MACHINE LEARNING

1211635

MAHASARAKHAM
UNIVERSITY

NEURAL NETWORKS

Using MATLAB

Olarik Surinta, PhD.
Lecturer

Neural network toolbox

- *Wine classification* with neural network pattern recognition tool
- Pattern recognition is the process of training a neural network to assign the correct target classes to a set of input patterns.
- Once trained the network can be used to classify patterns it has not seen before.

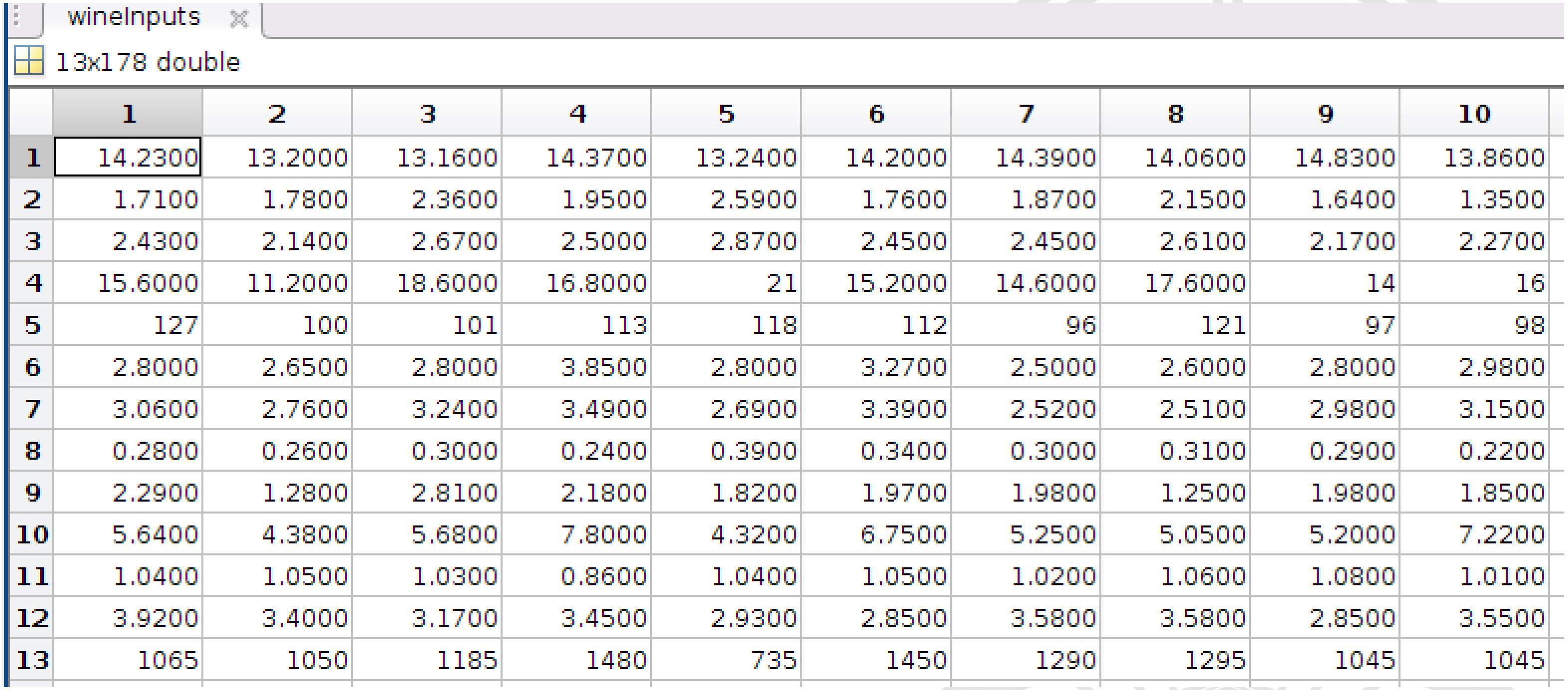
Wine dataset

- This dataset can be used to create a neural network that classifies wines from ***three wineries*** in Italy based on constituents found through chemical analysis.
- This dataset including 13 attributes and 178 instances.

Wine dataset - attribute

1. Alcohol
2. Malic acid
3. Ash
4. Alcalinity of ash
5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavanoid phenols
9. Proanthocyanins
10. Color intensity
11. Hue
12. OD280/OD315 of diluted wines
13. Proline

Wine dataset - attribute



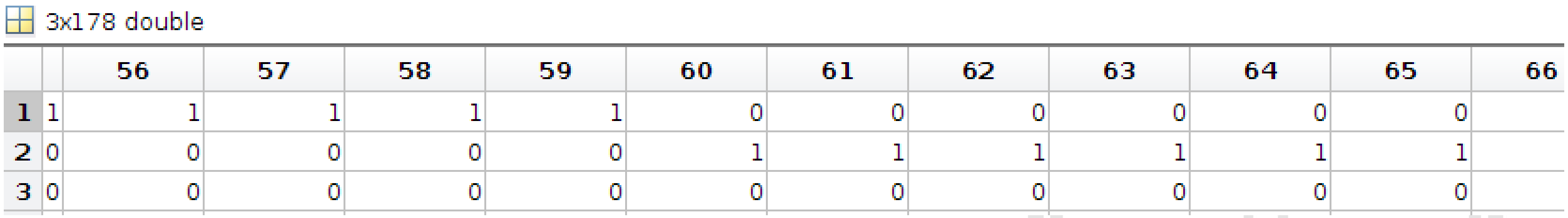
13x178 double

	1	2	3	4	5	6	7	8	9	10
1	14.2300	13.2000	13.1600	14.3700	13.2400	14.2000	14.3900	14.0600	14.8300	13.8600
2	1.7100	1.7800	2.3600	1.9500	2.5900	1.7600	1.8700	2.1500	1.6400	1.3500
3	2.4300	2.1400	2.6700	2.5000	2.8700	2.4500	2.4500	2.6100	2.1700	2.2700
4	15.6000	11.2000	18.6000	16.8000	21	15.2000	14.6000	17.6000	14	16
5	127	100	101	113	118	112	96	121	97	98
6	2.8000	2.6500	2.8000	3.8500	2.8000	3.2700	2.5000	2.6000	2.8000	2.9800
7	3.0600	2.7600	3.2400	3.4900	2.6900	3.3900	2.5200	2.5100	2.9800	3.1500
8	0.2800	0.2600	0.3000	0.2400	0.3900	0.3400	0.3000	0.3100	0.2900	0.2200
9	2.2900	1.2800	2.8100	2.1800	1.8200	1.9700	1.9800	1.2500	1.9800	1.8500
10	5.6400	4.3800	5.6800	7.8000	4.3200	6.7500	5.2500	5.0500	5.2000	7.2200
11	1.0400	1.0500	1.0300	0.8600	1.0400	1.0500	1.0200	1.0600	1.0800	1.0100
12	3.9200	3.4000	3.1700	3.4500	2.9300	2.8500	3.5800	3.5800	2.8500	3.5500
13	1065	1050	1185	1480	735	1450	1290	1295	1045	1045

Wine dataset – target/class

1. Vinyard #1
2. Vinyard #2
3. Vinyard #3

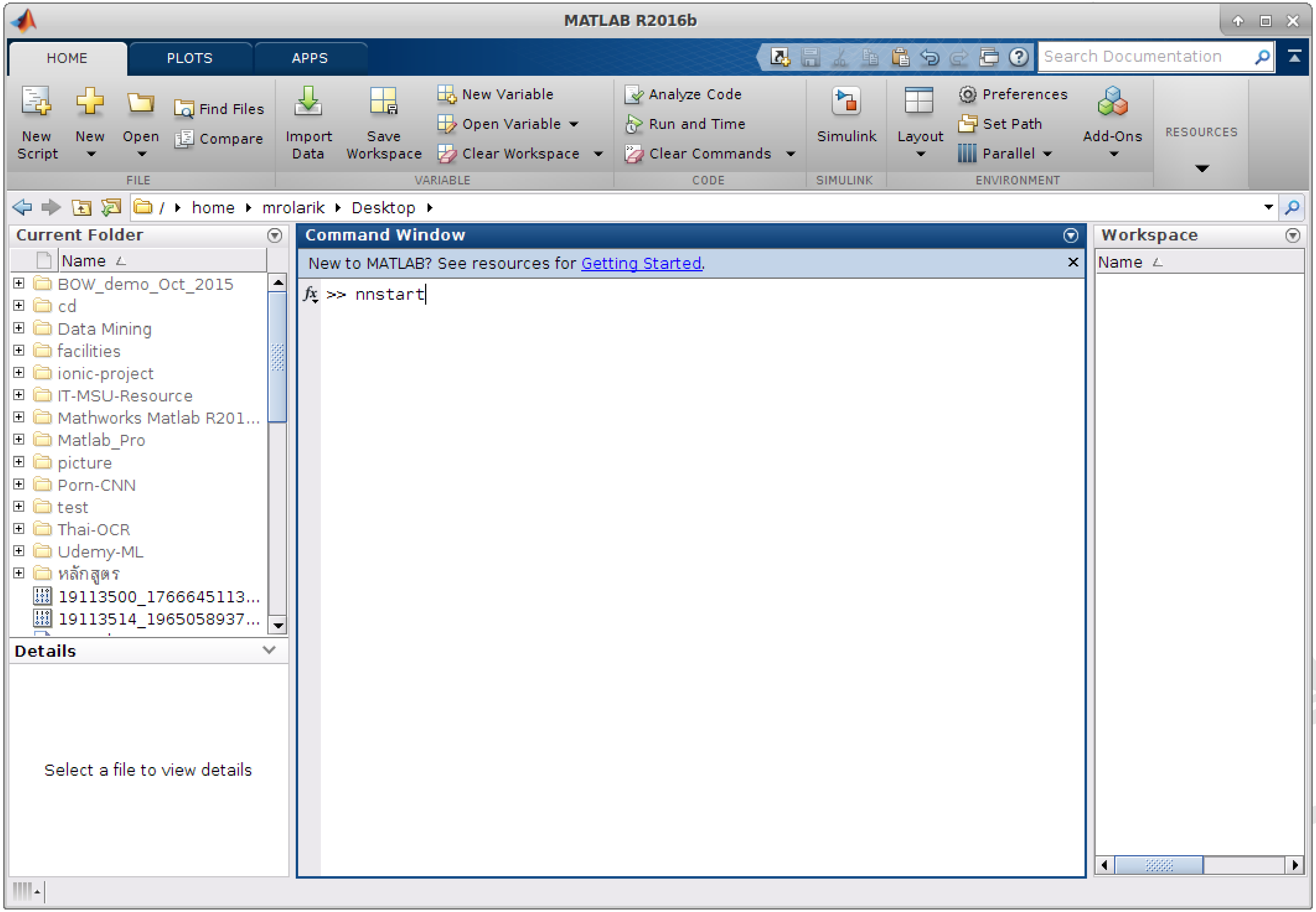
Wine dataset – target/class



3x178 double

		56	57	58	59	60	61	62	63	64	65	66
1	1	1	1	1	1	0	0	0	0	0	0	
2	0	0	0	0	0	1	1	1	1	1	1	
3	0	0	0	0	0	0	0	0	0	0	0	

MATLAB - nnstart

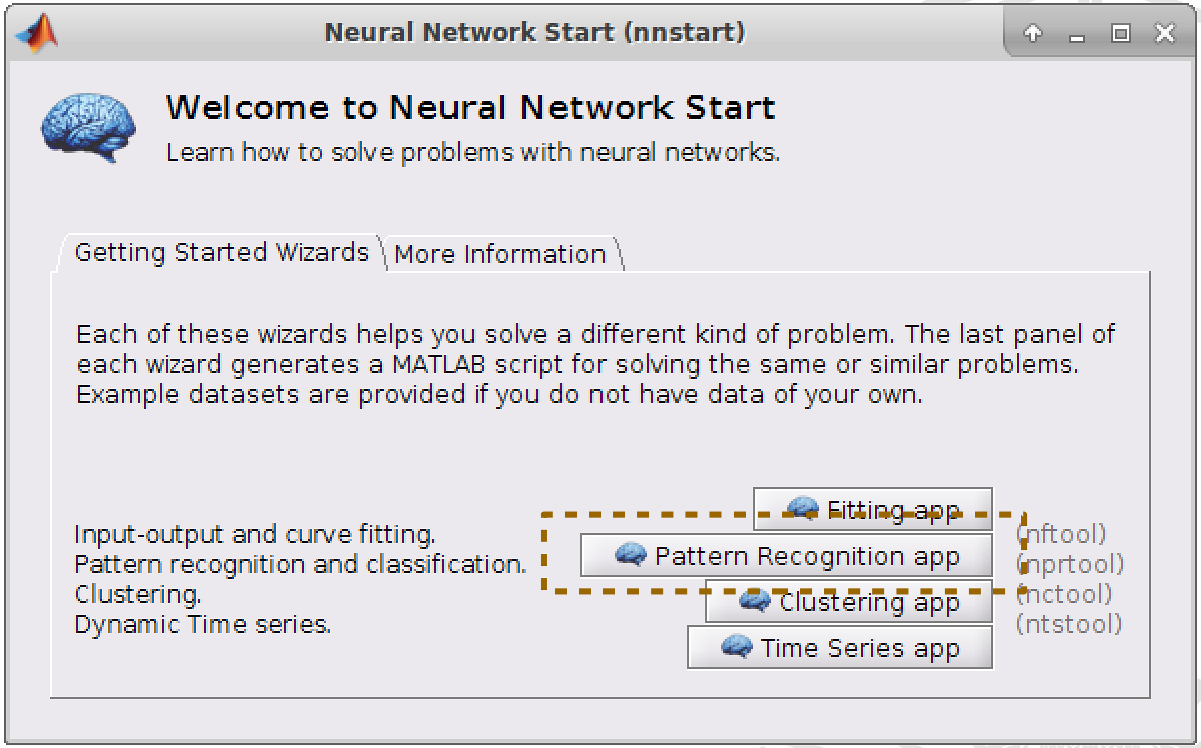


New to MATLAB? See resources for Getting Started.

```
>> nnstart
```

Select a file to view details

nnstart – pattern recognition app



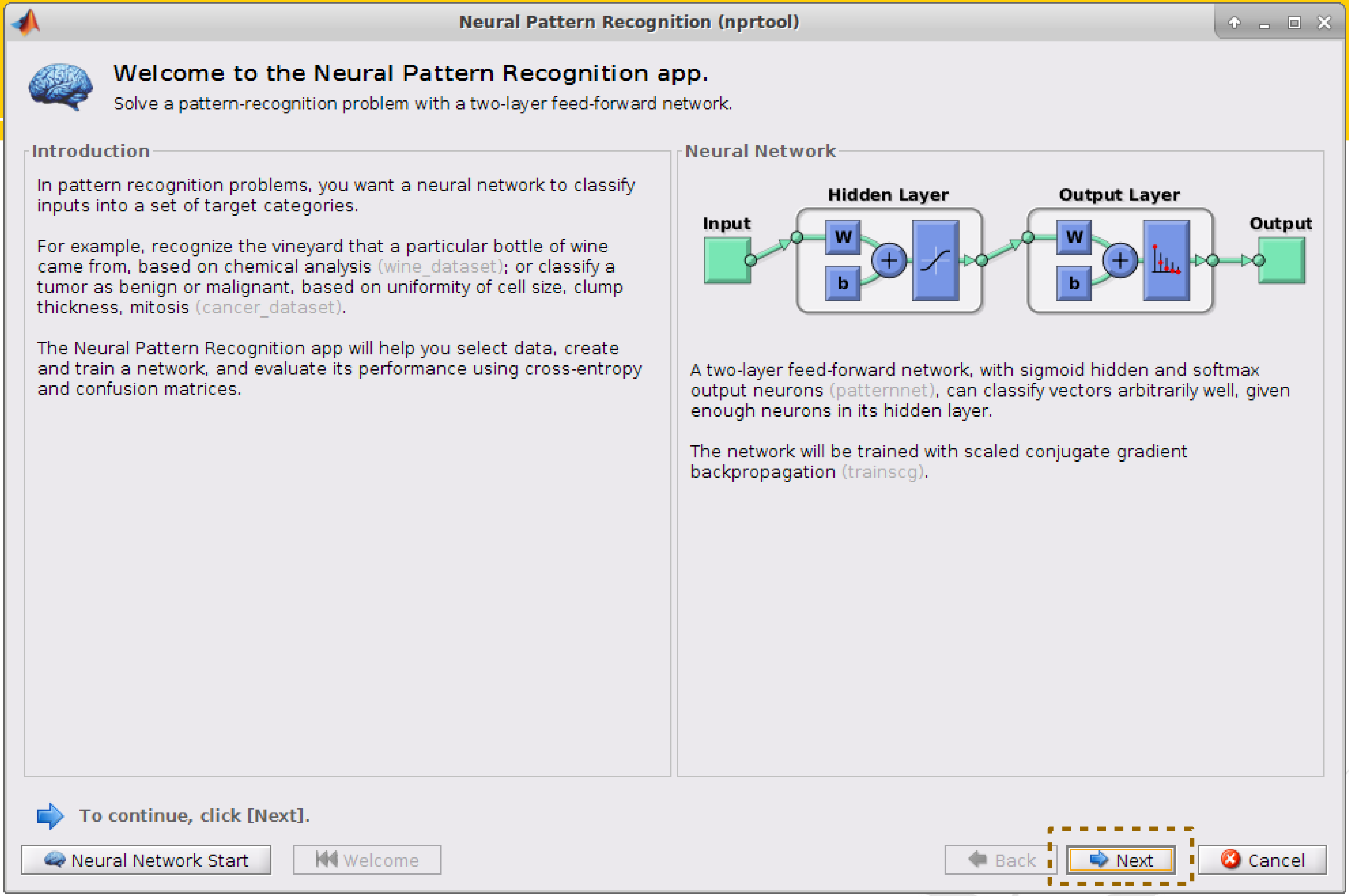
Neural Network Start (nnstart)

Welcome to Neural Network Start

Learn how to solve problems with neural networks.

Each of these wizards helps you solve a different kind of problem. The last panel of each wizard generates a MATLAB script for solving the same or similar problems. Example datasets are provided if you do not have data of your own.

Input-output and curve fitting. Fitting app (nftool)
Pattern recognition and classification. Pattern Recognition app (nprtool)
Clustering. Clustering app (nctool)
Dynamic Time series. Time Series app (ntstool)



Welcome to the Neural Pattern Recognition app.

Solve a pattern-recognition problem with a two-layer feed-forward network.

Introduction

In pattern recognition problems, you want a neural network to classify inputs into a set of target categories.

For example, recognize the vineyard that a particular bottle of wine came from, based on chemical analysis (wine_dataset); or classify a tumor as benign or malignant, based on uniformity of cell size, clump thickness, mitosis (cancer_dataset).

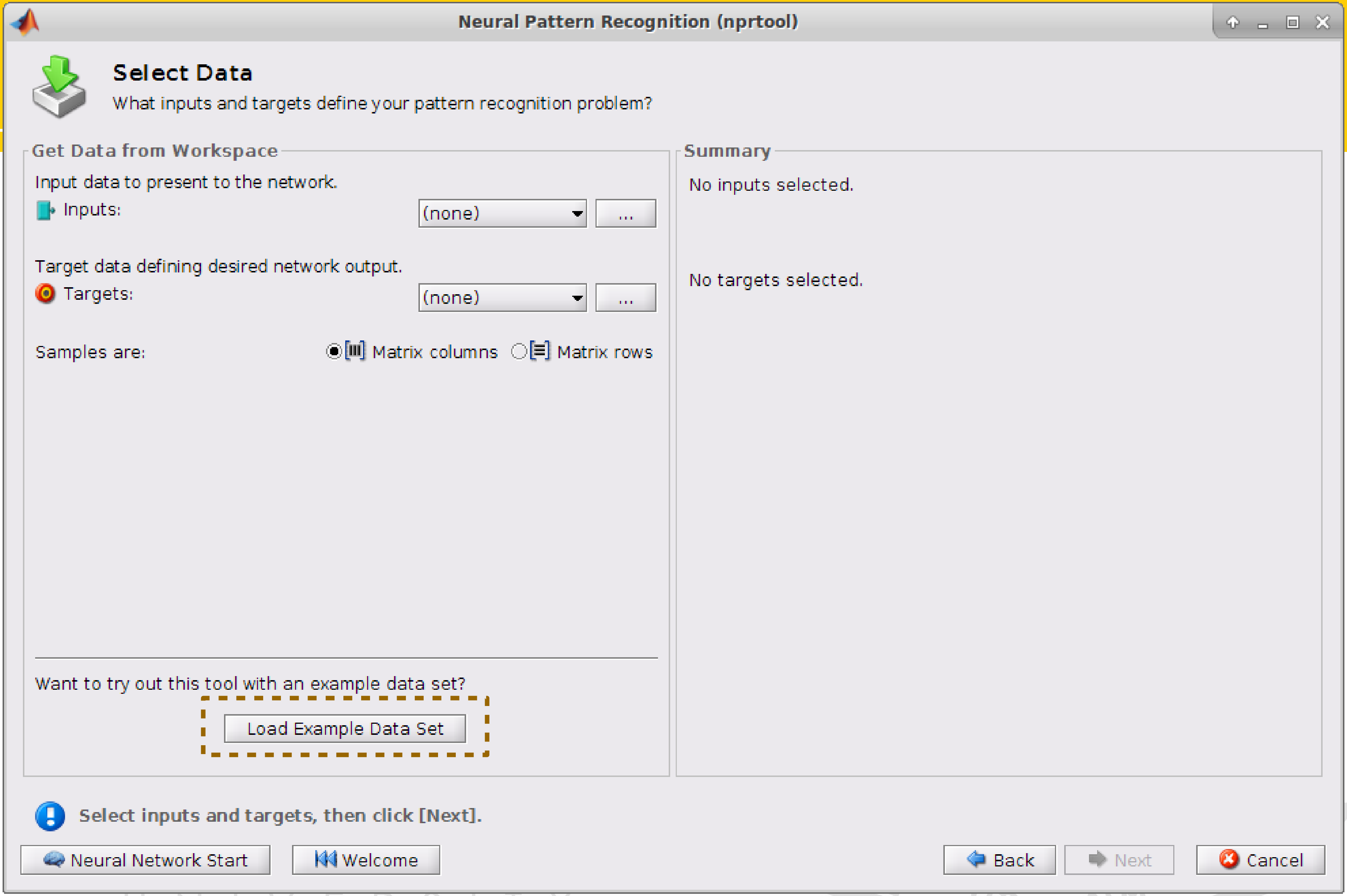
The Neural Pattern Recognition app will help you select data, create and train a network, and evaluate its performance using cross-entropy and confusion matrices.

Neural Network

A two-layer feed-forward network, with sigmoid hidden and softmax output neurons (patternnet), can classify vectors arbitrarily well, given enough neurons in its hidden layer.

The network will be trained with scaled conjugate gradient backpropagation (trainscg).

To continue, click [Next].



Select Data

What inputs and targets define your pattern recognition problem?

Get Data from Workspace

Input data to present to the network.

Inputs: (none) ...

Target data defining desired network output.

Targets: (none) ...

Samples are: ● Matrix columns ○ Matrix rows

Want to try out this tool with an example data set?

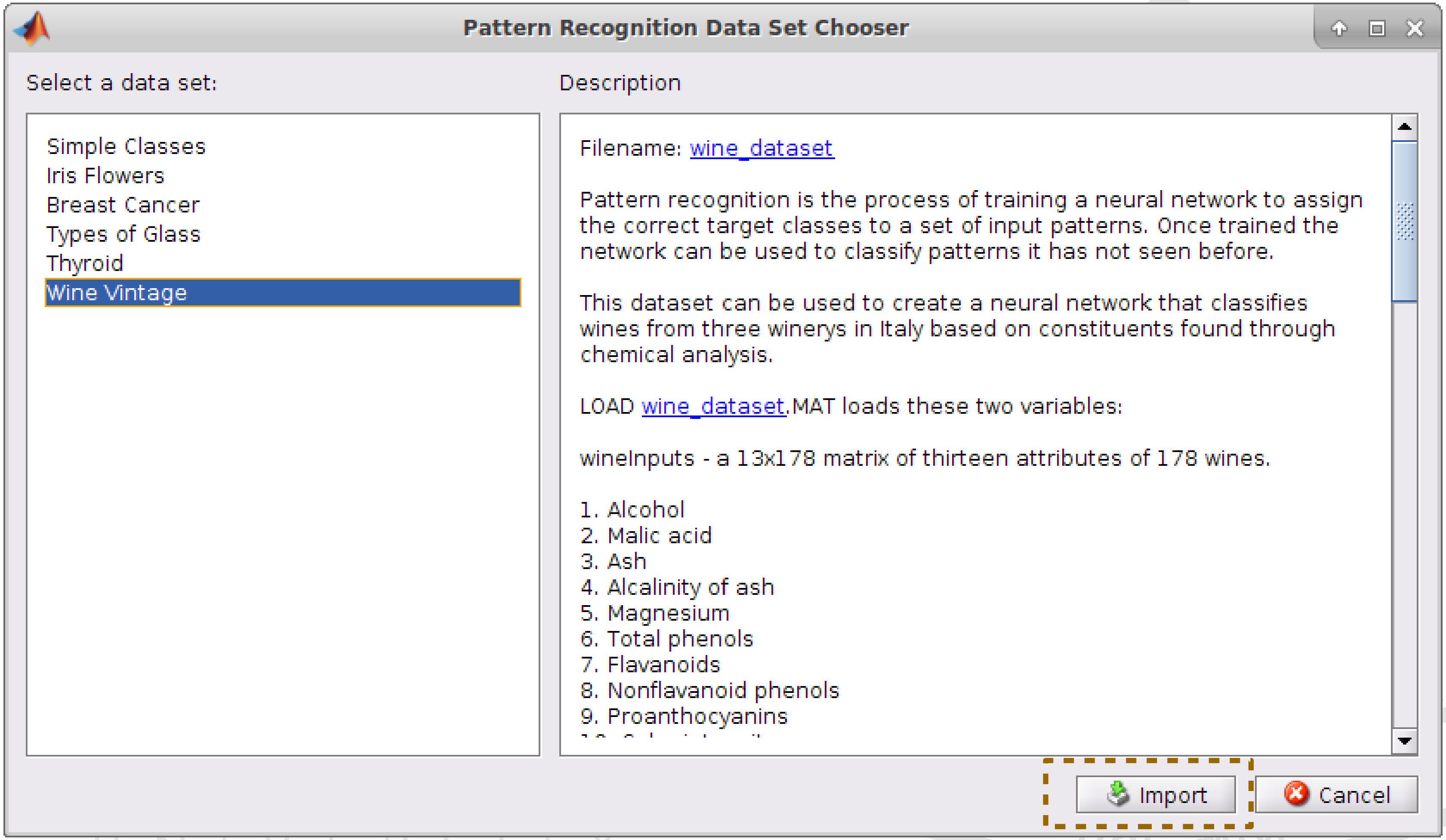
Summary

No inputs selected.

No targets selected.

Select inputs and targets, then click [Next].

Select wine dataset



Select a data set:

- Simple Classes
- Iris Flowers
- Breast Cancer
- Types of Glass
- Thyroid
- Wine Vintage

Description

Filename: wine_dataset

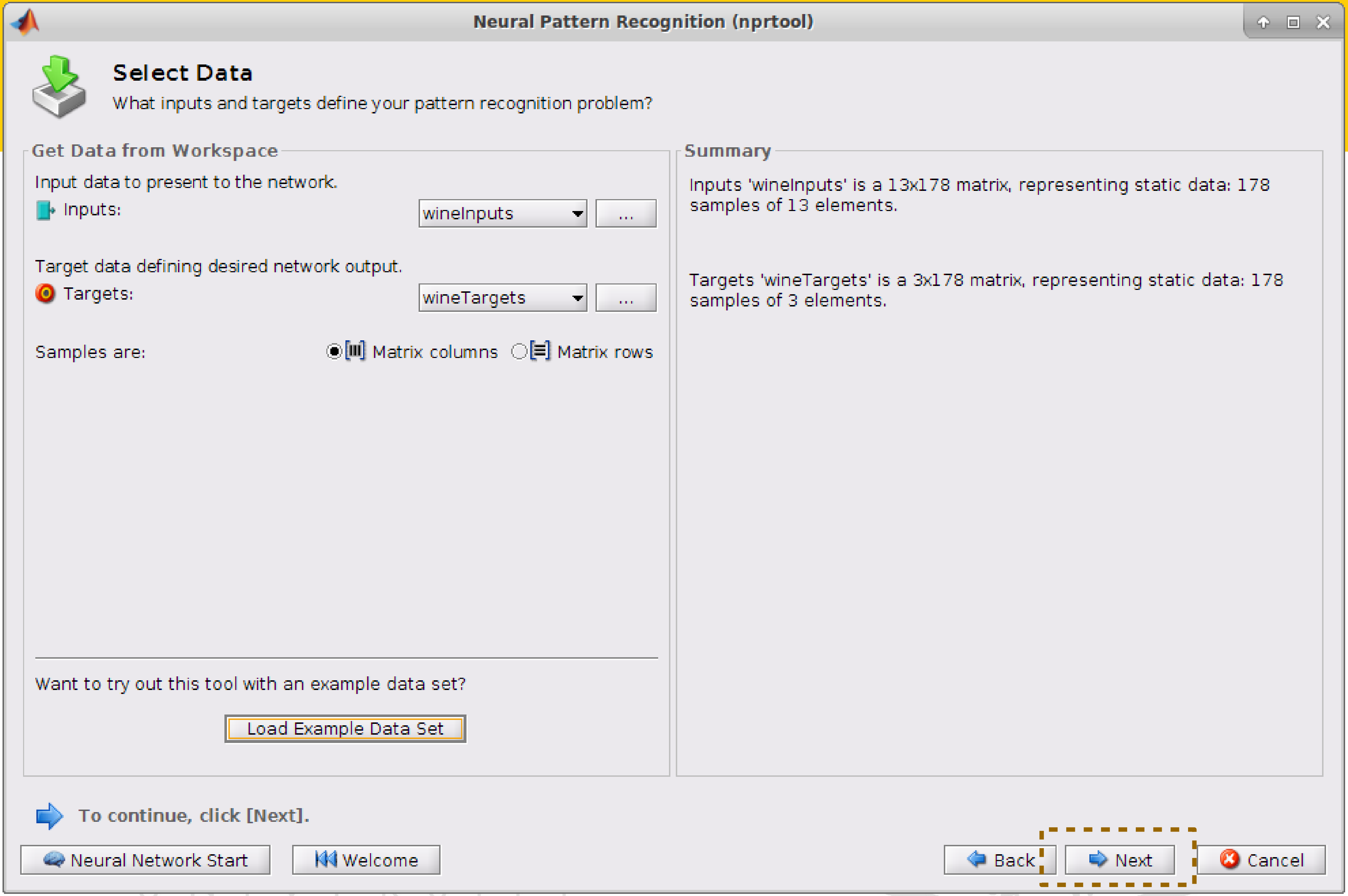
Pattern recognition is the process of training a neural network to assign the correct target classes to a set of input patterns. Once trained the network can be used to classify patterns it has not seen before.

This dataset can be used to create a neural network that classifies wines from three winerys in Italy based on constituents found through chemical analysis.

LOAD wine_dataset.MAT loads these two variables:

wineInputs - a 13x178 matrix of thirteen attributes of 178 wines.

1. Alcohol
2. Malic acid
3. Ash
4. Alcalinity of ash
5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavanoid phenols
9. Proanthocyanins



Neural Pattern Recognition (nprtool)

Select Data

What inputs and targets define your pattern recognition problem?

Get Data from Workspace

Input data to present to the network.

Inputs: wineInputs ▾ ...

Target data defining desired network output.

Targets: wineTargets ▾ ...

Samples are: ● Matrix columns ○ Matrix rows

Want to try out this tool with an example data set?

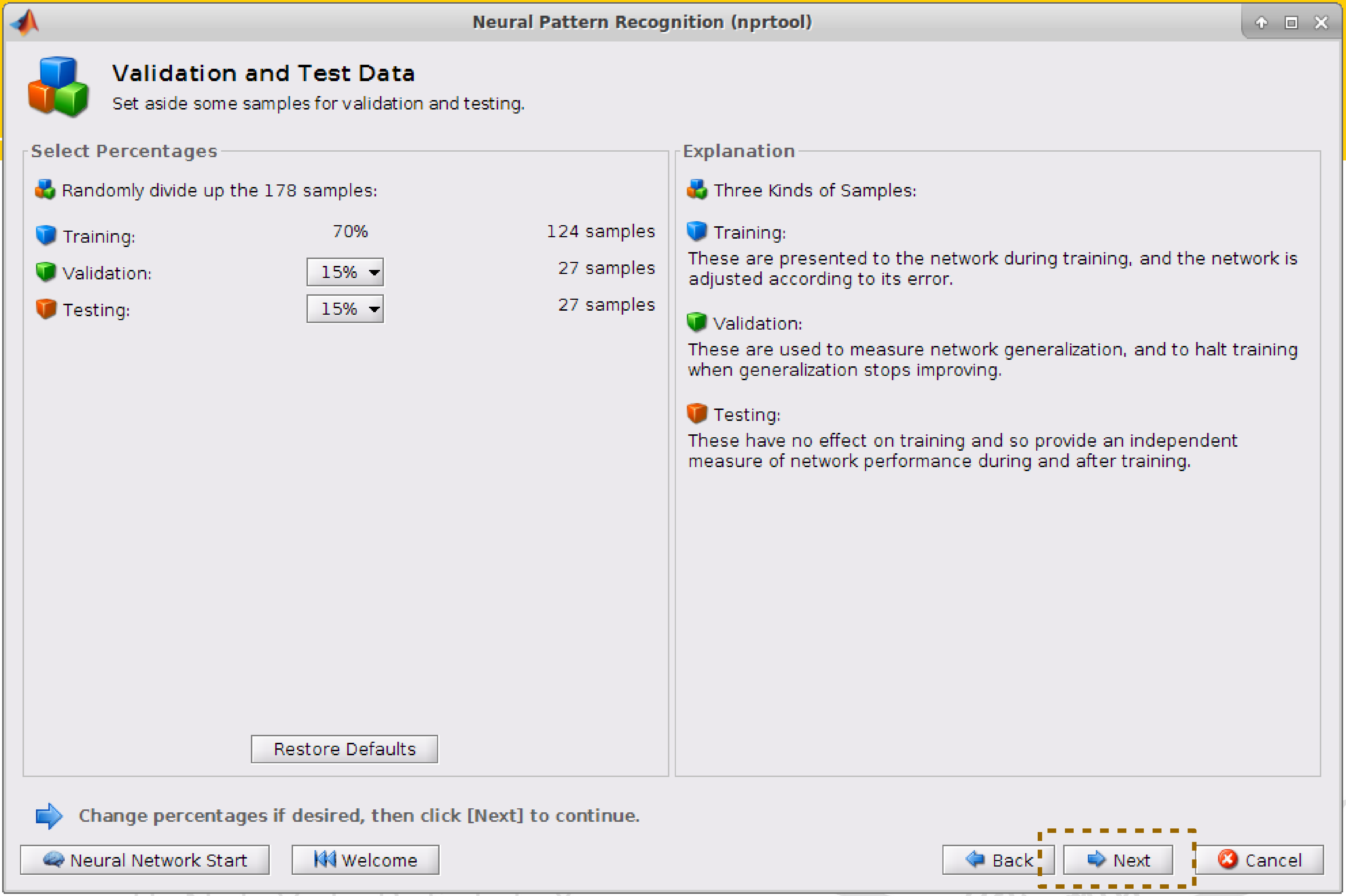
Load Example Data Set

Summary

Inputs 'wineInputs' is a 13x178 matrix, representing static data: 178 samples of 13 elements.

Targets 'wineTargets' is a 3x178 matrix, representing static data: 178 samples of 3 elements.

To continue, click [Next].



Neural Pattern Recognition (nprtool)

Validation and Test Data

Set aside some samples for validation and testing.

Select Percentages

Randomly divide up the 178 samples:

Training:	70%	124 samples
Validation:	15% ▾	27 samples
Testing:	15% ▾	27 samples

Explanation

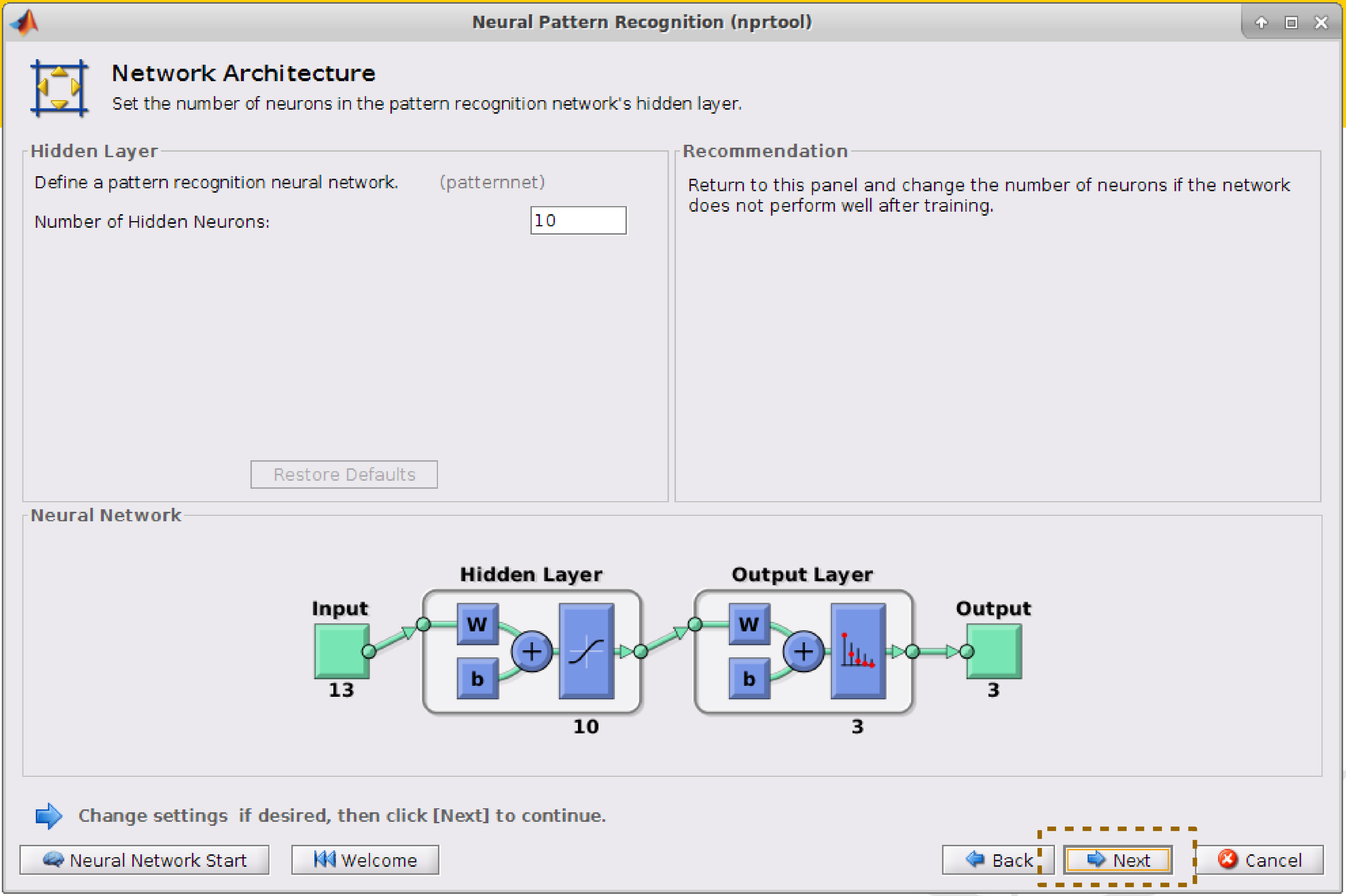
Three Kinds of Samples:

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.

Change percentages if desired, then click [Next] to continue.



Neural Pattern Recognition (nprtool)

Network Architecture

Set the number of neurons in the pattern recognition network's hidden layer.

Hidden Layer

Define a pattern recognition neural network. (patternnet)

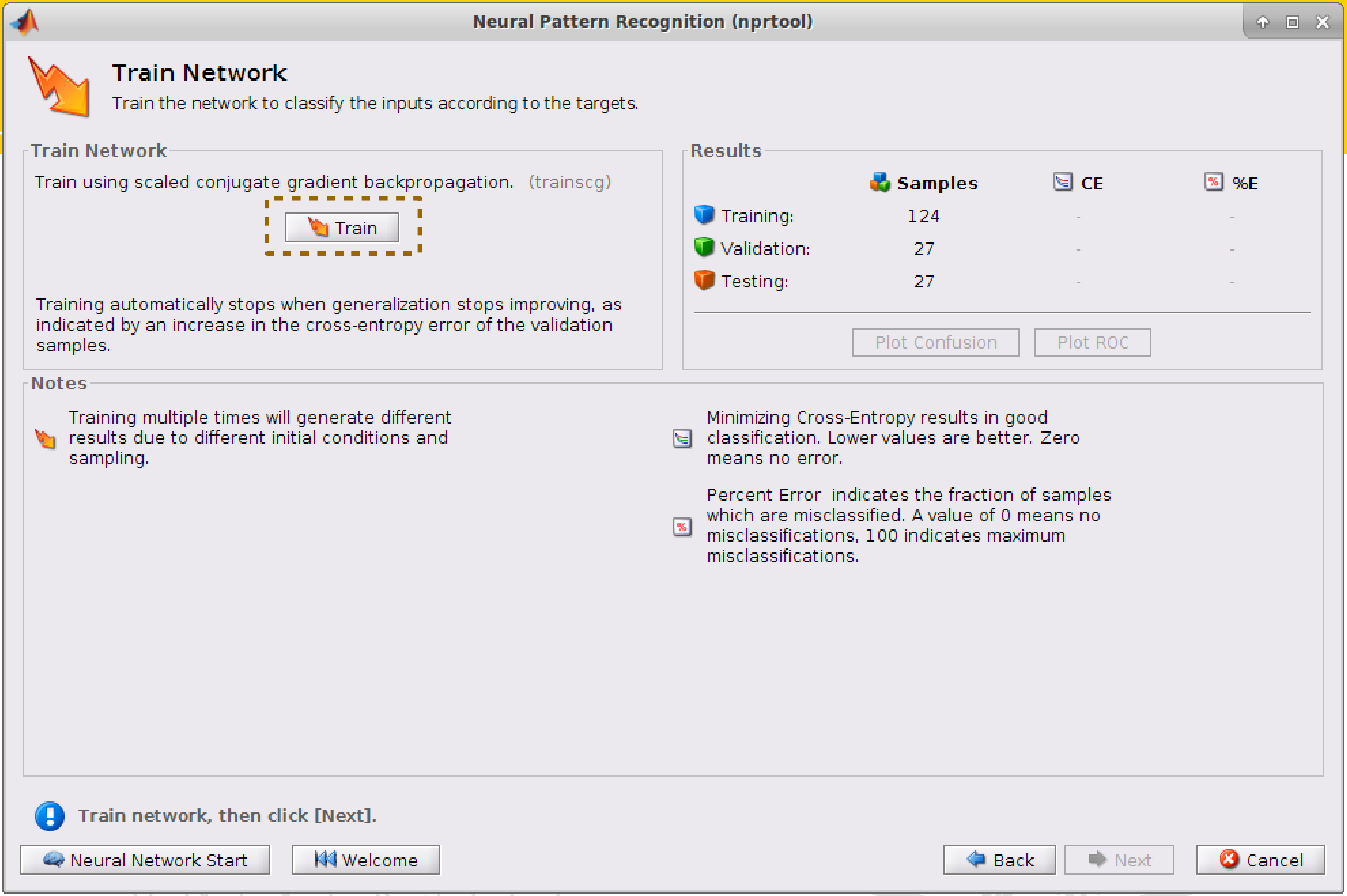
Number of Hidden Neurons: 10

Recommendation

Return to this panel and change the number of neurons if the network does not perform well after training.

Neural Network

Change settings if desired, then click [Next] to continue.



Train Network

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

Train Network

Train using scaled conjugate gradient backpropagation. (trainscg)

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:	124	-	-
Validation:	27	-	-
Testing:	27	-	-

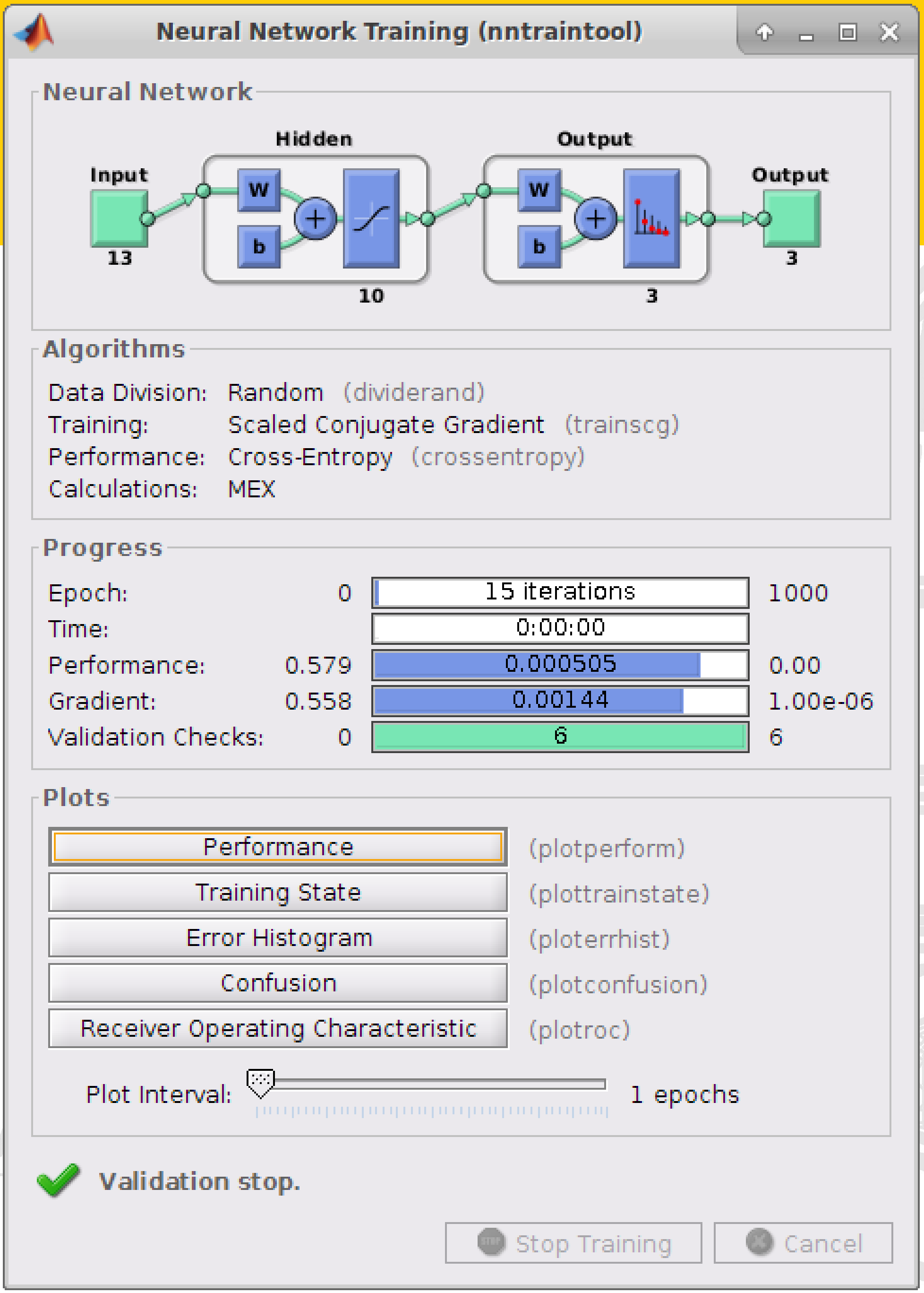
Notes

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.

Train network, then click [Next].



Algorithms

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

Progress

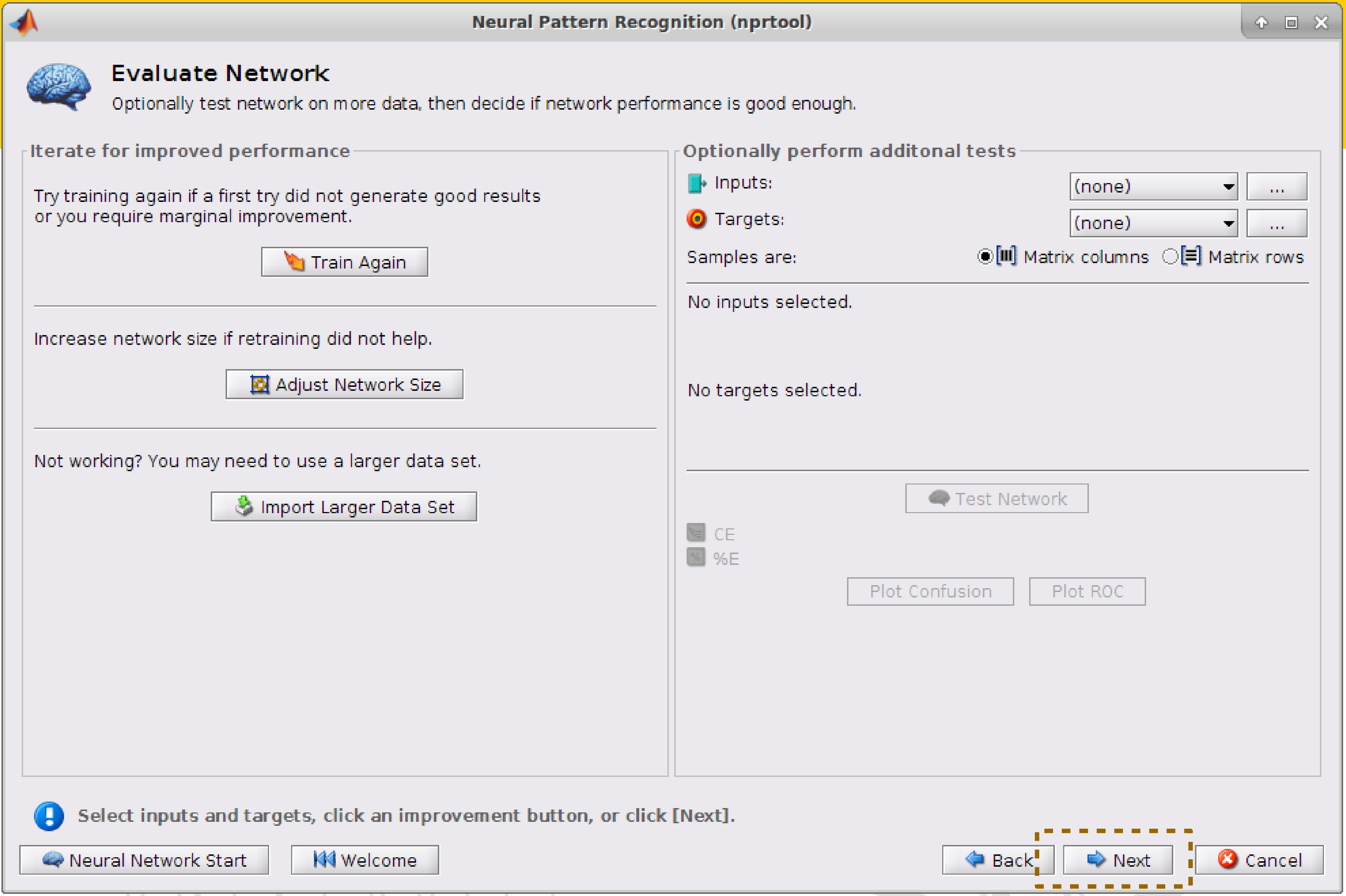
Epoch:	0	15 iterations	1000
Time:		0:00:00	
Performance:	0.579	0.000505	0.00
Gradient:	0.558	0.00144	1.00e-06
Validation Checks:	0	6	6

Plots

(plotperform)
(plottrainstate)
(ploterrhist)
(plotconfusion)
(plotroc)

Plot Interval: 1 epochs

✓ Validation stop.



Neural Pattern Recognition (nprtool)

Evaluate Network

Optionally test network on more data, then decide if network performance is good enough.

Iterate for improved performance

Try training again if a first try did not generate good results or you require marginal improvement.

Increase network size if retraining did not help.

Not working? You may need to use a larger data set.

Optionally perform additonal tests

Inputs: (none)

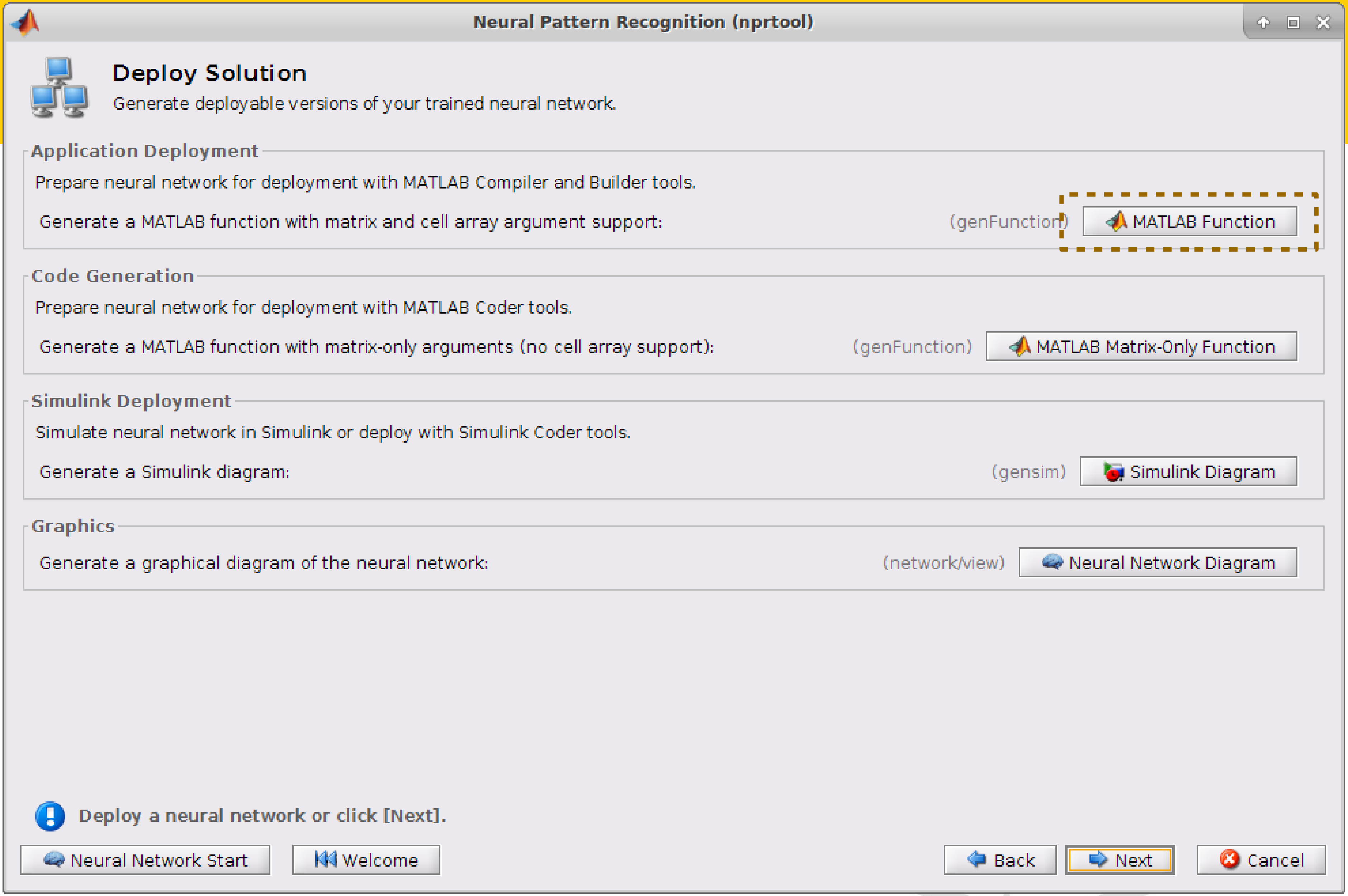
Targets: (none)

Samples are: ● Matrix columns ○ Matrix rows

No inputs selected.

No targets selected.

Select inputs and targets, click an improvement button, or click [Next].



Neural Pattern Recognition (nprtool)

Deploy Solution

Generate deployable versions of your trained neural network.

Application Deployment

Prepare neural network for deployment with MATLAB Compiler and Builder tools.

Generate a MATLAB function with matrix and cell array argument support: (genFunction) MATLAB Function

Code Generation

Prepare neural network for deployment with MATLAB Coder tools.

Generate a MATLAB function with matrix-only arguments (no cell array support): (genFunction) MATLAB Matrix-Only Function

Simulink Deployment

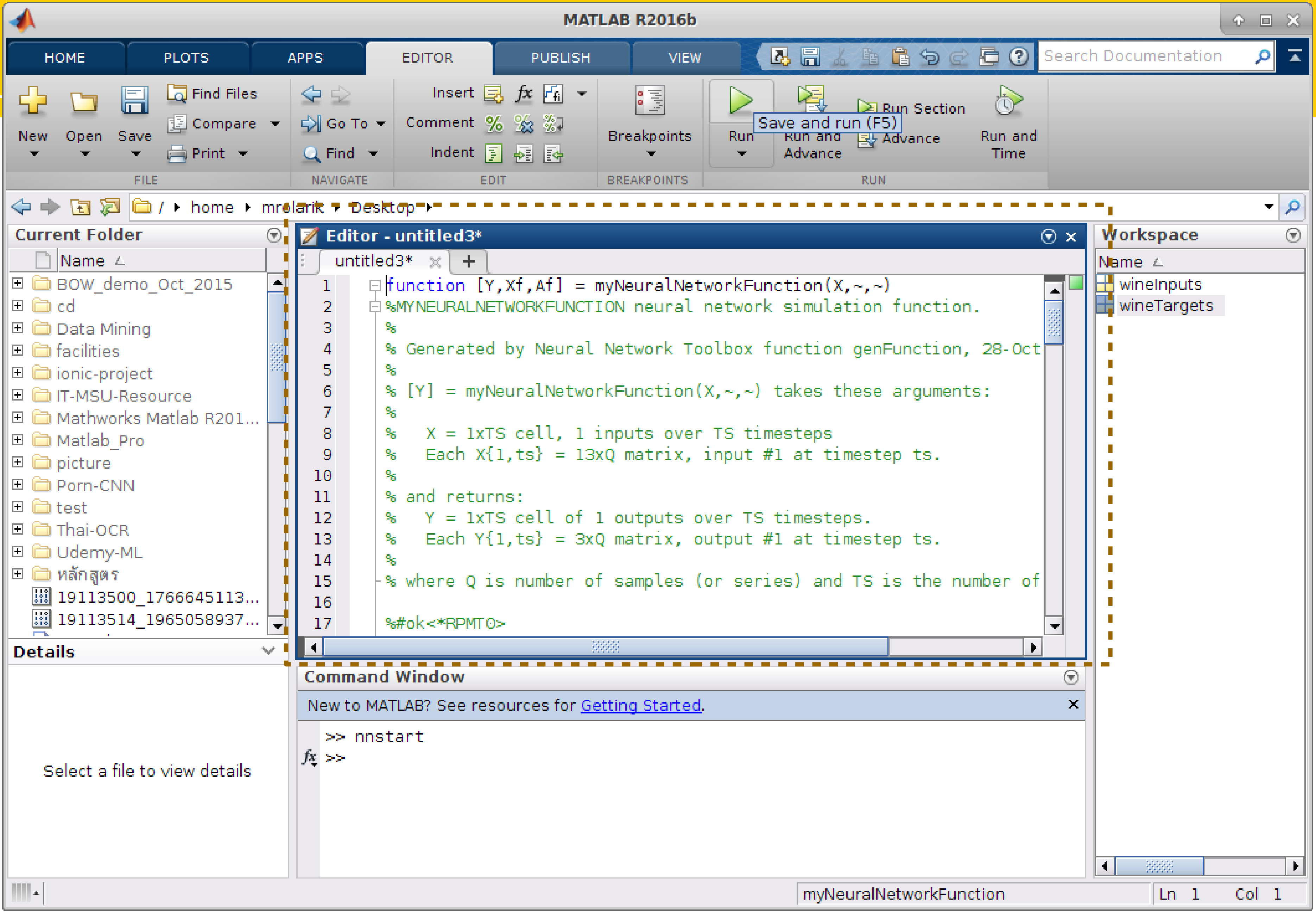
Simulate neural network in Simulink or deploy with Simulink Coder tools.

Generate a Simulink diagram: (gensim) Simulink Diagram

Graphics

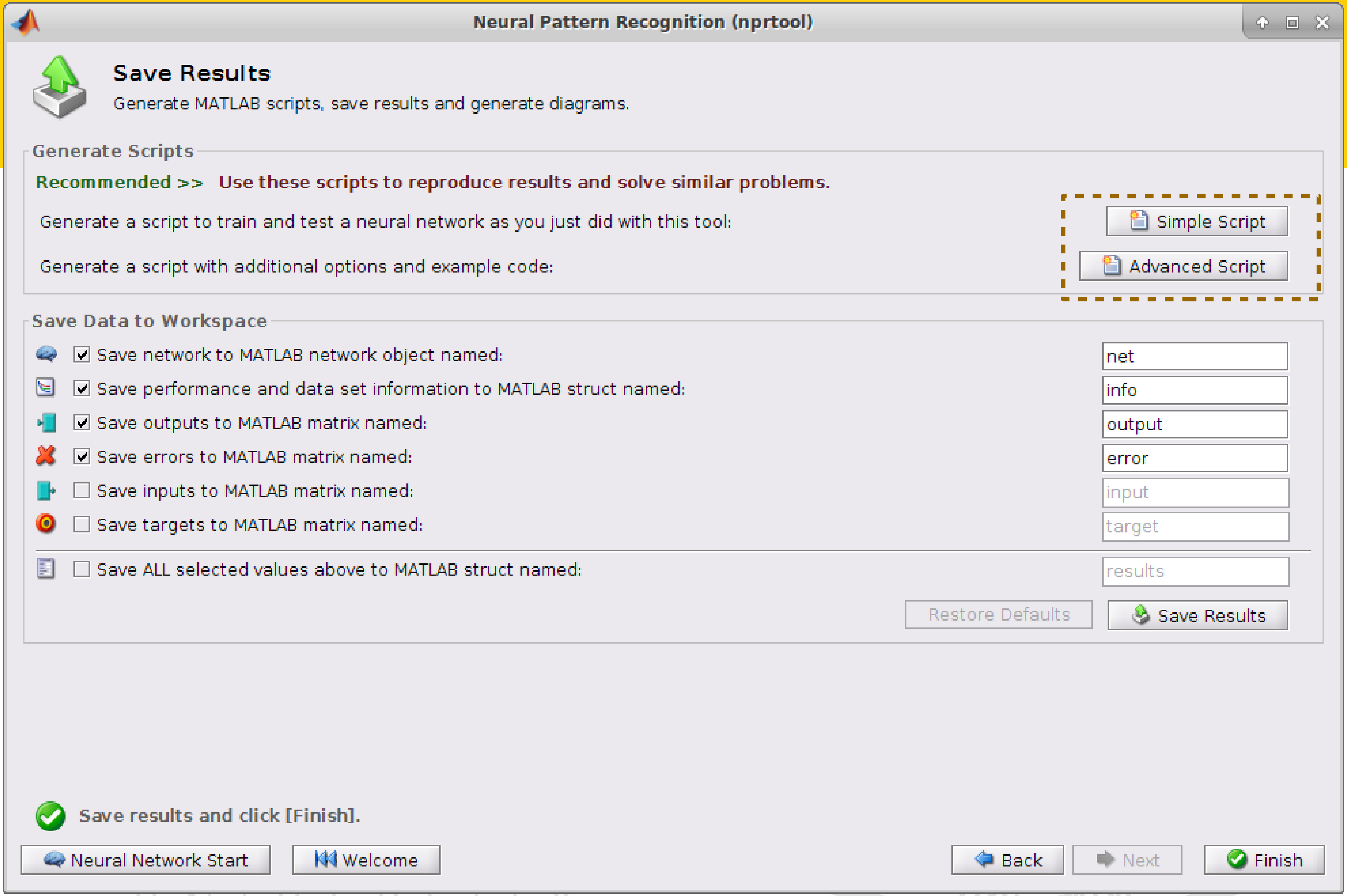
Generate a graphical diagram of the neural network: (network/view) Neural Network Diagram

Deploy a neural network or click [Next].



```
function [Y,Xf,Af] = myNeuralNetworkFunction(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 28-Oct
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = 13xQ matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
%   Each Y{1,ts} = 3xQ matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of

%#ok<*RPMT0>
```

```
>> nnstart
>>
```



Save Results

Generate MATLAB scripts, save results and generate diagrams.

Generate Scripts

Recommended >> **Use these scripts to reproduce results and solve similar problems.**

Generate a script to train and test a neural network as you just did with this tool:

Generate a script with additional options and example code:

Save Data to Workspace

☑ Save network to MATLAB network object named: net

☑ Save performance and data set information to MATLAB struct named: info

☑ Save outputs to MATLAB matrix named: output

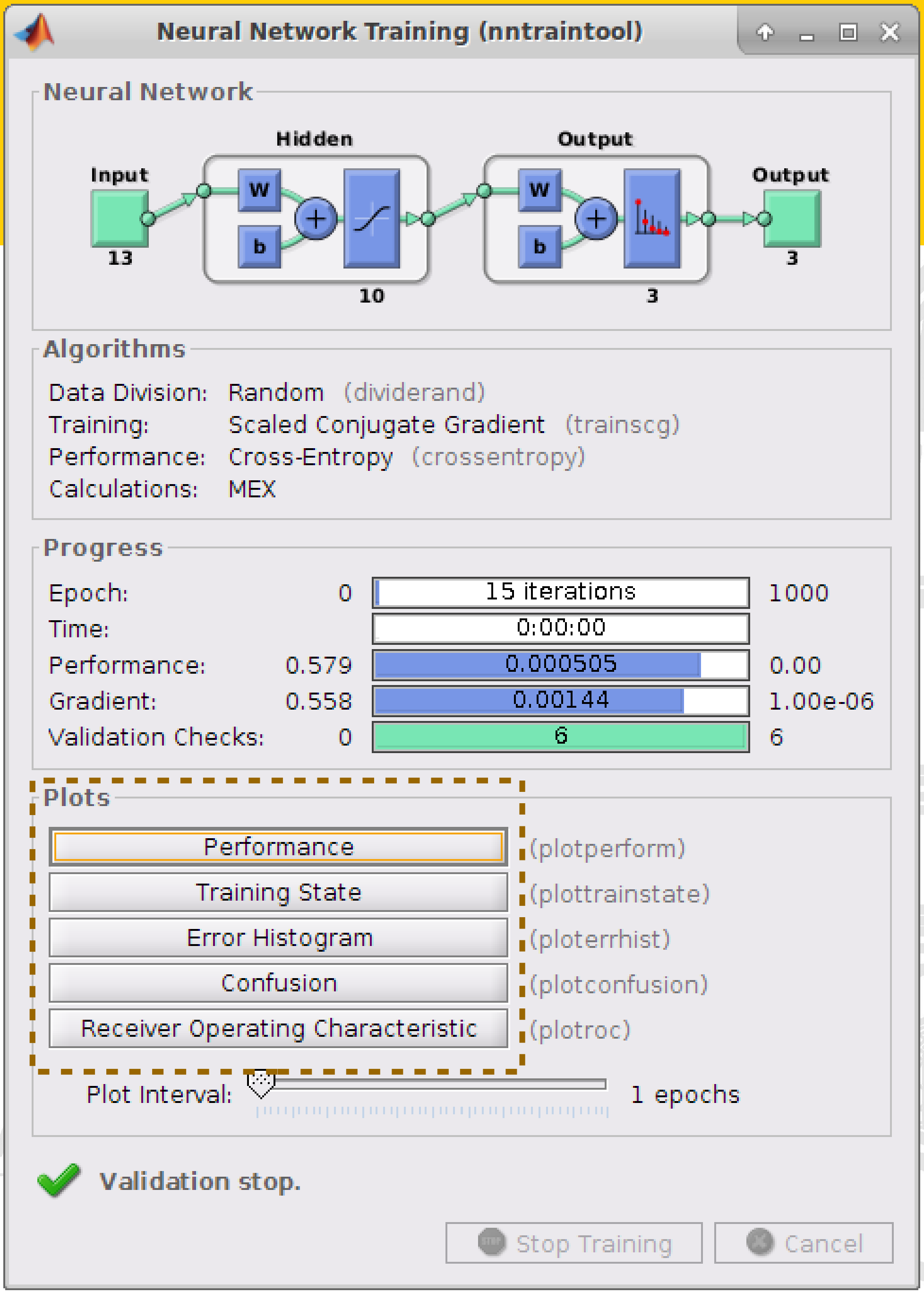
☑ Save errors to MATLAB matrix named: error

□ Save inputs to MATLAB matrix named: input

□ Save targets to MATLAB matrix named: target

□ Save ALL selected values above to MATLAB struct named: results

Save results and click [Finish].


Hidden
Output
Input
13
W
b
10
W
b
3
Output
3

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 | 15 iterations | 1000
Time: | 0:00:00 |
Performance: 0.579 | 0.000505 | 0.00
Gradient: 0.558 | 0.00144 | 1.00e-06
Validation Checks: 0 | 6 | 6

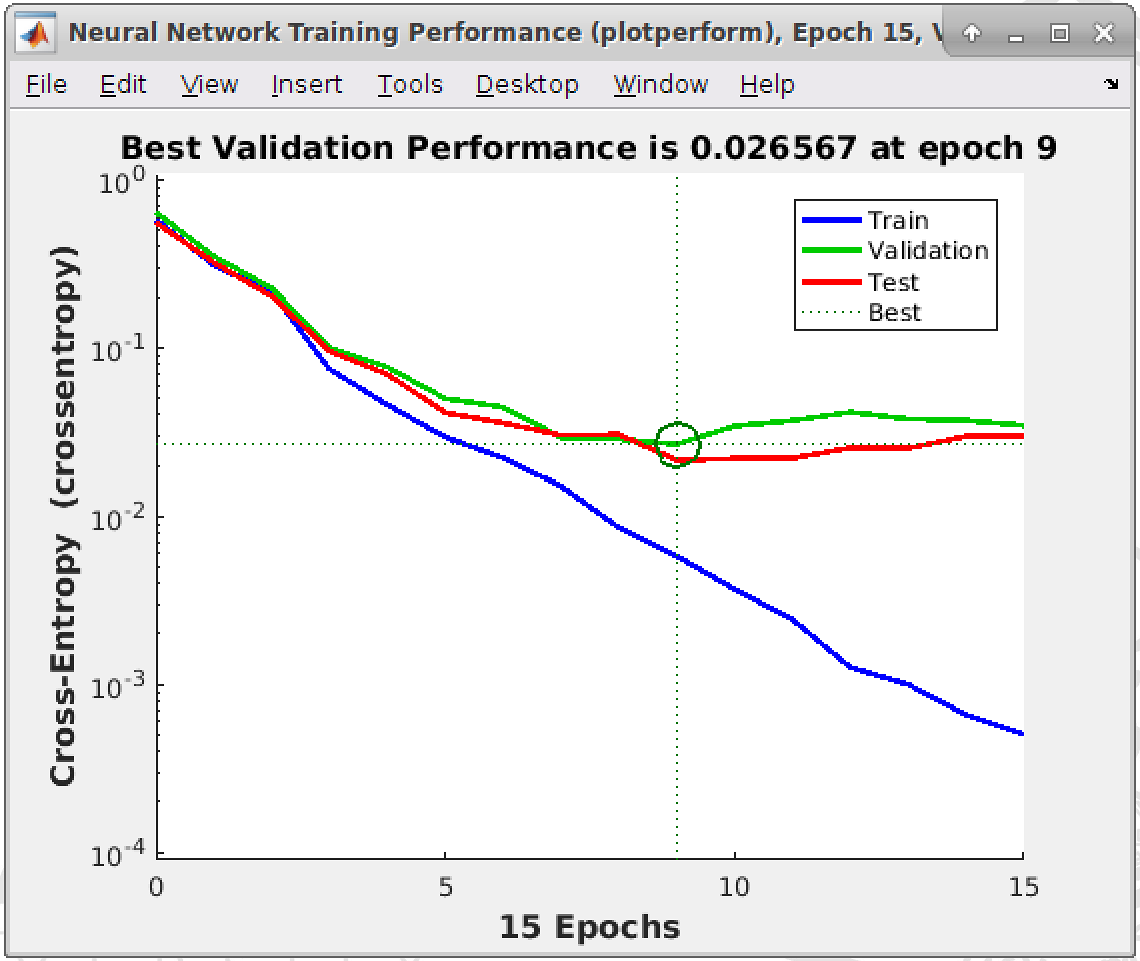
Plots

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

Plot Interval: 1 epochs

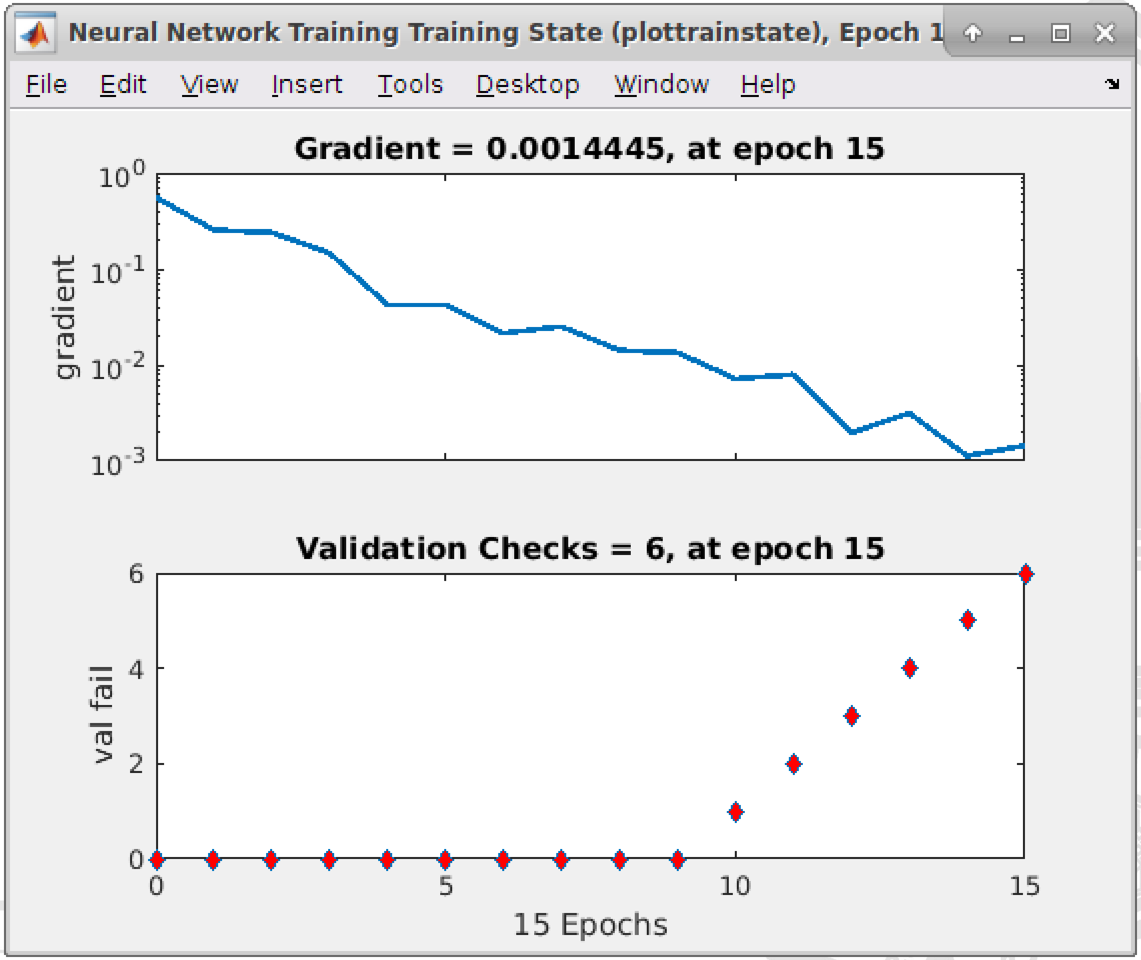
✓ Validation stop.

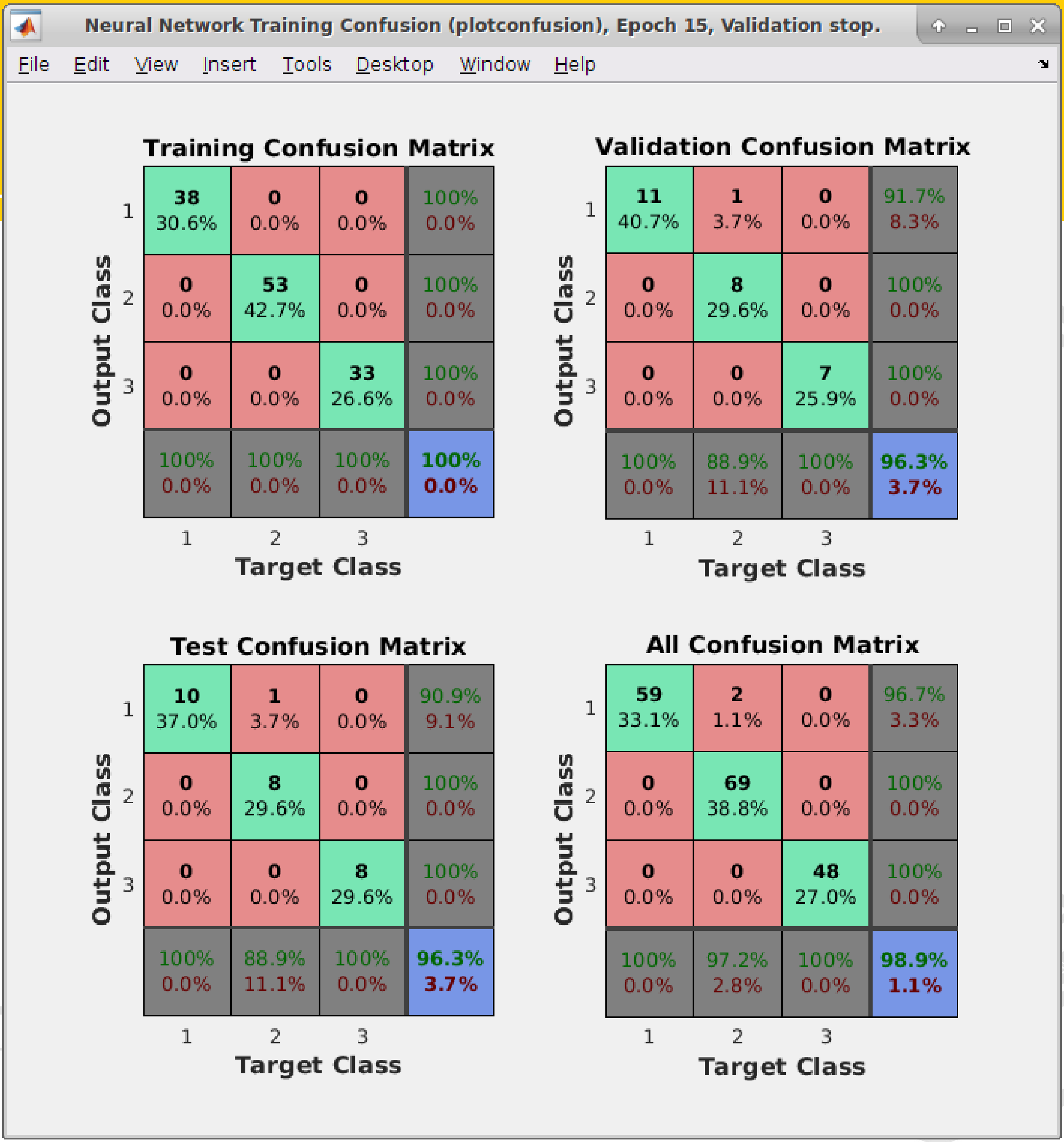
performance



Neural Network Training Performance (plotperform), Epoch 15, V
File
Edit
View
Insert
Tools
Desktop
Window
Help
Best Validation Performance is 0.026567 at epoch 9
Cross-Entropy (crossentropy)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
5
10
15
15 Epochs
Train
Validation
Test
Best

Training state


Neural Network Training Training State (plottrainstate), Epoch 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Gradient = 0.0014445, at epoch 15
gradient
10^{0}
10^{-1}
10^{-2}
10^{-3}
Validation Checks = 6, at epoch 15
val fail
6
4
2
0
0
5
10
15
15 Epochs


Neural Network Training Confusion (plotconfusion), Epoch 15, Validation stop.
File
Edit
View
Insert
Tools
Desktop
Window
Help

Training Confusion Matrix

Output Class

	1	2	3	
1	38 30.6%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	53 42.7%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	33 26.6%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%

Target Class

Validation Confusion Matrix

Output Class

	1	2	3	
1	11 40.7%	1 3.7%	0 0.0%	91.7% 8.3%
2	0 0.0%	8 29.6%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	7 25.9%	100% 0.0%
	100% 0.0%	88.9% 11.1%	100% 0.0%	96.3% 3.7%

Target Class

Test Confusion Matrix

Output Class

	1	2	3	
1	10 37.0%	1 3.7%	0 0.0%	90.9% 9.1%
2	0 0.0%	8 29.6%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	8 29.6%	100% 0.0%
	100% 0.0%	88.9% 11.1%	100% 0.0%	96.3% 3.7%

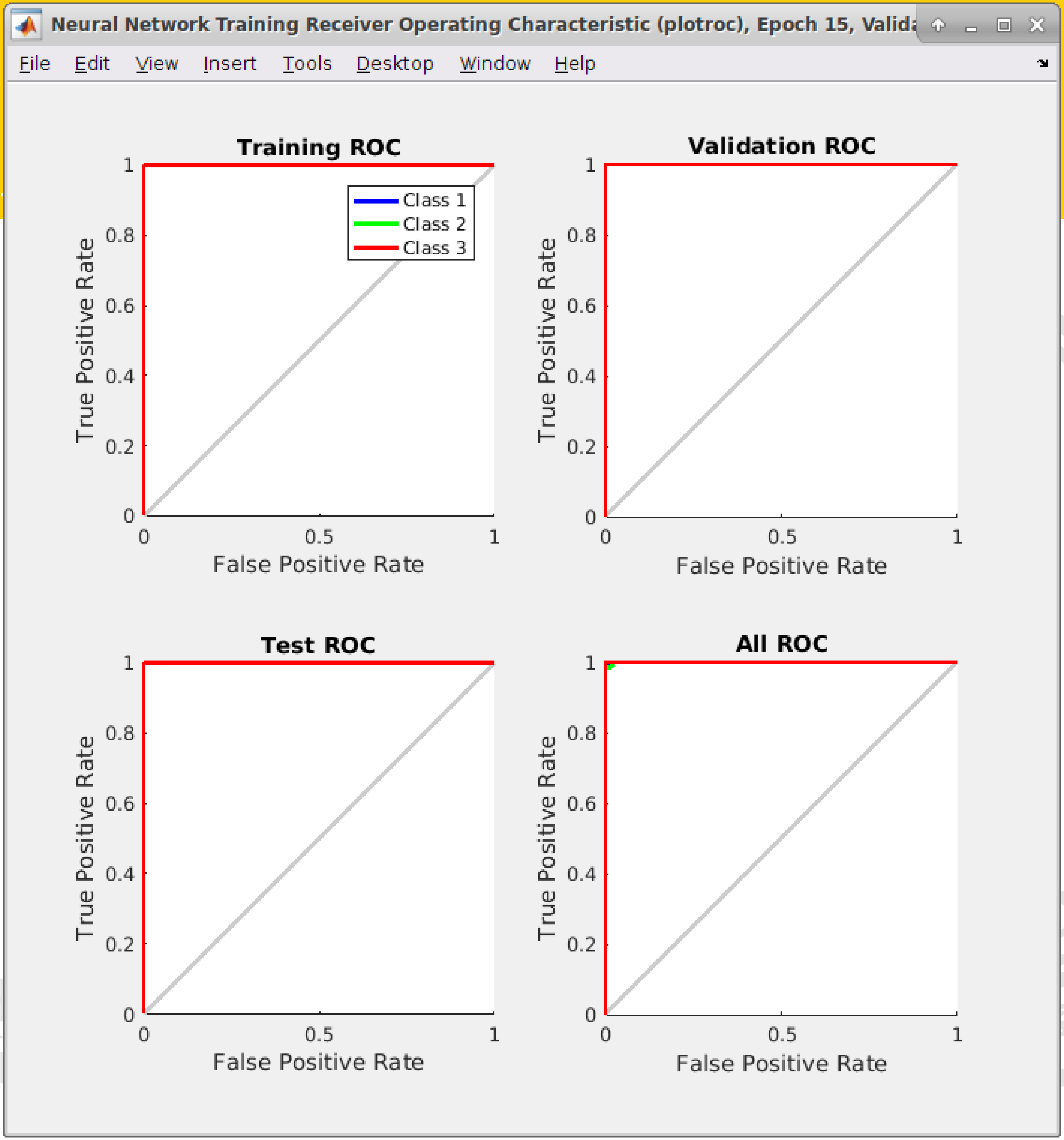
Target Class

All Confusion Matrix

Output Class

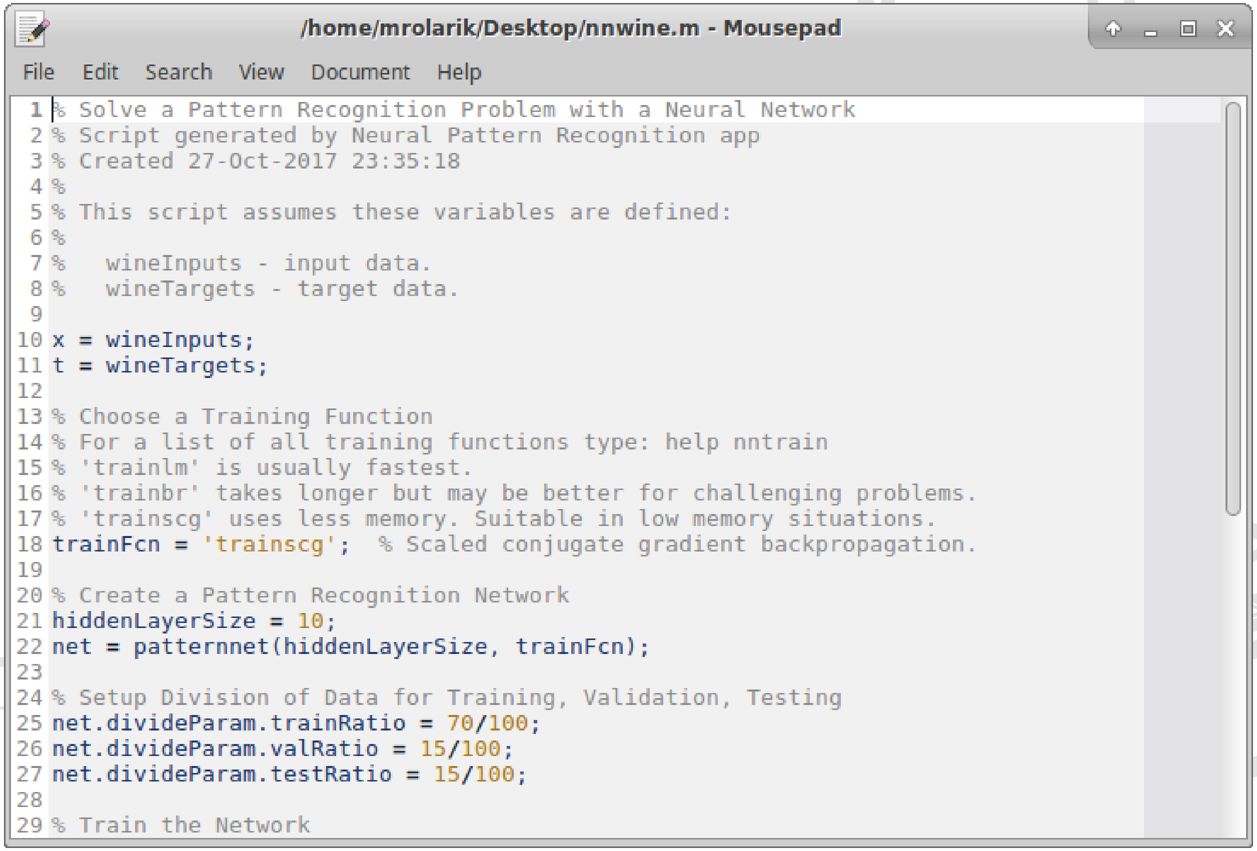
	1	2	3	
1	59 33.1%	2 1.1%	0 0.0%	96.7% 3.3%
2	0 0.0%	69 38.8%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	48 27.0%	100% 0.0%
	100% 0.0%	97.2% 2.8%	100% 0.0%	98.9% 1.1%

Target Class


Neural Network Training Receiver Operating Characteristic (plotroc), Epoch 15, Valid
File
Edit
View
Insert
Tools
Desktop
Window
Help
Training ROC
Class 1
Class 2
Class 3
Validation ROC
Test ROC
All ROC
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1
0.5

Simple script that generate by MATLAB

- MATLAB file extension is **.m**
- Save this script as **nnwine.m**

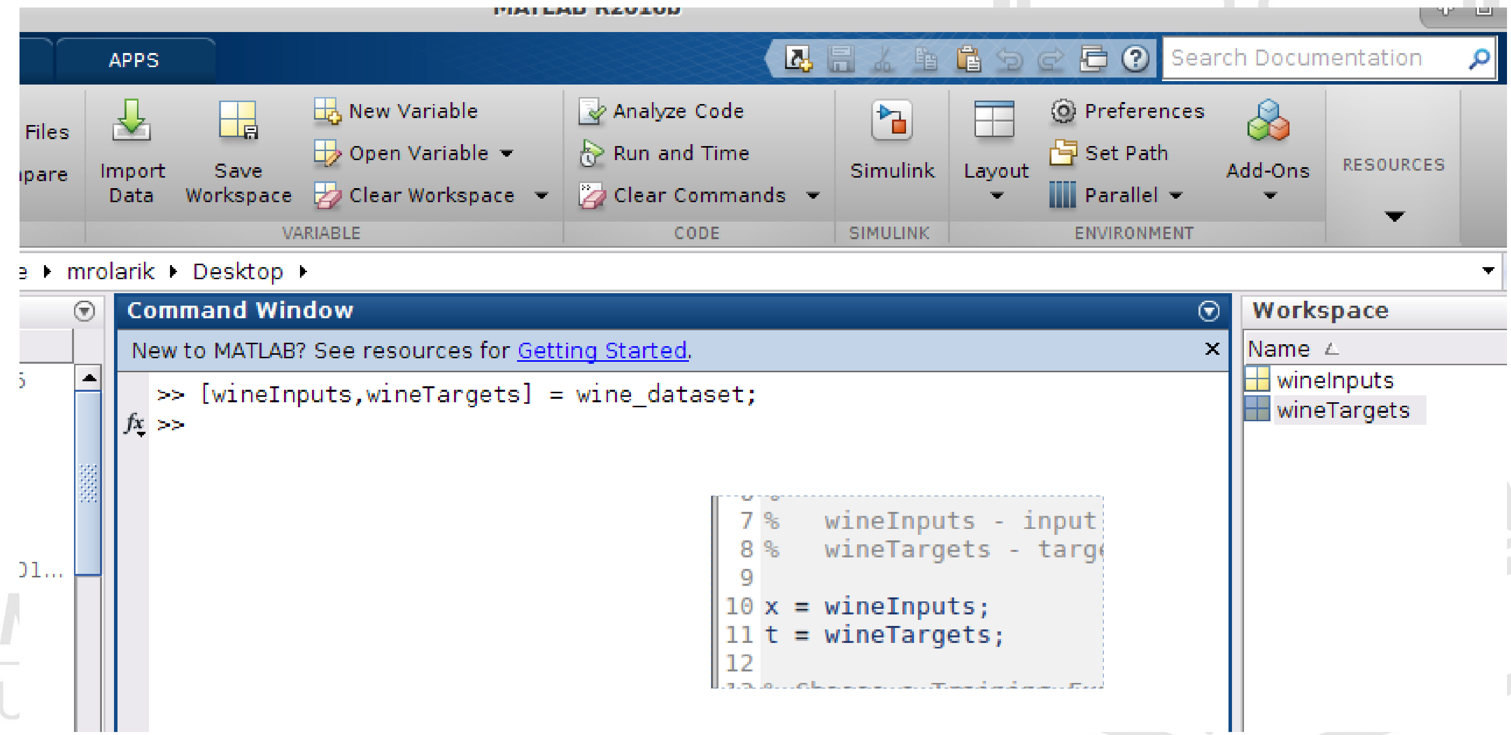


```
% Solve a Pattern Recognition Problem with a Neural Network
% Script generated by Neural Pattern Recognition app
% Created 27-Oct-2017 23:35:18
%
% This script assumes these variables are defined:
%
%   wineInputs - input data.
%   wineTargets - target data.

x = wineInputs;
t = wineTargets;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainscg';  % Scaled conjugate gradient backpropagation.

% Create a Pattern Recognition Network
hiddenLayerSize = 10;
net = patternnet(hiddenLayerSize, trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
```

commands

# load wine dataset

>> [wineInputs,wineTargets] = wine_dataset;



New to MATLAB? See resources for Getting Started.

```
>> [wineInputs,wineTargets] = wine_dataset;
>>
```

```
%   wineInputs - input
%   wineTargets - targe

x = wineInputs;
t = wineTargets;

```

Run your script

```
# load wine dataset
>> [wineInputs,wineTargets] = wine_dataset;
# run your script
>> nnwine
```