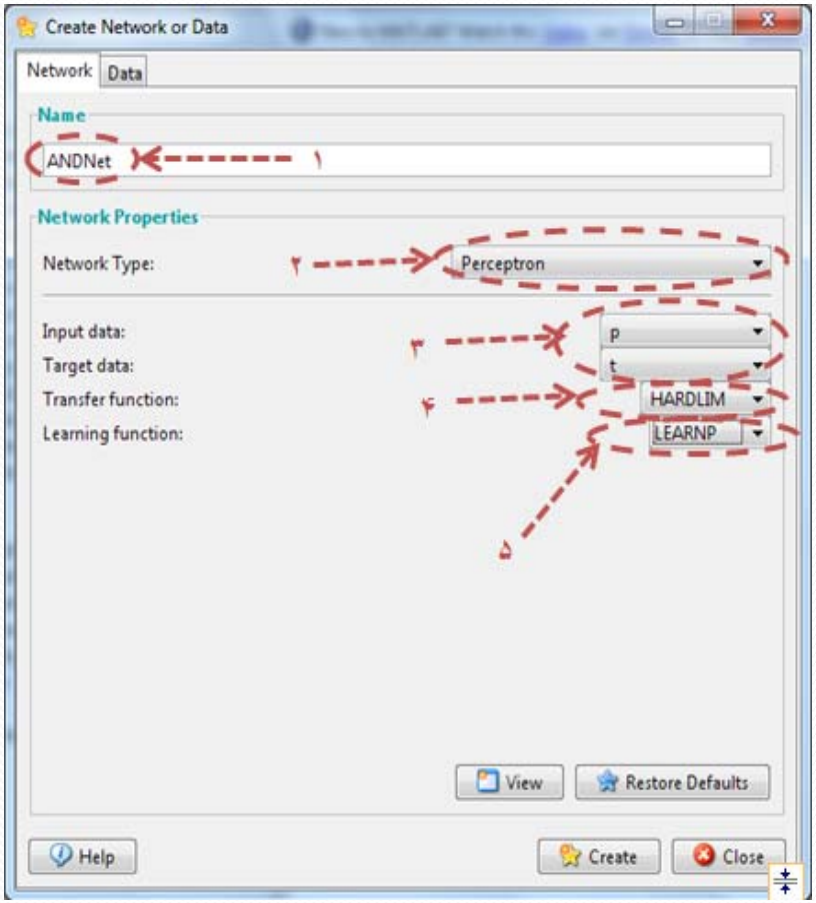
- پس از فشردن گزينه new داده را در همان جا وارد کنيد.

# Perceptron Neural Network




- **وارد كردن ورودي از گزينه new**
  - در پنجره Network/Data Manager
    گزينه new را بزنيد.
  - پنجره Create Network or Data ظاهر مي شود.
  - 1- data tab را انتخاب كنيد .
  - 2- inputs/targets را انتخاب كنيد.
  - 3- داده ورودي/خروجي را در قسمت
    value وارد كنيد .
  - 4- گزينه Create را فشار دهيد.
  - اين نتايج در پنجره
    Network/Data Manager ظاهر مي شود.

# Perceptron Neural Network

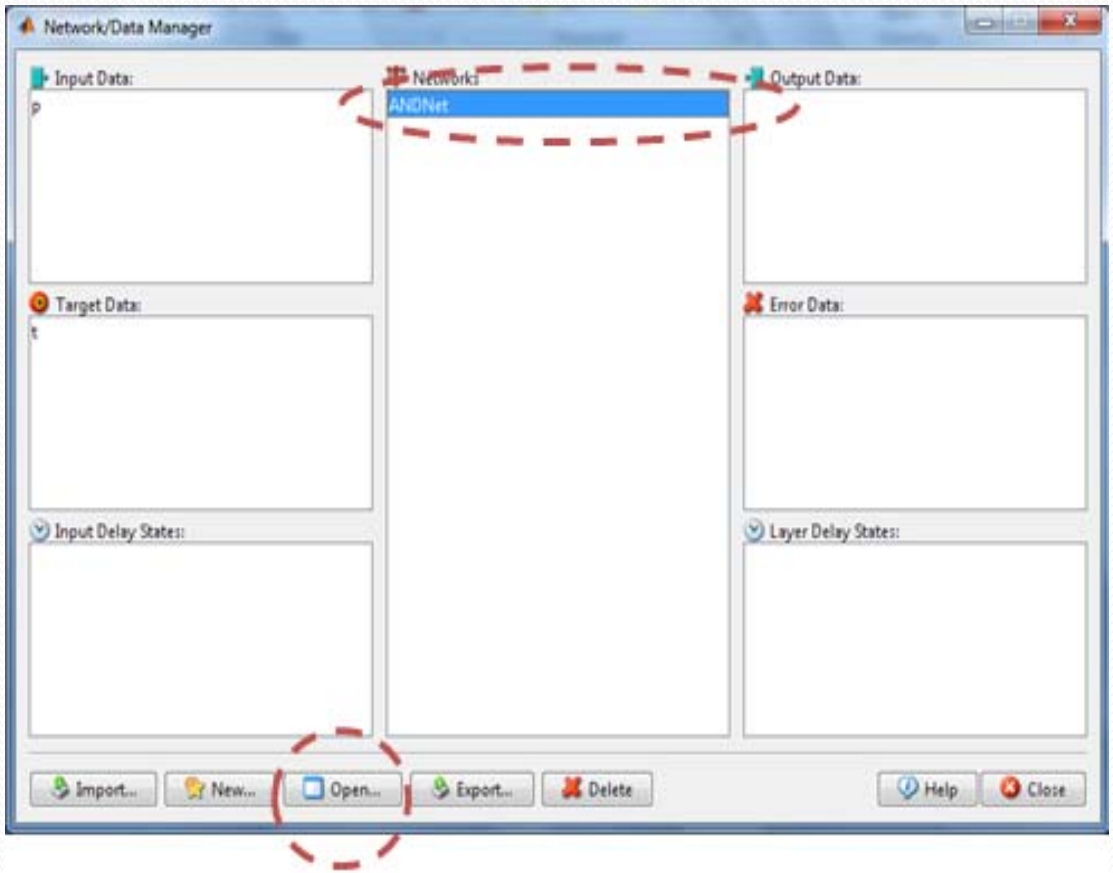



- **ساختن network**
  - در همان پنجره Create Network or Data ، Network tab را انتخاب کنید.
  - 1- اسم شبکه را ANDNet انتخاب کنید.
  - 2- روي Network Type رفته و گزینه perceptron را انتخاب کنید.
  - 3- در قسمت input data و target data نامي را كه براي ورودي و خروجيِ مطلوب خود انتخاب کردید قرار دهید.
  - 4- transfer function (activation function) خود را از بین دو گزینه موجود انتخاب کنید:

$$\begin{cases} unipolar \rightarrow \textbf{hardlim} \\ bipolar \rightarrow \textbf{hardlims} \end{cases}$$

  - 5- learning function خود را انتخاب کنید.
  - 6- create را فشار دهید.

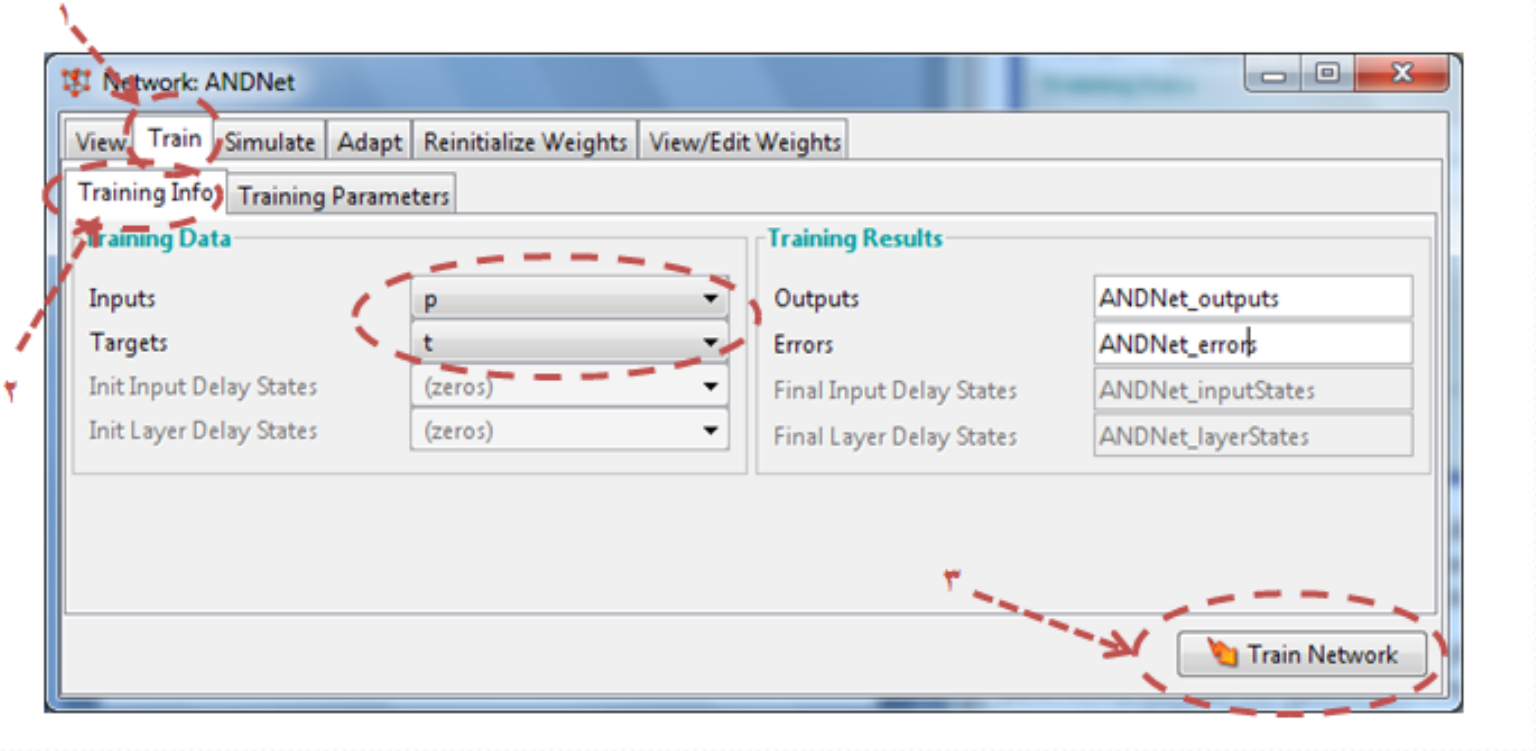
# Perceptron Neural Network



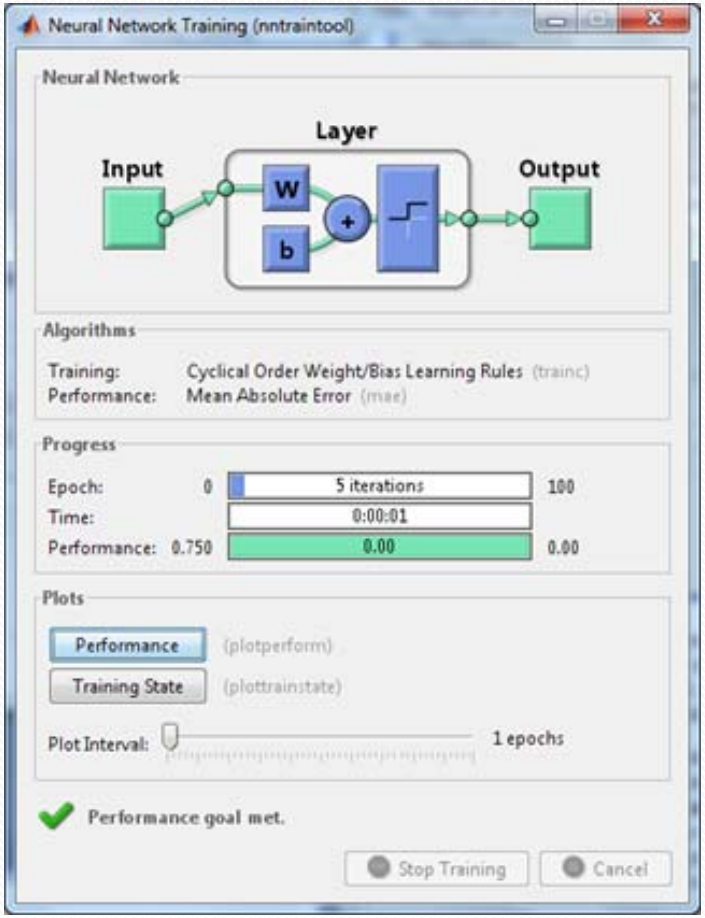

- **آموزش network**
- براي train كردن شبكه ، در پنجره Network/Data Manager بر روي اسم شبكه (ANDNet) كليک كنيد و سپس open را بزنيد.
- با اين كار پنجره جديدي به نام Network: ANDNet باز مي شود.

# Perceptron Neural Network

- 1- وارد Train tab شوید.
- 2- در قسمت Training Info ورودي ها و خروجي ها را تعيين كنيد.
- 3- Train Network را فشار دهید تا شبکه train شود.



# Perceptron Neural Network




- با فشار دادن train network،

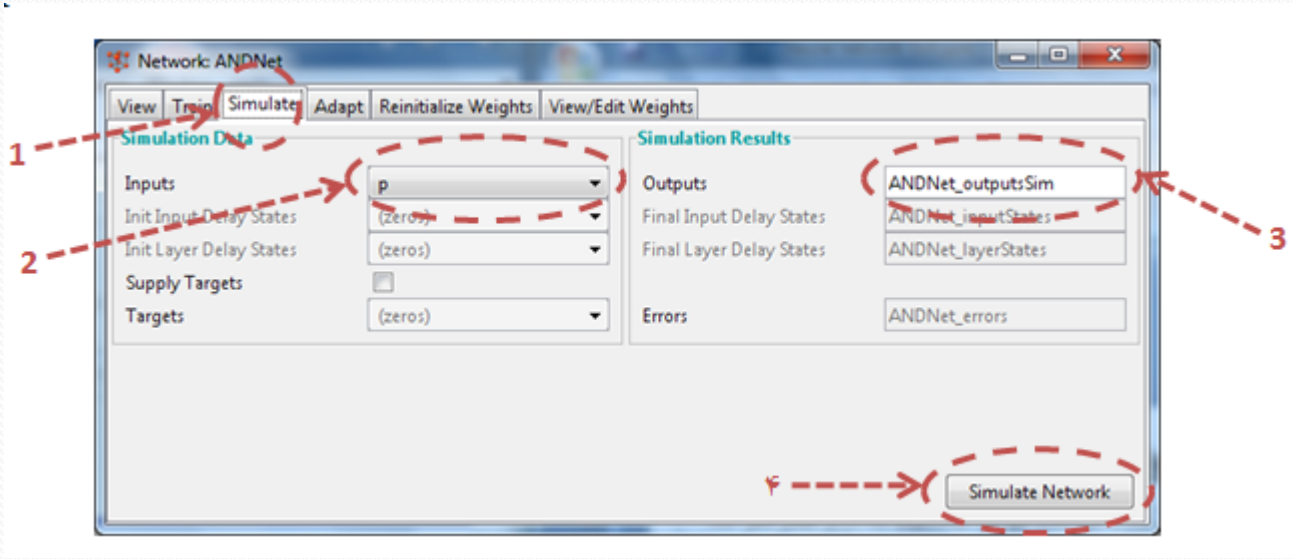
پنجره Neural Network Training باز مي شود.

- همانطور كه مي بينيد ، براي اين مثال خاص

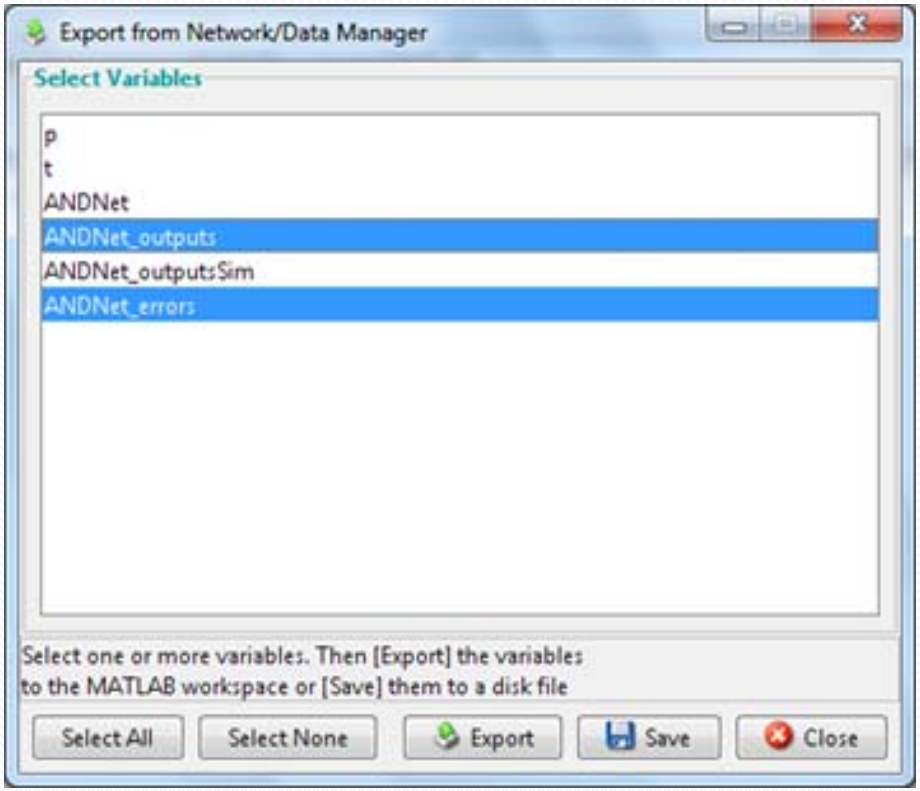
با 5 مرحله تكرار به خطاي صفر مي رسيم.

# Perceptron Neural Network

- **شبيه سازي**
- مي توان با شبيه سازي تأييد كرد شبكه train شده داراي خطاي صفر است.
- 1- براي اينكار به پنجره Network: ANDNet رفته و وارد Simulate tab شويد.
- 2- ورودي p را تعيين كنيد.
- 3- خروجي شبيه سازي را ANDNet_outputsSim قرار دهيد تا با خروجي شبكه train شده تمايز قائل شويد.
- 4 - Simulate Network را فشار دهيد.
- همانگونه كه مشاهده مي كنيد ، در پنجره Network/Data Manager يک متغير خروجي جديد به نام ANDNet_outputsSim ظاهر مي شود.
- با دو بار كليک روي اين متغير پنجره Data: ANDNet_outputsSim ظاهر مي شود و نتيجه [1 0 0 0] را مشاهده مي كنيم.



# Perceptron Neural Network




- **استخراج نتایج به workspace**
  - در پنجره Network/Data Manager بر روي Export مي كنند.
  - به عنوان مثال ANDNet_outputs و ANDNet_errors را انتخاب كنيد.
  - بر روي Export كليک كنيد.
  - اين دو متغير به workspace منتقل مي شود.

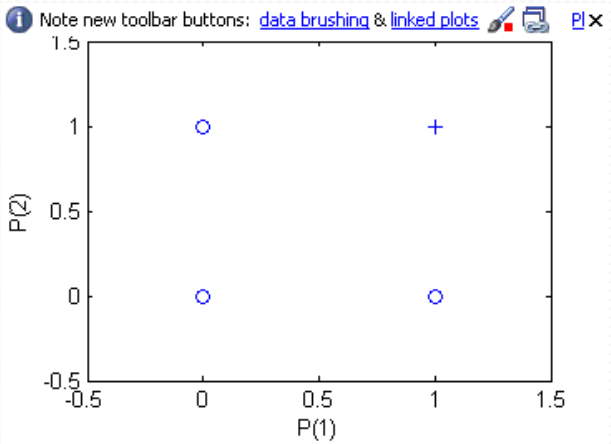
# Perceptron Neural Network

- **رسم ورودی و خروجی با تابع plotpv**
  - اگر ورودی و خروجی به صورت مقابل باشد:

$$p = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

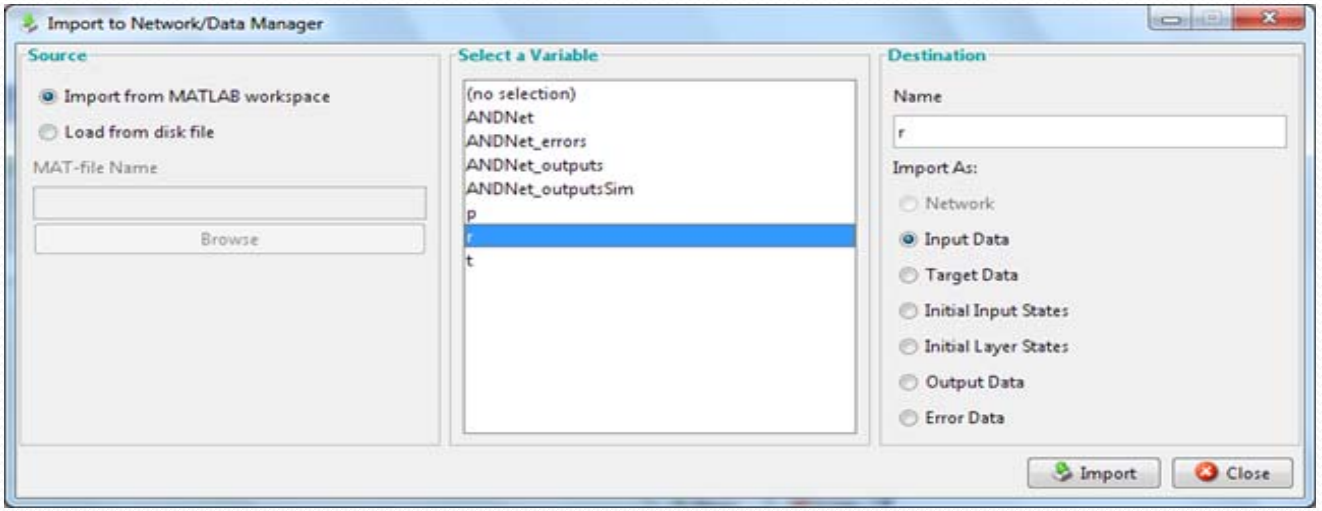
$$t = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$

- می توان برای طبقه بندی ترسیم زیر را نشان داد که با دستور plotpv(p,t) انجام می شود.



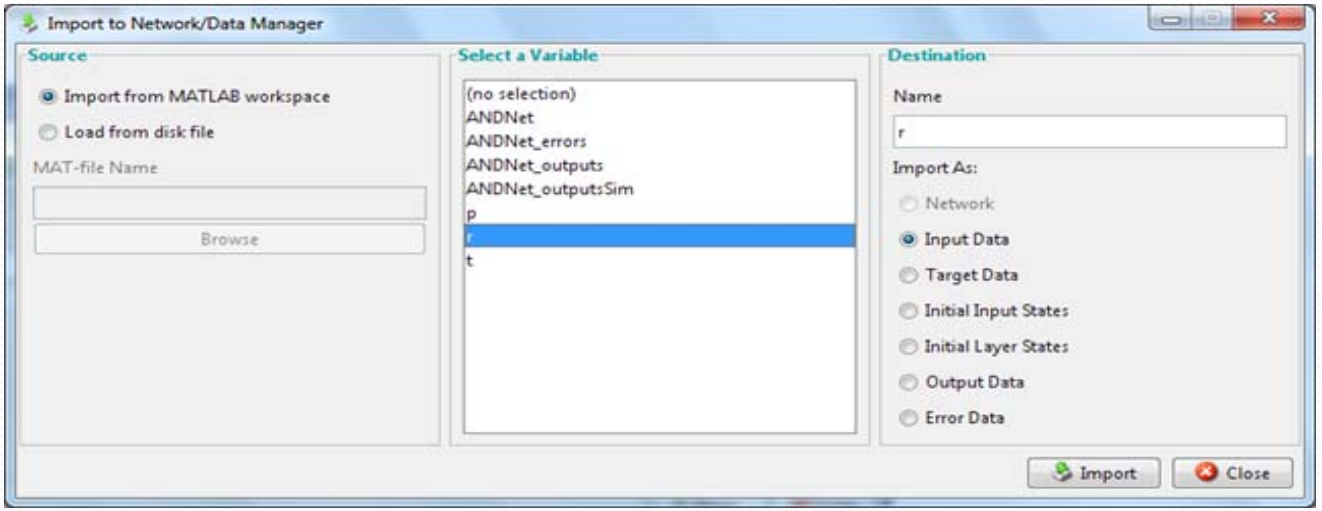
# Perceptron Neural Network

- **وارد کردن ورودي و خروجي از command line**
- در قسمت command line بردار r را به صورت r=[0;1;2;3] تعريف كنيد.
- در پنجره Network/Data Manager بر روي import كليک كنيد. تا پنجره Import to Network/Data Manager باز شود.
- r را انتخاب كنيد.
- گزينه Input Data را انتخاب كنيد.
- بر روي Import كليک كنيد.



# Perceptron Neural Network

- **وارد کردن ورودي و خروجي از command line**
- در قسمت command line بردار r را به صورت r=[0;1;2;3] تعريف كنيد.
- در پنجره Network/Data Manager بر روي import كليک كنيد. تا پنجره Import to Network/Data Manager باز شود.
- r را انتخاب كنيد.
- گزينه Input Data را انتخاب كنيد.
- بر روي Import كليک كنيد.



# Perceptron Neural Network

Feed-Forward GUI

# Perceptron Neural Network

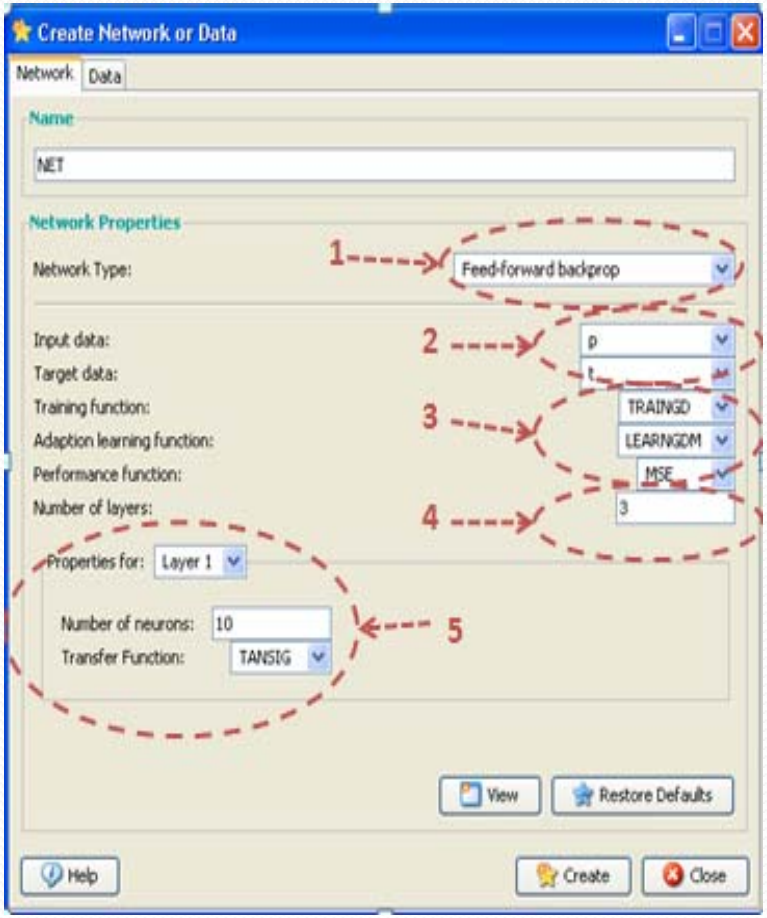
- یک network جدید تعریف مي کنیم که ورودي هاي آن به صورت زیر باشد.

$$p = \begin{bmatrix} 4 & 1 & 2 \\ 0 & 4 & 5 \\ 2 & 5 & 3 \end{bmatrix}$$

$$t = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

- براي اینکار از روش هاي گفته شده در قبل استفاده مي کنیم.

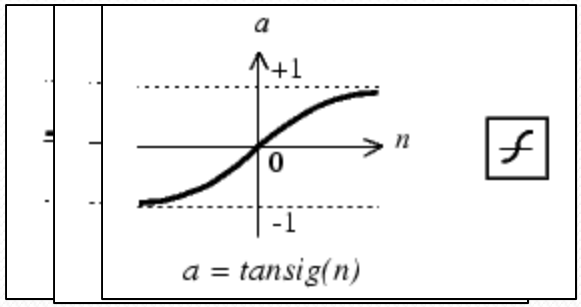
# Perceptron Neural Network




- در پنجره Create Network or Data
  گزينه هاي مطلوب را انتخاب مي كنيم
- براي اين مثال از 3 لايه استفاده
  مي شود.(2 لايه hidden و 1 لايه خروجي)
- 1- براي Network type گزينه
  feed-forward backprop را انتخاب مي كنيم.
- 2- ورودي و خروجي ها را مشخص مي كنيم.
- 3- Training function و
  Adaption learning function و
  performance function مورد نظر را انتخاب مي كنيم.
- 4- تعداد لايه ها را كه در اين مثال سه فرض مي كنيم.
- 5- تعداد neuron ها در هر لايه و
  transfer function مورد نظر را انتخاب مي كنيم.

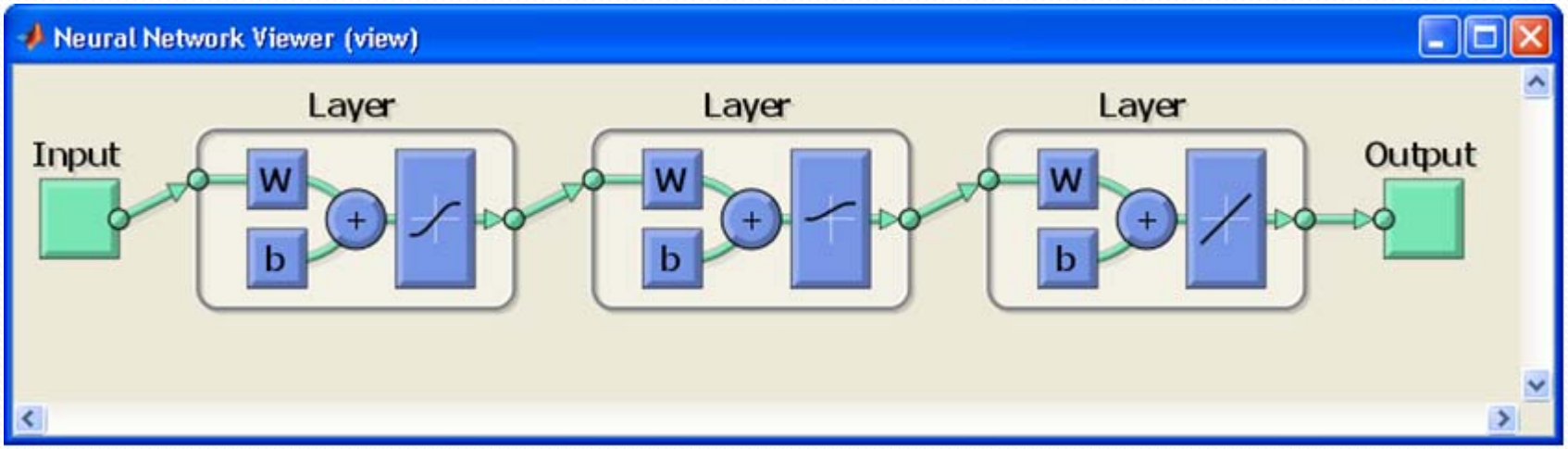
# Perceptron Neural Network

- Training function
  - TRAINGD ← روش آموزش شبکه با استفاده از گرادیان نزولي
  - TRAINGDM ← روش آموزش شبکه با استفاده از گرادیان نزولي و momentum
  - TRAINGDA ← روش آموزش شبکه با استفاده از گرادیان نزولي و Adaptive Learning Rate
  - و ...
- Adaption learning function
  - Gradient descent weight and bias learning function ← LEARNGD
  - Gradient descent with momentum weight and bias learning function ← LEARNGDM
- Performance function
  - MSE ← روش میانگین مربعات خطاي نرمال شده
  - SSE ← جمع مربعات خطا
  - Mean squared error with regularization performance function ← MSEREG
- Properties of layer
  - Transfer function :
    - LOGSIG ←
    - PURELIN ←
    - TANSIG ←



# Perceptron Neural Network

- در مثال گفته شده از مقادیر زیر استفاده کردیم:
  - Layer 1 ← 5 neurons ← TANSIG
  - Layer 2 ← 2 neurons ← LOGSIG
  - Layer 3 ← خود برنامه بر اساس خروجي ها انتخاب مي كند ← PURELIN
- شبکه ساخته شده به صورت زیر است :



# Perceptron Neural Network

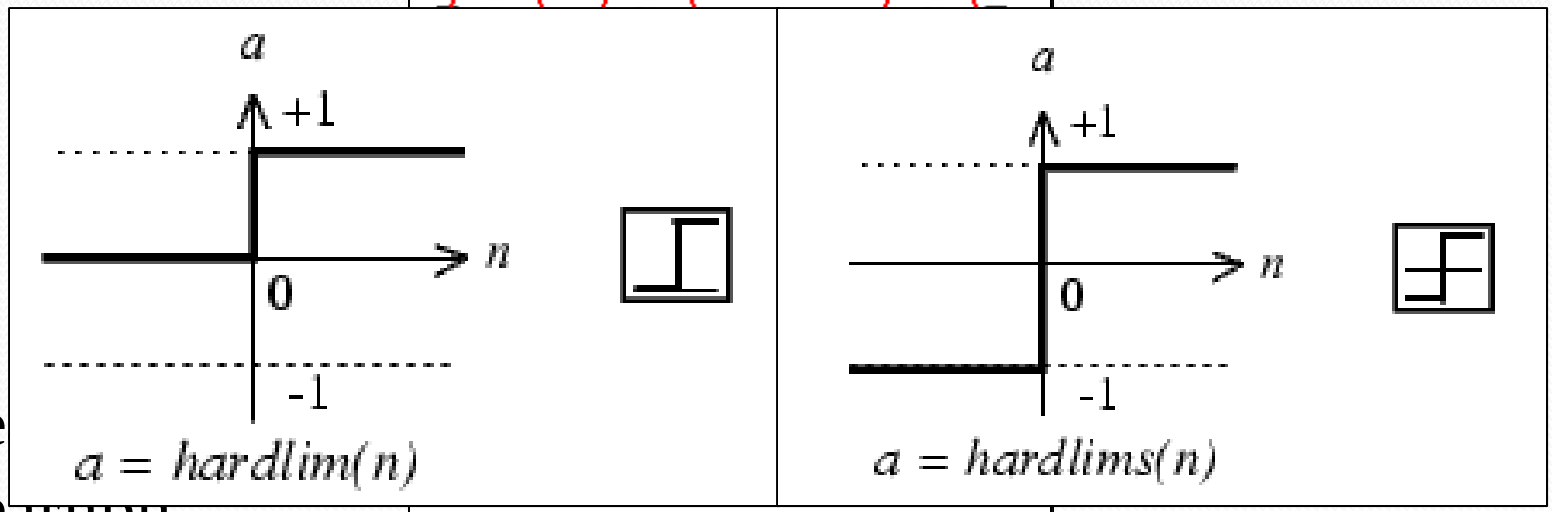
## Perceptron Functions

# Perceptron Neural Network

- ساختن شبکه Perceptron:

**net = newp (P,T,TF,LF)**

- P:
- T:
- TF:
- LF:
  - Le
  - Learnpn
- Example: **net = newp(P,T,'hardlim','learnp')**



# Perceptron Neural Network

- مقداردهي اوليه شبكه Perceptron:
  - تابع مقداردهي اوليه وزن ها:
    - initzero
    - rands
  - تابع مقداردهي اوليه بایاس ها:
    - randnc
    - initzero
    - randnr
    - midpoint
    - initsompc
    - initcon
    - revert
  - مقدار دهي اوليه شبكه:

| |
|---|
| **net.inputWeights{i,j}.initFcn** |
| **net. layerWeights{i,j}.initFcn** |
| **net.biases{i}.initFcn** |
| **net = init(net)** |

# Perceptron Neural Network

- آموزش شبکه Perceptron:
  - انتخاب حالت آمورش:
    - trains ⟸ incremental
    - trainc ⟸ batch
  - انتخاب تابع عملکرد:
    - mae
    - mse
    - msne
    - ...
  - تعیین تعداد تکرارها:
  - تعیین مقدار خطاي هدف:
  - آموزش شبکه:

**net.trainFcn**

**net.performFcn**

**net.trainParam.epochs**

**net.trainParam.goal**

**net = train(net, P, T)**

# Perceptron Neural Network

- آموزش شبکه Perceptron:

- Example:
  - **net.trainFcn = 'trains‘ ;**
  - **net.performFcn = ‘mse’ ;**
  - **net.trainParam.epochs = 50 ;**
  - **net.trainParam.goal = 0.01 ;**
  - **net = train(net, P, T) ;**

# Perceptron Neural Network

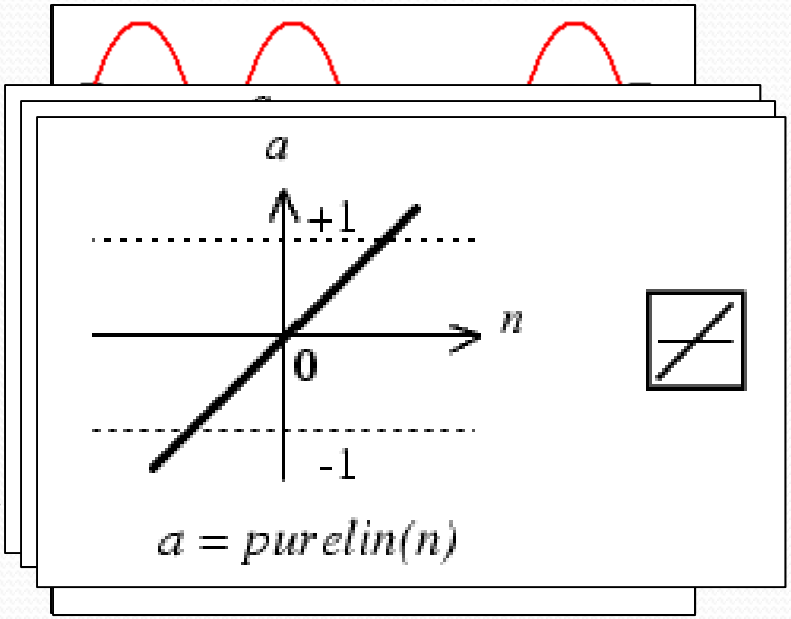
## Feed-Forward Functions

# Feed-Forward Neural Network

- ساختن شبکه feed-forward:

**net = newff(P,T,[S1 S2...S(N-l)],{TF1 TF2...TFNl}, BTF,BLF,PF,IPF,OPF,DDF)**

- P:
- T:
- [S1 S2 … S(N-1)]:
  - Example: [3 4]
- {TF1 TF2 … TFN}



# Feed-Forward Neural Network

- ساختن شبکه feed-forward:

**Net = newff(P,T,[S1 S2...S(N-l)],{TF1 TF2...TFNl}, BTF,BLF,PF,IPF,OPF,DDF)**

- BTF: Backpropagation network Training Function
  - ‘**trainlm**’ , ‘**trainbfg** ’ , ‘**trainbfgc**’ , …
- BLF: Backpropagation weight/bias Learning Function
  - ‘**learngdm**’ , ‘**learngd**’ , …
- PF: Performance Function
  - ‘**mae**’ , ‘**mse**’ , ‘ **msne**’ , …
- IPF: Row cell array of Input Processing Functions
  - **{'fixunknowns','removeconstantrows','mapminmax'}**
- OPF: Row cell array of Output Processing Functions
  - {'**removeconstantrows**','**mapminmax**'}
- DDF: Data Divison Function
  - ‘**dividerand**’

# Feed-Forward Neural Network

- مقداردهي اوليه شبکه feed-forward:
  - تابع مقداردهي اوليه وزن ها:
    - initzero
    - rands
  - تابع مقداردهي اوليه بایاس ها:
    - randnc
    - initzero
    - rands
    - initcon
    - revert
  - مقدار دهي اوليه شبکه:

| |
|---|
| **net.inputWeights{i,j}.initFcn** |
| **net.layerWeights{i,j}.initFcn** |
| **net.biases{i}.initFcn** |
| **net = init(net)** |

# Feed-Forward Neural Network

- آموزش شبکه feed-forward:
  - حالت traingd:

| net.trainParam.lr |
| --- |
| net.trainParam.show |
| net.trainParam.epochs |
| net.trainParam.goal |
| net.trainParam.time |
| net.trainParam.min_grad |



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

# Feed-Forward Neural Network

- آموزش شبکه feed-forward:
  - حالت traingdm:

| |
|---|
| **net.trainParam.lr** |
| **net.trainParam.show** |
| **net.trainParam.epochs** |
| **net.trainParam.goal** |
| **net.trainParam.time** |
| **net.trainParam.min_grad** |
| **net.trainParam.mc** |



| |
|---|
| **net = train(net, P, T)** |

# Neural Network

- شبيه سازي شبكه:

**a = sim(net,p)**

# Neural Network

RBF

# توابع کار با RBF

رابط گرافیکی شبکه RBF

توابع کار با شبکه های RBF

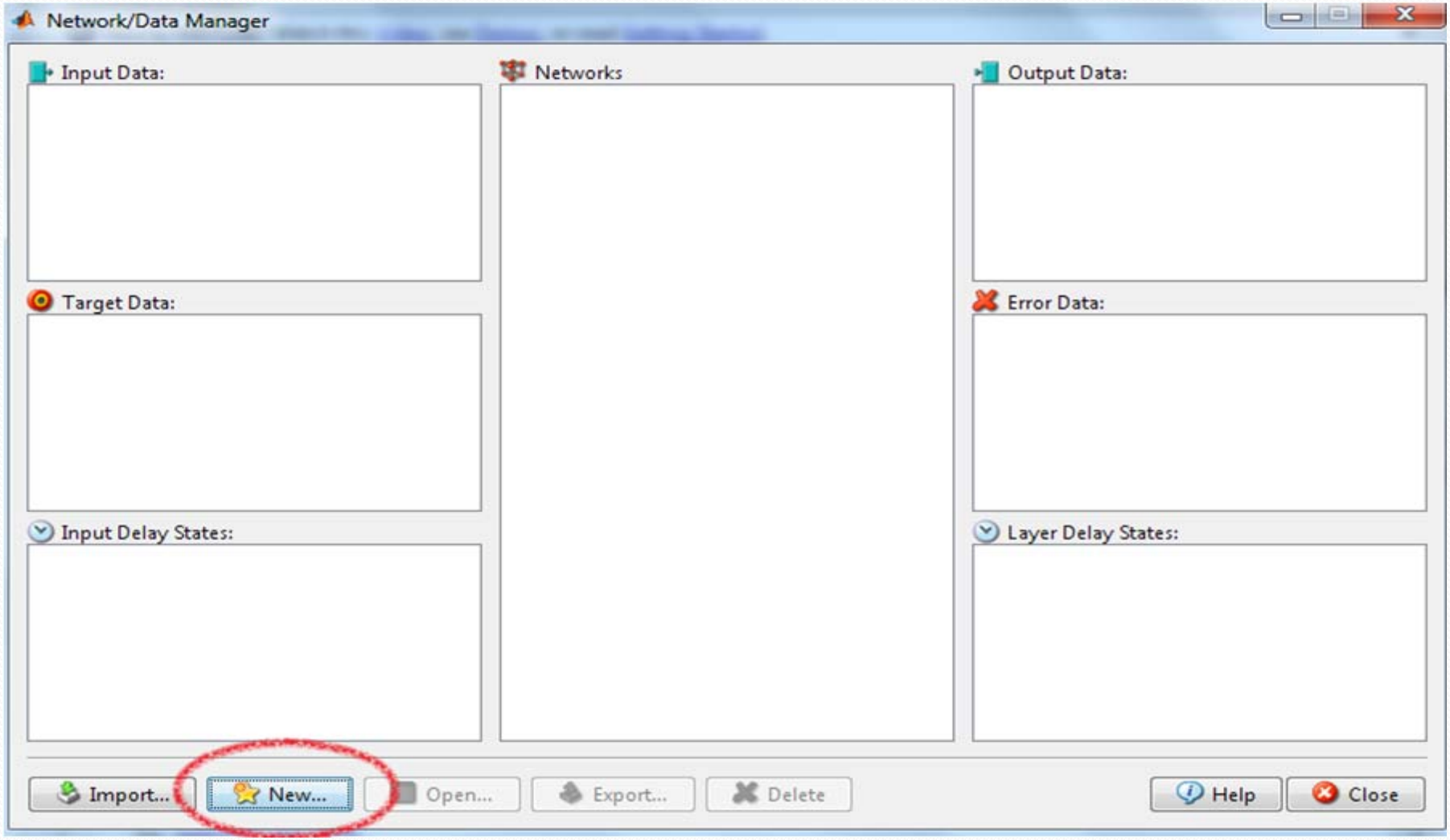
- ساخت شبکه RBF:

**Newrb**

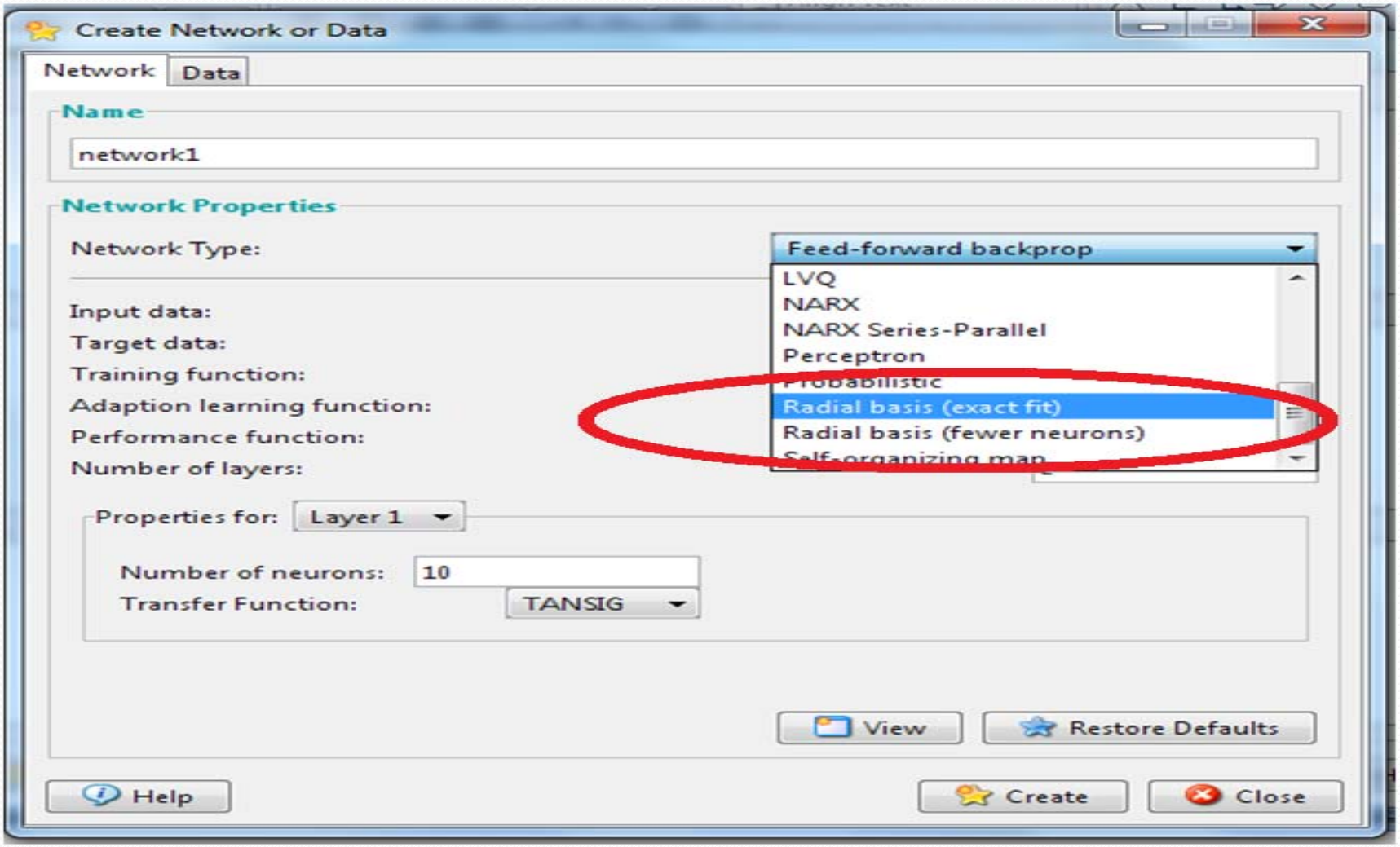
- ساخت شبکه RBF دقیق:

**Newrbe**

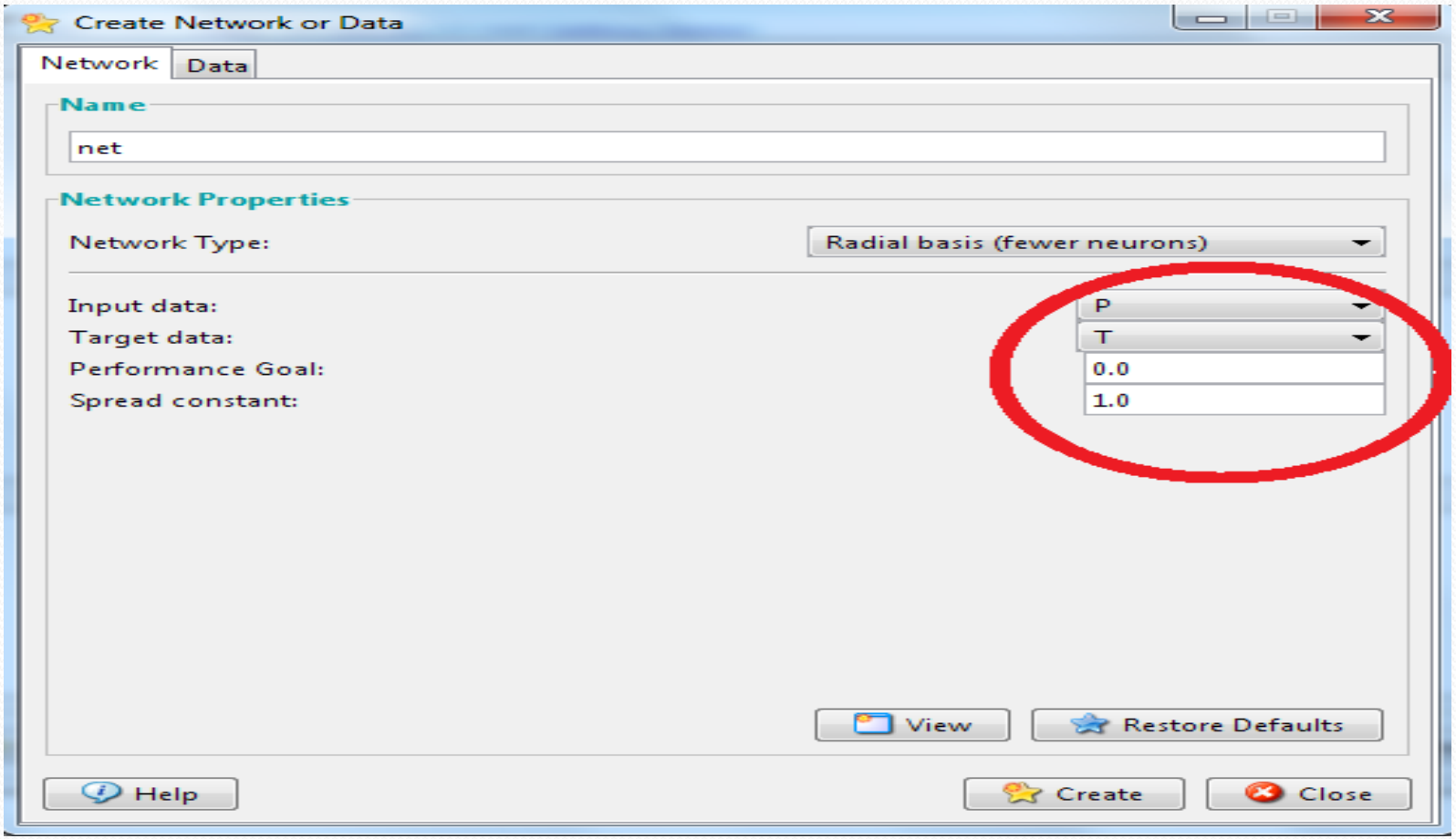
# nntool


Network/Data Manager
Input Data:
Networks
Output Data:
Target Data:
Error Data:
Input Delay States:
Layer Delay States:
Import...
New...
Open...
Export...
Delete
Help
Close

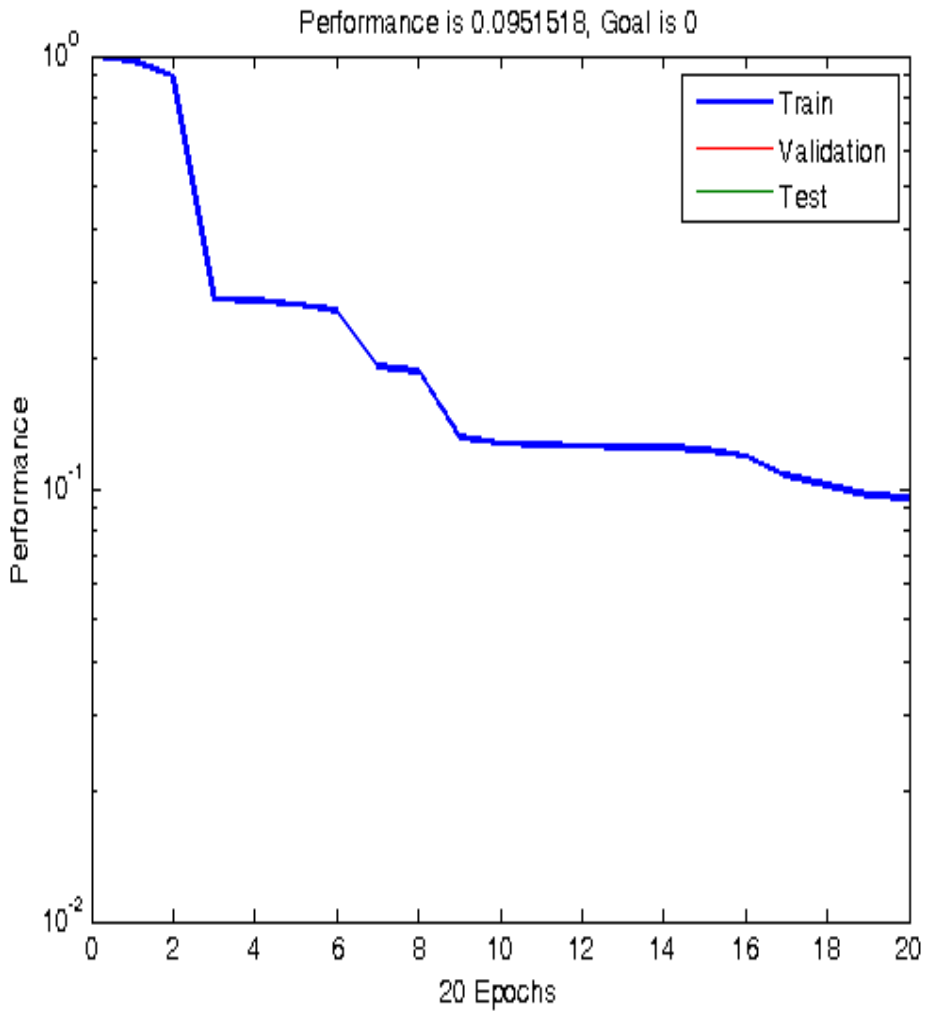
# انتخاب نوع شبکه



# انتخاب ورودي و خروجي شبكه براي شبكه RBF

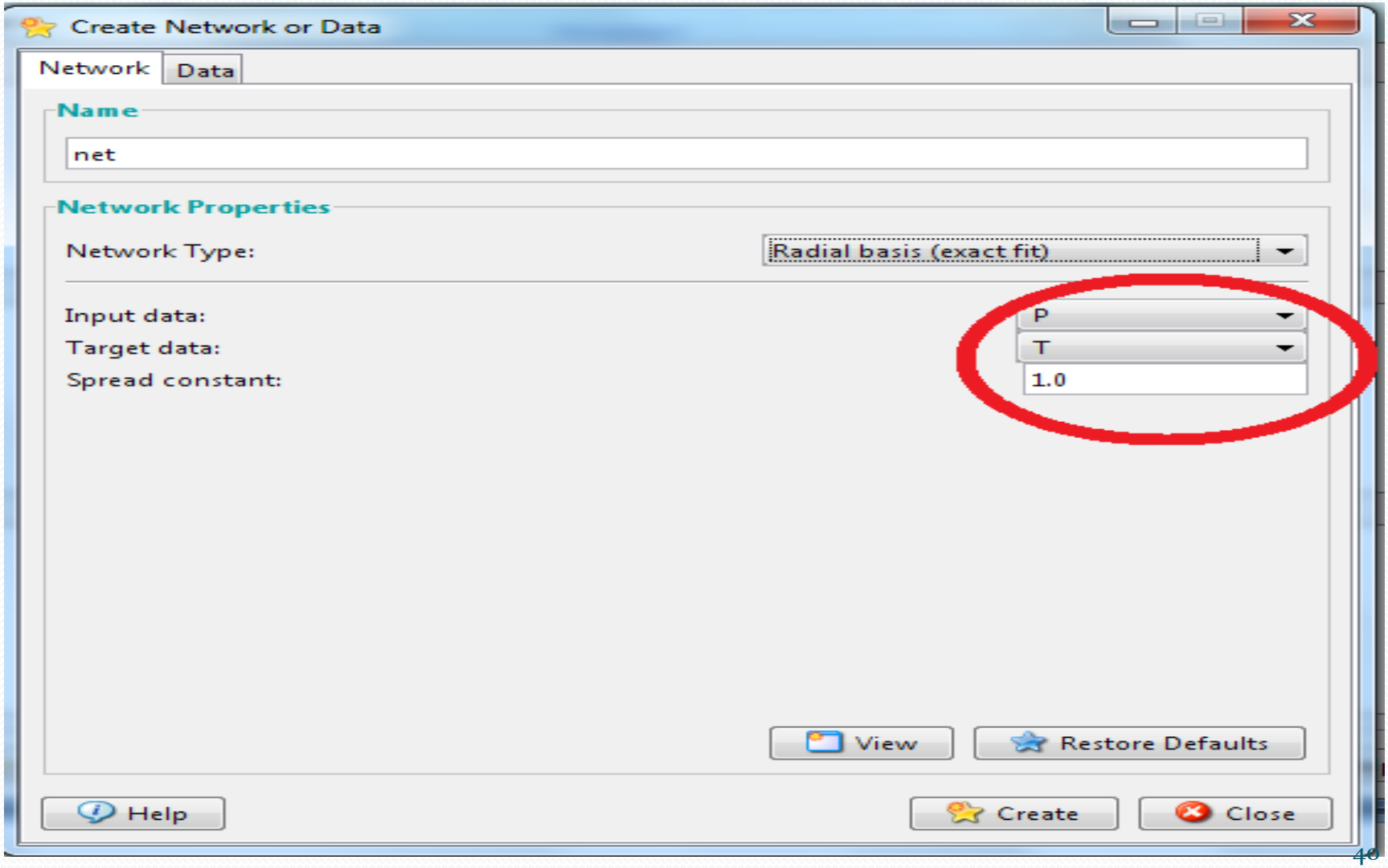


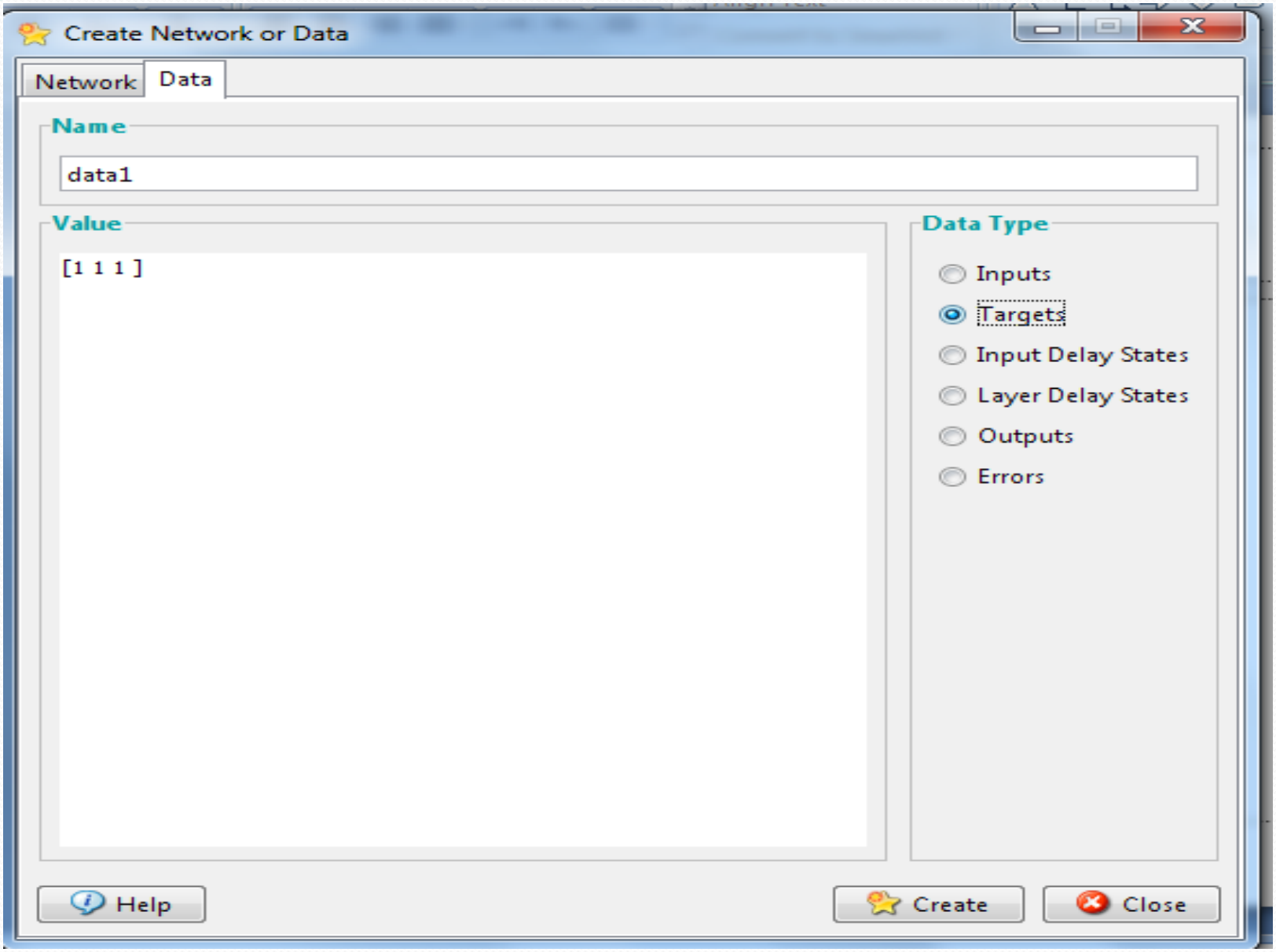
# رسم نمودار پس از ساخت شبکه




- NEWRB, neurons = 0, MSE = 1
- NEWRB, neurons = 4, MSE = 0.271592
- NEWRB, neurons = 8, MSE = 0.185994
- NEWRB, neurons = 12, MSE = 0.125461
- NEWRB, neurons = 16, MSE = 0.118941
- NEWRB, neurons = 20, MSE = 0.0951518

# انتخاب ورودي و خروجي شبكه براي شبكه exact RBF




Create Network or Data
Network
Data
Name
data1
Value
[1 1 1 ]
Data Type
Inputs
Targets
Input Delay States
Layer Delay States
Outputs
Errors
Help
Create
Close

# دستور newrb

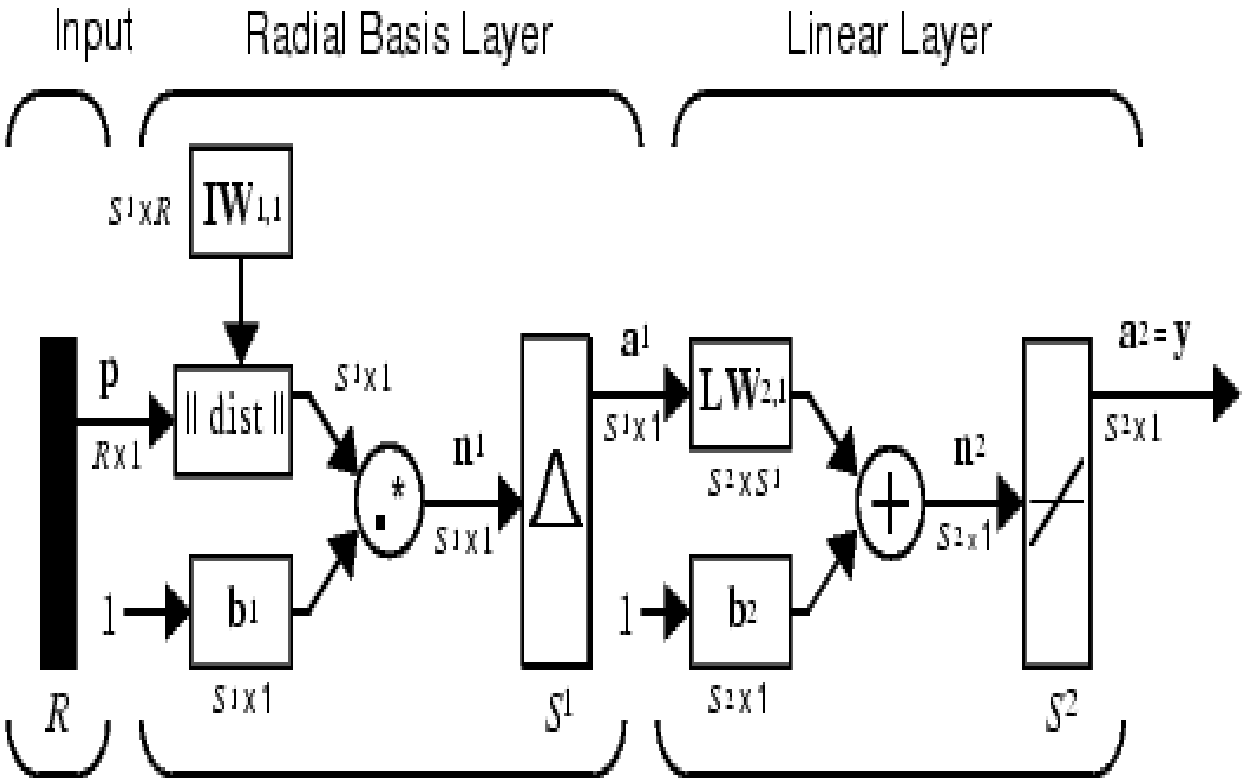
- ساخت شبکه RBF

این دستور یک شبکه با دولایه مي سازد . لايه اول نرون هايي با با تابع گوسين و وزن هاي ورودي را نيز با تابع DIST (تابع فاصله وزن اقليدسي ) و لايه دوم با يک تابع خطي و محاسبه وزن هاي آن با يک تابع ضرب نقطه اي .

از اين دستور مي توان براي تخمين توابع استفاده کرد .

اين دستور به سرعت يک شبکه RBF بر اساس داده هاي ورودي طراحي مي کند .

در اين دستور شبکه RBF با ساختن يک نرون شروع مي شود و تا زماني که خطا به حداقل خود برسد يا نرون ها به حداکثر برسند به ساختن شبکه ادامه ميدهد و در هر مرحله خطا را چک کرده و اگر به ميزان کافي نرسيده بود به شبکه يک نرون ديگر اضافه مي کند.




Where...

$R$ = number of elements in input vector

$S1$ = number of neurons in layer 1

$S2$ = number of neurons in layer 2

$$a1_i = radbas(\| IW1,1_i - p \| b1_i) \qquad a2 = purelin(LW2,1\ a1 + b2)$$

$a1_i$ is $i^{th}$ element of $a1$ where $IW1,1_i$ is a vector made of the $i^{th}$ row of $IW1,1$

# newrb دستور

- **Syntax**

  [net,tr] = newrb(P,T,goal,spread,MN,DF)

  P: R x Q matrix of Q input vectors

  T:  S x Q matrix of Q target class vectors

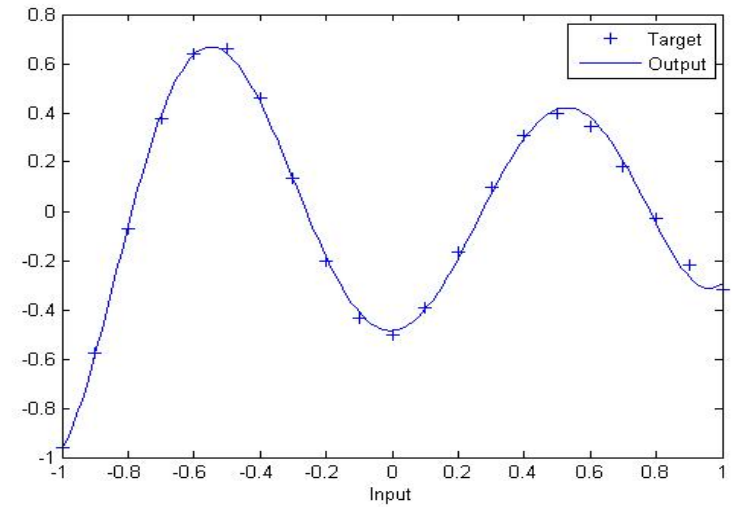
  goal: Mean squared error goal (default = 0.0)

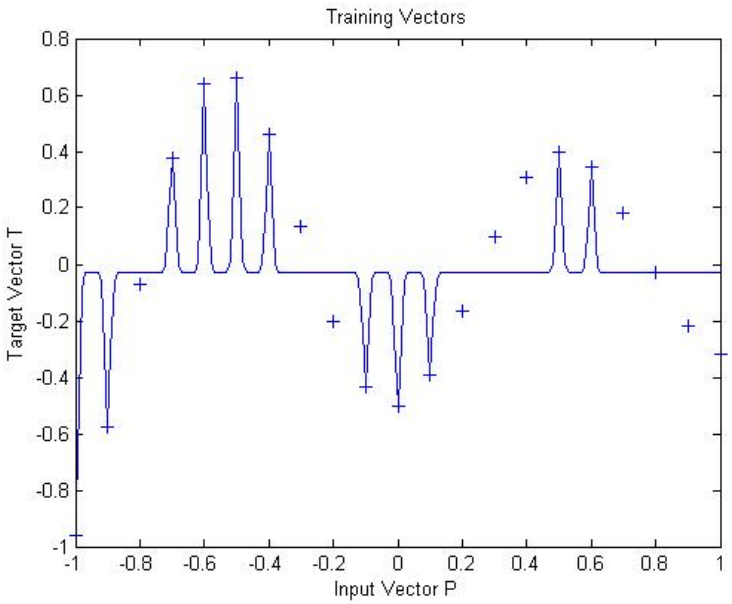
  spread: Spread of radial basis functions (default = 1.0)

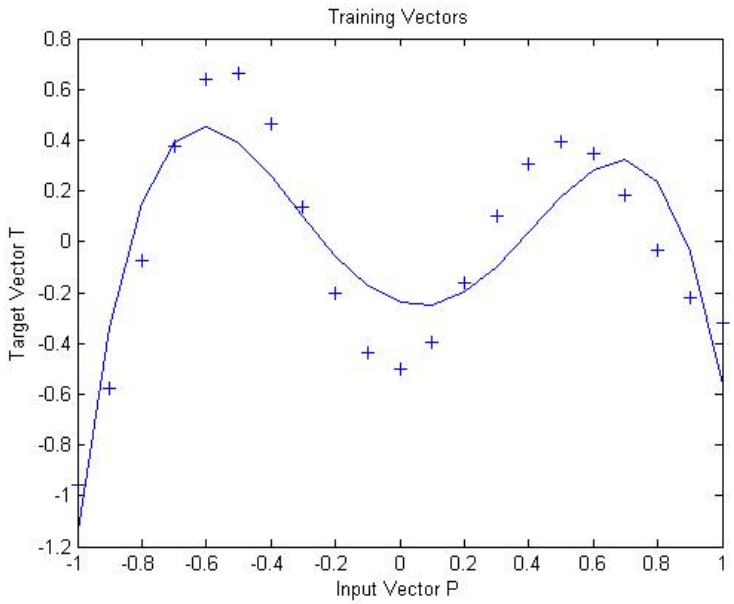
  MN: Maximum number of neurons (default is Q)

  DF: Number of neurons to add between displays (default = 25)

# تاثير انتخاب مقدار Spread براي تخمين توابع



# دستور Newrbe

نحوه به كارگيري دستور

net = newrbe(P,T,spread)

اين دستور يک شبکه با دولايه مي سازد . لايه اول نرون هايي با با تابع گوسين و وزن هاي ورودي را نيز با تابع DIST (تابع فاصله وزن اقليدسي ) و لايه دوم با يک تابع خطي و محاسبه وزن هاي آن با يک تابع ضرب نقطه اي .

از اين دستور مي توان براي تخمين توابع استفاده کرد .

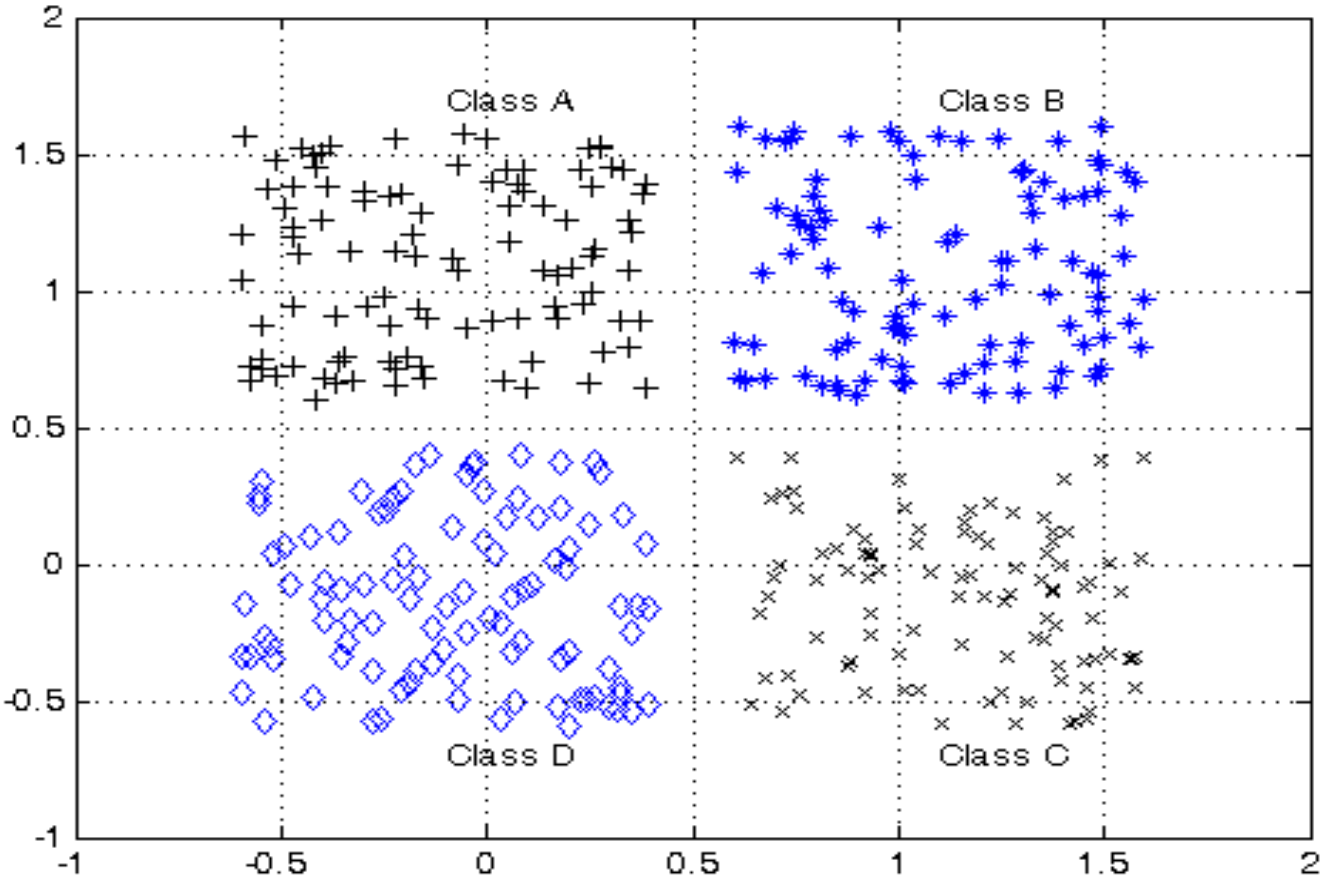
اين دستور به سرعت يک شبکه RBF با خطاي صفر طراحي مي کند . بر اساس داده هاي ورودي

P: ماتريس ورودي به شبکه است.

T: ماتريس خروجي شبکه است.

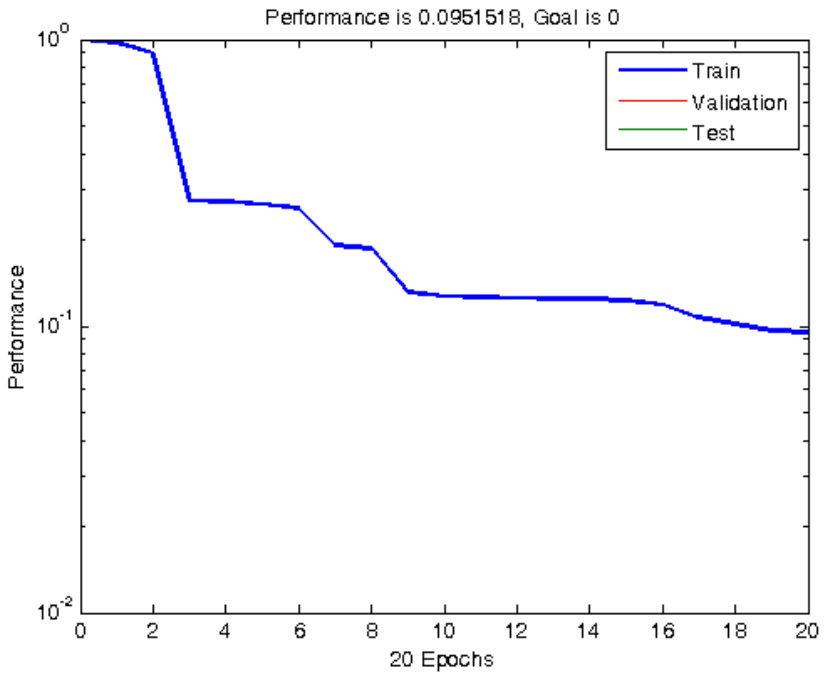
Spread :انتشار تابع RBF که مقدار پيش فرض آن 1 است.

# Classification of an XOR problem with a RBFN



# Newrbe

- spread = 1;
- net = newrbe(P,T,spread);



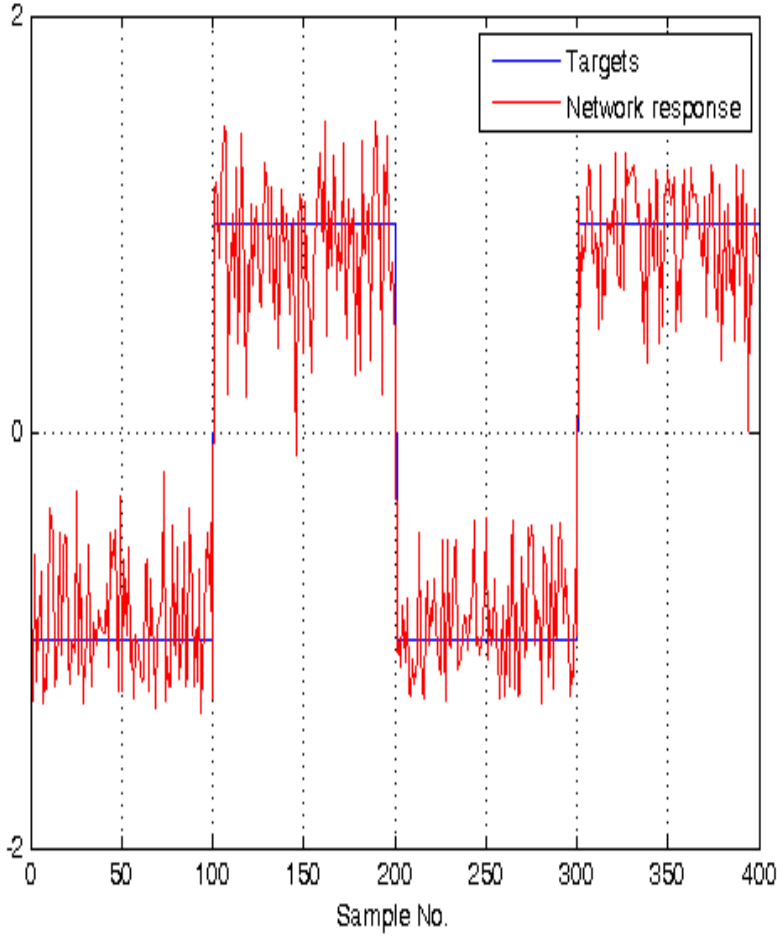

# newrb

- spread = 2;
- K = 20;
- goal = 0;
- Ki = 4;
- net = newrb(P,T,goal,spread,K,Ki);

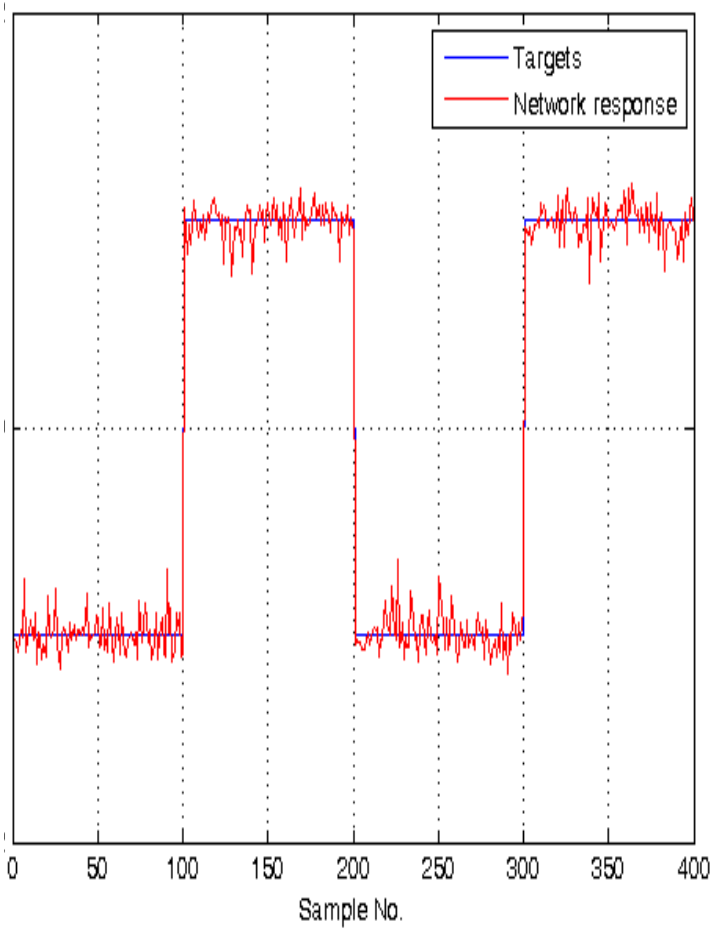
- NEWRB, neurons = 0, MSE = 1
- NEWRB, neurons = 4, MSE = 0.271592
  NEWRB, neurons = 8, MSE = 0.185994
  NEWRB, neurons = 12, MSE = 0.125461
  NEWRB, neurons = 16, MSE = 0.118941
  NEWRB, neurons = 20, MSE = 0.0951518

# نمودار خروجي شبكه و مقادير هدف

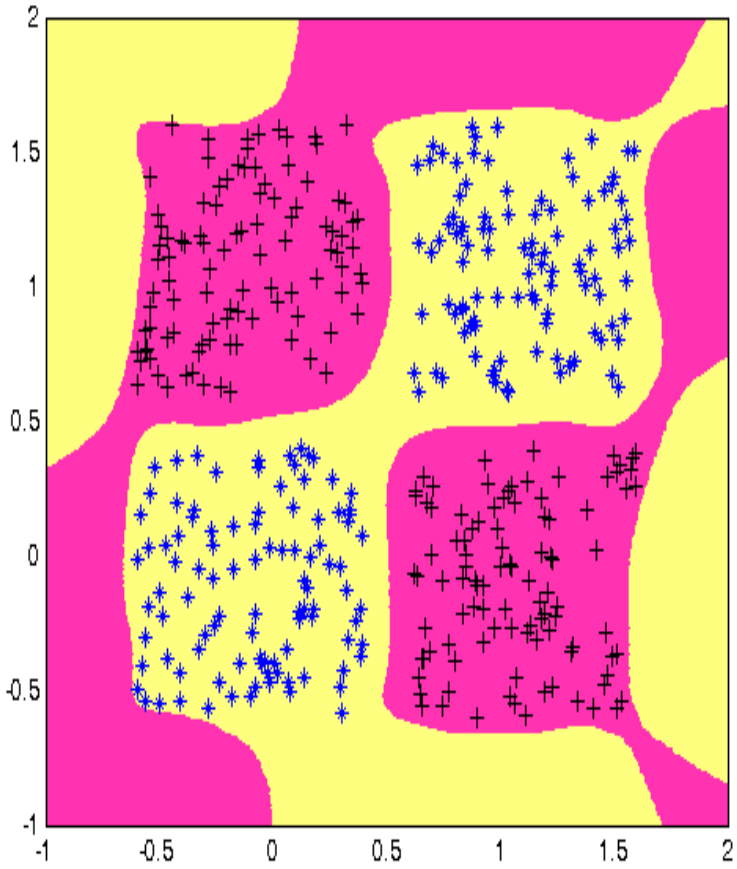
newrb (20)




newrbe(400)



# Newrbe




# newrb

