Lab-3 Object Characterization and Classification using Artificial Neural Network

Resources required: MATLAB: Neural Network Toolbox

Each student has to carry out the experiment of Object Characterization and recognition using Back-propagation Neural Network. There are two methods available in MATLAB: Neural Network Toolbox, the *nftool*, *nntool* and *command-line*. However, *nftool* is specialized in fitting problem only. Therefore, it is not covered in this lab assignment. Use the *nntool* GUI or *command-lines* for design, training and simulation of the back-propagation network.

Objective: To train the Neural network for Lab-3 (without noise (1 sample) and with noise (4 samples)) and demonstrate how the network identifies the learned object when the Boolean values of a character are input to the back-propagation network. Test the network using object with noise and without noise.

Use at least FOUR of the TRAINBFG, TRAINBFGC, TRAINBR, TRAINC, TRAINCGB, TRAINCGF, TRAINCGP, TRAINGD, TRAINGDA, TRAINGDM, TRAINGDX, TRAINLM, TRAINOSS, TRAINR, TRAINRP, TRAINSCG algorithms for training the network and **give a comparative study on the performance**. Fix the following parameters and **comment** on the training time and errors:

- Epochs (train until reach termination conditions)
- Number of Hidden Layers (do not exceed 2)
- Number of Neurons in Each Hidden Layer (do not exceed 40 neuron)
- Transfer Function of Each Layer (your choice)

Laboratory Activities:

Week 7 Session 1: Demonstration on “How to use Neural Network Toolbox” at the beginning of the class by tutor. Practice with NN Toolbox. Start working on the lab assignment. **Read carefully the sample demonstration as given in page-2 to 17**

Week 8 Session 2: Continue working on the lab assignment.

Week 9 Session 1: Wednesday 5.00pm

Report Format

You are required to prepare a formal report for this lab assignment. Please make sure that your name and Student ID is included in the first page of your softcopy report. Your report will be assessed based on the demonstration shown and how the work done is documented. However, the following aspects are essential:

1. Details of object characterization you designed (with noise and without noise).
2. I/O preparation for training.
3. The network architecture of the working network.
4. Training parameters of the working network.
5. Comparative study on the performances of different training algorithms
6. Maximum page of the report is limited to 5. Rest of the materials can be added as an appendix and submit as a single file (.pdf)

Steps of Object Characterization, Classification and recognition using Artificial Neural Network.

Neural networks have a vast range of applications including predicting/determining the most probable result given a set of data. This exercise will show the steps required to set up a basic neural network and the relevant dataset to be tested.

The required set of image objects are identified and retrieved on the basis of object recognition. To preprocess the input natural images follow the procedures as given in steps below. Quantize the numerous regions and extract the configured set of features from the image using a simple procedures as stated below. Finally to classify the natural images in an efficient manner using neural network classifier.

The dataset used is the Iris Flower Dataset. This dataset is a typical test case for machine learning techniques including neural network. The dataset consists of various parameters for three different Iris types, namely ***iris – virginica, iris – setosa and iris - versicolor***. These parameters include sepal width and length, and petal width and length. It contains 4 attributes (sepal length and width in centimetres, and petal length and width in centimentres) of 150 observations of Iris flowers split equally among 3 classes: the Iris Virginica, Iris Setosa and Iris Versicolour.

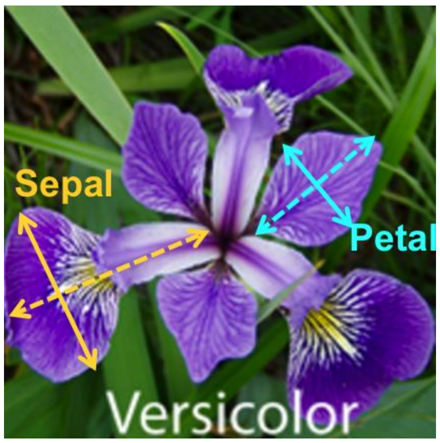


Figure 1: Iris Flower Example

Required Software

1. MATLAB
2. Microsoft Excel

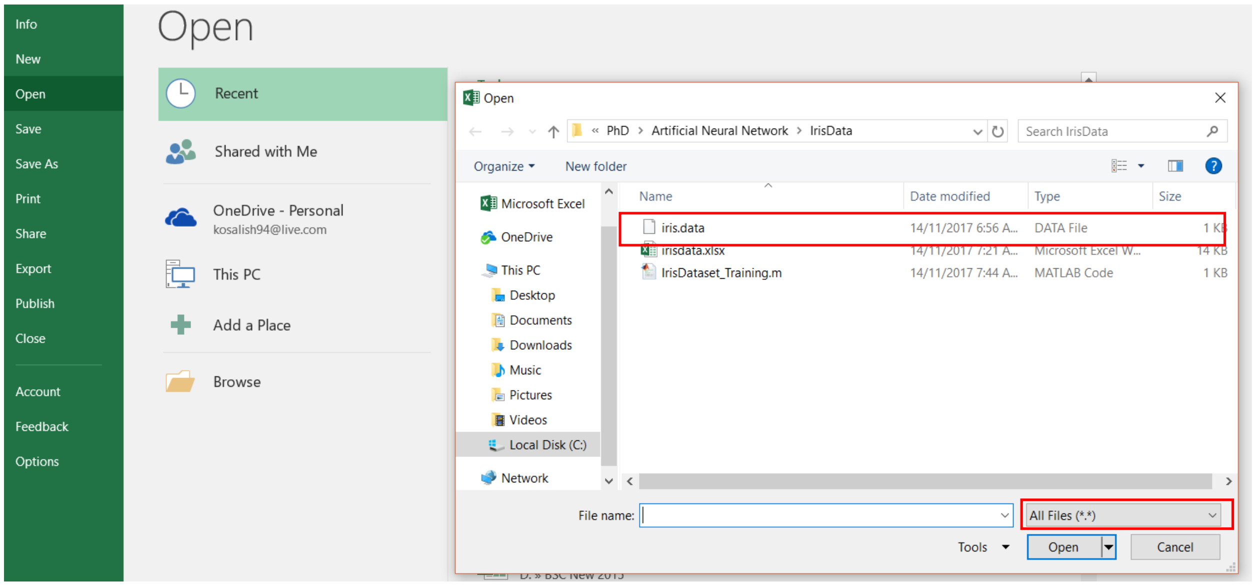
Procedures

PART 1: Download and Import Dataset to Microsoft Excel

The Iris Flower dataset can be obtained from various sources. However, to allow for a smooth experience, use the link below to download the dataset and save it in the working directory of MATLAB. (Example C:\Documents\Matlab\NeuralNetwork)

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

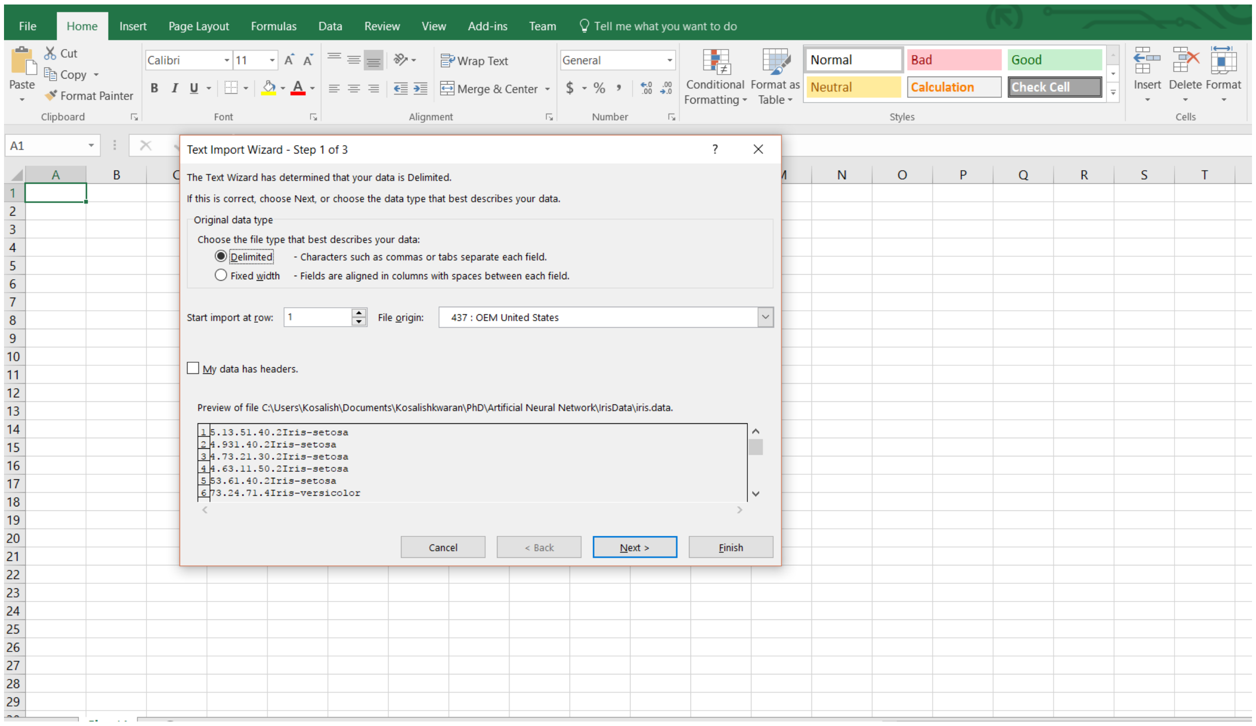
Open Microsoft Excel and the downloaded file by using the Open menu from Excel. Make sure to select All Files in dropdown menu to display the downloaded file as well.



Name	Date modified	Type	Size
iris.data	14/11/2017 6:56 A...	DATA File	1 KB
irisdata.xlsx	14/11/2017 7:21 A...	Microsoft Excel W...	14 KB
IrisDataset_Training.m	14/11/2017 7:44 A...	MATLAB Code	1 KB

All Files (*.*)

Figure 2: MS Excel File Open



Text Import Wizard - Step 1 of 3

The Text Wizard has determined that your data is Delimited.

If this is correct, choose Next, or choose the data type that best describes your data.

Original data type

Choose the file type that best describes your data:

[✓ Delimited] - Characters such as commas or tabs separate each field.

Fixed width - Fields are aligned in columns with spaces between each field.

Start import at row: 1 File origin: 437 : OEM United States

My data has headers.

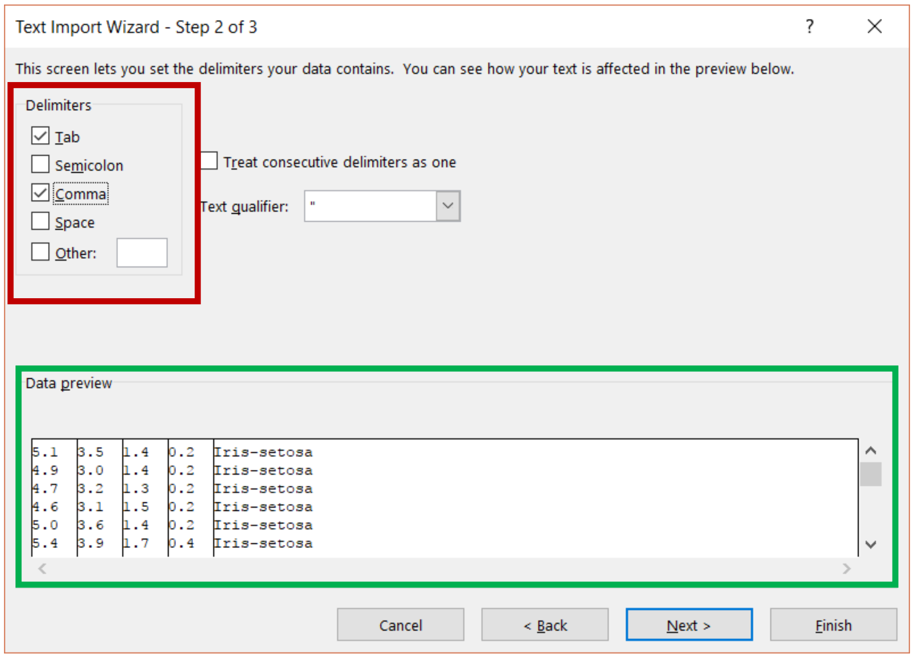
Preview of file C:\Users\Kosalish\Documents\Kosalishkwaran\PhD\Artificial Neural Network\IrisData\iris.data.

```
1 5.13.51.40.2Iris-setosa
2 4.931.40.2Iris-setosa
3 4.73.21.30.2Iris-setosa
4 4.63.11.50.2Iris-setosa
5 53.61.40.2Iris-setosa
6 73.24.71.4Iris-versicolor
```

Cancel | < Back | Next > | Finish

Figure 3: MS Excel Text Import Wizard

Click on Next one time. In the next window, select the tab and comma options. You will be able to see the resulting data import in Excel. It should look exactly like the picture below.


Text Import Wizard - Step 2 of 3

This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.

Delimiters

- [✓] Tab
- [ ] Semicolon
- [✓] Comma
- [ ] Space
- [ ] Other:

[ ] Treat consecutive delimiters as one

Text qualifier: "

Data preview

```
5.1  3.5  1.4  0.2  Iris-setosa
4.9  3.0  1.4  0.2  Iris-setosa
4.7  3.2  1.3  0.2  Iris-setosa
4.6  3.1  1.5  0.2  Iris-setosa
5.0  3.6  1.4  0.2  Iris-setosa
5.4  3.9  1.7  0.4  Iris-setosa
```

Figure 4: Text Import Wizard Options

Click Next once again and finally click Finish to close the text import wizard. Your data should now be arranged in cells accordingly.

PART 2: Organize Data in Microsoft Excel

First, it is wise to understand what each column represents. The default arrangement is as follows.

Table 1: Data Representation in MS Excel

Column	Data
A	Sepal Length
B	Sepal Width
C	Petal Length
D	Petal Width
E	Type/Species of Flower

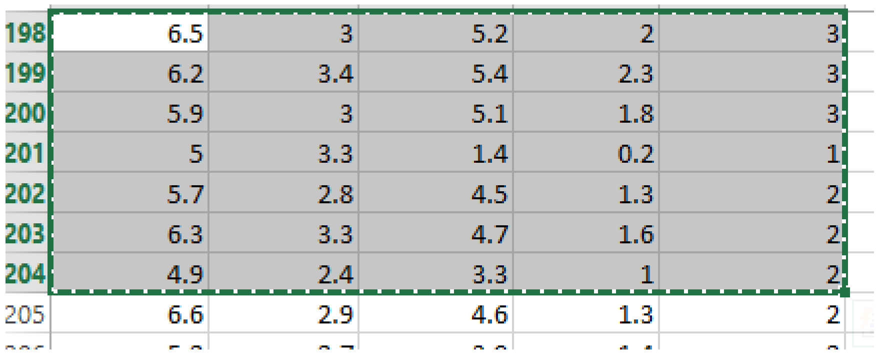
You may insert these headings into the actual table if you prefer, however, for this example, the table will remain unedited for simplicity. Similarly, you may adjust the column widths should you prefer a more neatly arranged table.

Create a new sheet in the current Excel file and rename it to **test**. This sheet will be used to check the network's efficiency later in the exercise.

Next, in the original sheet, replace the type of Flower column (Col E) with numbers representing the flowers. Put **1** in place of ***iris-setosa***, **2** in place of ***iris-versicolor*** and **3** for ***iris-virginica***.

Before continuing, we need to add noise to the current dataset. Noise can be added in a variety of ways. This exercise will show one of those suggested methods as follows:

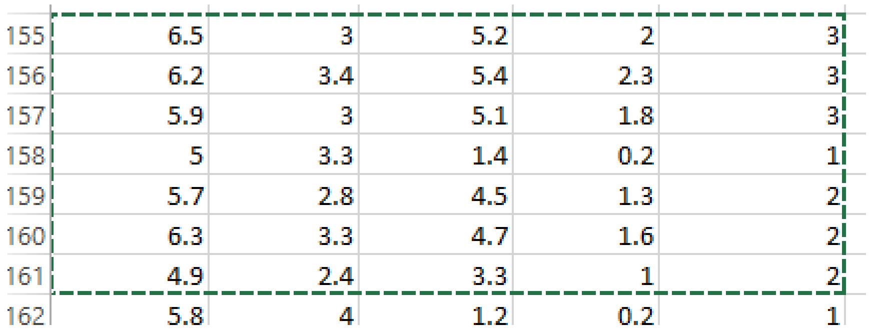
1. Copy and paste the current dataset (150 data) multiple times below each other.
2. Once a satisfactory number of data reached (>200), **CUT random rows** from each flower type and insert the rows randomly into the current dataset.



198	6.5	3	5.2	2	3
199	6.2	3.4	5.4	2.3	3
200	5.9	3	5.1	1.8	3
201	5	3.3	1.4	0.2	1
202	5.7	2.8	4.5	1.3	2
203	6.3	3.3	4.7	1.6	2
204	4.9	2.4	3.3	1	2
205	6.6	2.9	4.6	1.3	2

Figure 5: Select Data as shown (Row 198 – Row 204)

3. Select **Insert Cut Row** from Excel command by right clicking at desired row.



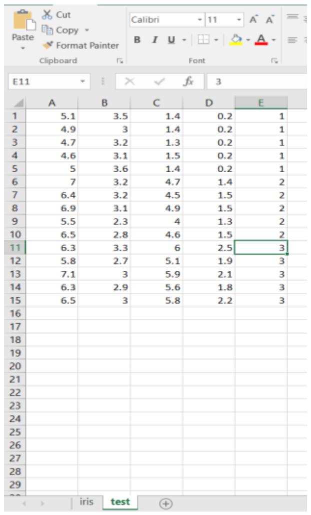
155	6.5	3	5.2	2	3
156	6.2	3.4	5.4	2.3	3
157	5.9	3	5.1	1.8	3
158	5	3.3	1.4	0.2	1
159	5.7	2.8	4.5	1.3	2
160	6.3	3.3	4.7	1.6	2
161	4.9	2.4	3.3	1	2
162	5.8	4	1.2	0.2	1

Figure 6: Final Result after Modification (Row 155 - Row 161)

Adding noise can also be done by introducing new data to the dataset (ie. Adding new measurements for the three flowers). Noise is added to ensure that the network is trained properly.

To prepare the test data (for testing the network), cut data from 5 to 10 rows (randomly chosen) of each **category** of flowers. i.e. 5 from '1', 5 from '2' and 5 from '3'. Paste the data into the sheet named **test**. The final data in **test** sheet should look something like the figure below.

Save the file as an Excel format file (.xls, .xlsx)



	A	B	C	D	E
1	5.1	3.5	1.4	0.2	1
2	4.9	3	1.4	0.2	1
3	4.7	3.2	1.3	0.2	1
4	4.6	3.1	1.5	0.2	1
5	5	3.6	1.4	0.2	1
6	7	3.2	4.7	1.4	2
7	6.4	3.2	4.5	1.5	2
8	6.9	3.1	4.9	1.5	2
9	5.5	2.3	4	1.3	2
10	6.5	2.8	4.6	1.5	2
11	6.3	3.3	6	2.5	3
12	5.8	2.7	5.1	1.9	3
13	7.1	3	5.9	2.1	3
14	6.3	2.9	5.6	1.8	3
15	6.5	3	5.8	2.2	3

Figure 7: Test sheet data

PART 3: MATLAB Commands

The following sections involve using MATLAB commands to perform network training and testing. The codes will be explained in detail.

Initialize Workspace

```
clc;clear all;close all;
```

The first batch of codes initialize the workspace before performing any operations. These include clearing the command window and variables. **DO NOT** include these commands in your scripts (to prevent accidental deletion of trained neural networks). Instead, run them from the command window.

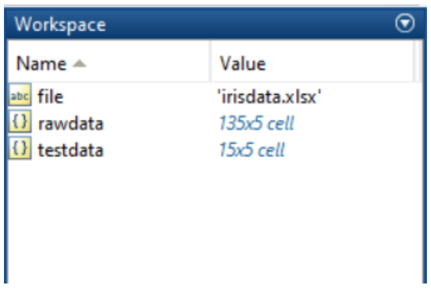
Read Data from Excel file

```
file = 'irisdata.xlsx';
[~,~,rawdata] = xlsread(file,1); % Read from first spreadsheet
[~,~,testdata] = xlsread(file,2);
input = cell2mat(rawdata(:,1:4));
output = cell2mat(rawdata(:,5));
test = cell2mat(testdata(:,1:4));
test_out = cell2mat(testdata(:,5));
```

Assign a variable to the Excel file being worked on. Ensure that the Excel file is in the same directory as the script.

```
>>> [~,~,rawdata] = xlsread(file,1);
>>> [~,~,testdata] = xlsread(file,2);
```

Use the ***xlsread*** built – in function of MATLAB to import the xls file and separate the variables. Notice that the first sheet containing the raw dataset is placed in a *rawdata* array, whereas the test sheet is placed in *testdata* array.



Workspace

Name	Value
file	'irisdata.xlsx'
rawdata	*135x5 cell*
testdata	*15x5 cell*

Notice that rawdata and testdata are in “cell” format. These need to be converted into arrays. Hence, use the **cell2mat** function to perform conversion.

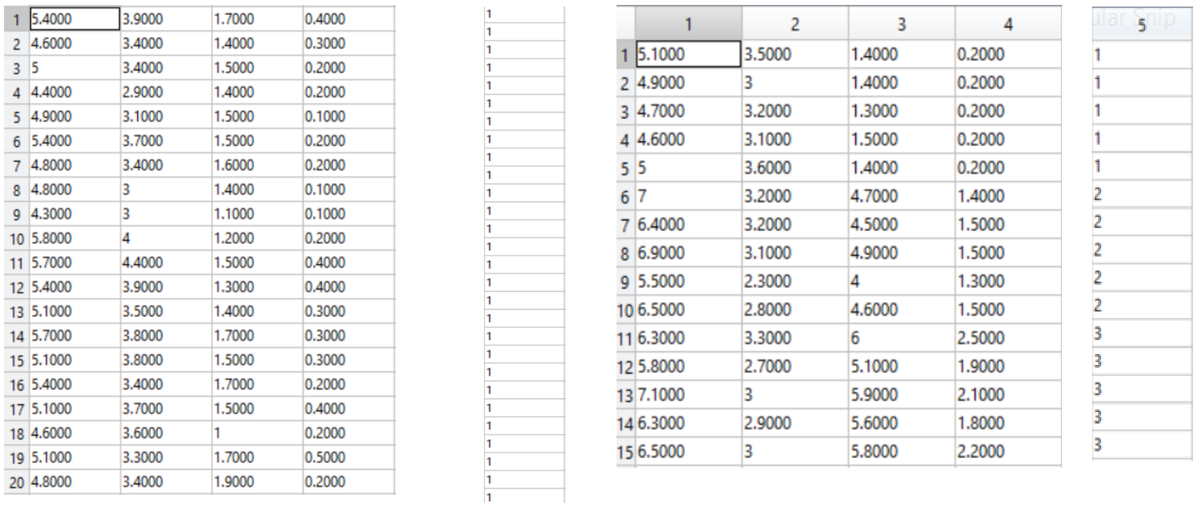
```
>>> input = cell2mat(rawdata(:,1:4));
>>> output = cell2mat(rawdata(:,5));
>>> test = cell2mat(testdata(:,1:4));

>>> test_out = cell2mat(testdata(:,5));
```

The data is separated into 4 variables, **input, output, test, test_out.** You can use whatever variable name that you prefer, as long as you are consistent with naming the variable.

The **test_out** variable contains the output of the test data. This output is the type/species of Iris flowers.

Input contains the sepal width and length, petal width and length. Output variable contains the type of iris and test contains the array of data previously set aside to test the efficiency of the trained network.

You may check the imported data and compare it with the data in Excel file if you wish to affirm the data imported. The figures below show data for the **Input, Output, test, test_out** variables respectively.



1	5.4000	3.9000	1.7000	0.4000
2	4.6000	3.4000	1.4000	0.3000
3	5	3.4000	1.5000	0.2000
4	4.4000	2.9000	1.4000	0.2000
5	4.9000	3.1000	1.5000	0.1000
6	5.4000	3.7000	1.5000	0.2000
7	4.8000	3.4000	1.6000	0.2000
8	4.8000	3	1.4000	0.1000
9	4.3000	3	1.1000	0.1000
10	5.8000	4	1.2000	0.2000
11	5.7000	4.4000	1.5000	0.4000
12	5.4000	3.9000	1.3000	0.4000
13	5.1000	3.5000	1.4000	0.3000
14	5.7000	3.8000	1.7000	0.3000
15	5.1000	3.8000	1.5000	0.3000
16	5.4000	3.4000	1.7000	0.2000
17	5.1000	3.7000	1.5000	0.4000
18	4.6000	3.6000	1	0.2000
19	5.1000	3.3000	1.7000	0.5000
20	4.8000	3.4000	1.9000	0.2000

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

	1	2	3	4
1	5.1000	3.5000	1.4000	0.2000
2	4.9000	3	1.4000	0.2000
3	4.7000	3.2000	1.3000	0.2000
4	4.6000	3.1000	1.5000	0.2000
5	5	3.6000	1.4000	0.2000
6	7	3.2000	4.7000	1.4000
7	6.4000	3.2000	4.5000	1.5000
8	6.9000	3.1000	4.9000	1.5000
9	5.5000	2.3000	4	1.3000
10	6.5000	2.8000	4.6000	1.5000
11	6.3000	3.3000	6	2.5000
12	5.8000	2.7000	5.1000	1.9000
13	7.1000	3	5.9000	2.1000
14	6.3000	2.9000	5.6000	1.8000
15	6.5000	3	5.8000	2.2000

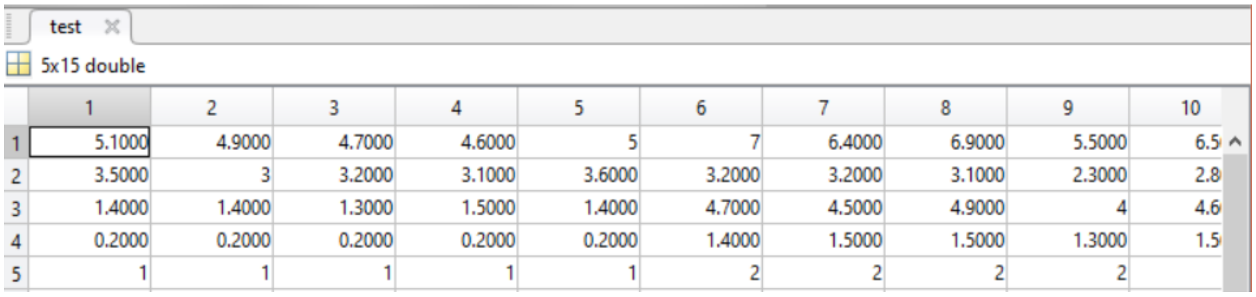
5
1
1
1
1
1
2
2
2
2
2
3
3
3
3
3

Figure 8: Data content for Variables

One final step is to transpose the matrices in order to work with MATLAB. The following commands show that. Transposing of matrices is required prior to working with the imported data in MATLAB.

```
% Change rows into columns (Transpose)
>>> input = input';
>>> output = output';
>>> test = test';
>>> test_out = test_out';
```

Transposing the matrices will cause the data to appear such as the figure shown below.



	1	2	3	4	5	6	7	8	9	10
1	5.1000	4.9000	4.7000	4.6000	5	7	6.4000	6.9000	5.5000	6.5
2	3.5000	3	3.2000	3.1000	3.6000	3.2000	3.2000	3.1000	2.3000	2.8
3	1.4000	1.4000	1.3000	1.5000	1.4000	4.7000	4.5000	4.9000	4	4.6
4	0.2000	0.2000	0.2000	0.2000	0.2000	1.4000	1.5000	1.5000	1.3000	1.5
5	1	1	1	1	1	2	2	2	2	

Figure 9: Transposed Data

Note that the dimensions of the matrices have changed (as it should).

Now the process of creating the neural network can begin.

<u>PART 4: Artificial Neural Network Toobox</u>

```
>>>nntool
```

Type the above command into the command line of MATLAB to bring up the NN Toolbox. The following window will appear.

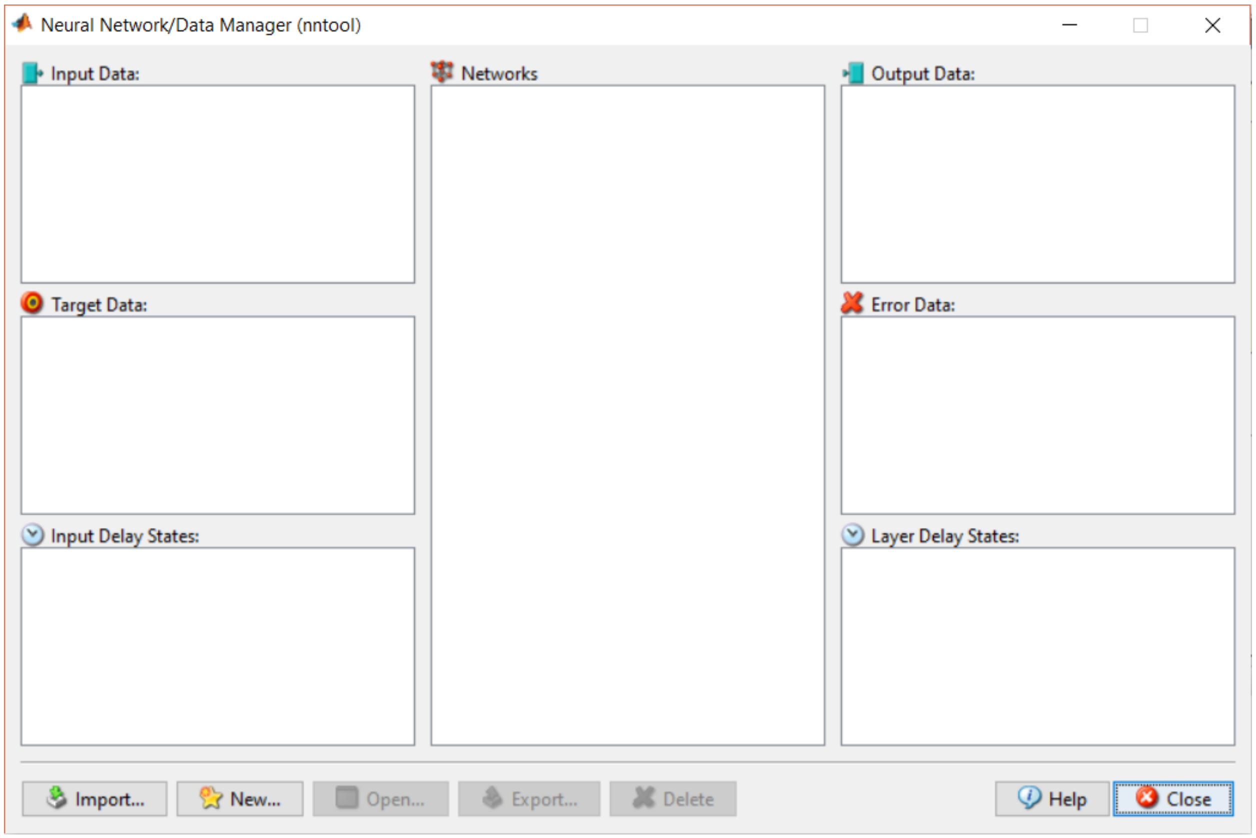
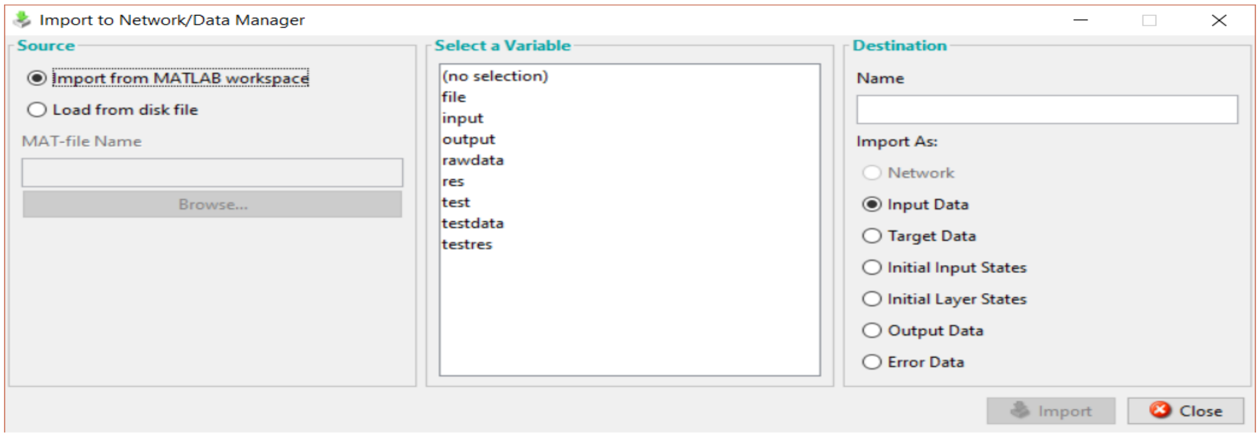


Figure 10: Neural Network Toolbox Main Window

First, the data need to be imported into the toolbox. To do this, select Import to open the import wizard. Ensure that the "Select from Workspace" option is chosen.



Source: [✓ Import from MATLAB workspace] Load from disk file

MAT-file Name

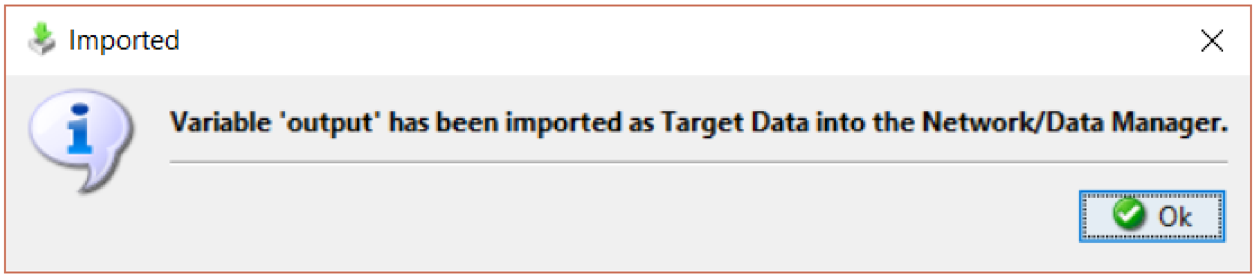
Select a Variable:
(no selection)
file
input
output
rawdata
res
test
testdata
testres

Name

Import As: Network [✓ Input Data] Target Data Initial Input States Initial Layer States Output Data Error Data

Figure 11: Data Import Manager (NN Toolbox)

The **input** and **output** data need to be imported. DO NOT import the **test** data. Leave it untouched for the time being. Import the **input** variable as **Input Data** and **output** as **Target Data**. Upon successful import of data, the following window will appear.


Imported

Variable 'output' has been imported as Target Data into the Network/Data Manager.

Figure 12: Completion Pop up

The toolbox window should look something similar to the figure below.

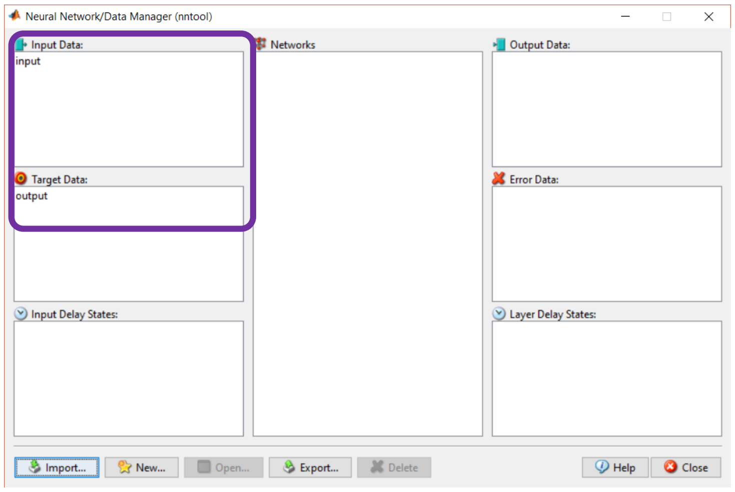
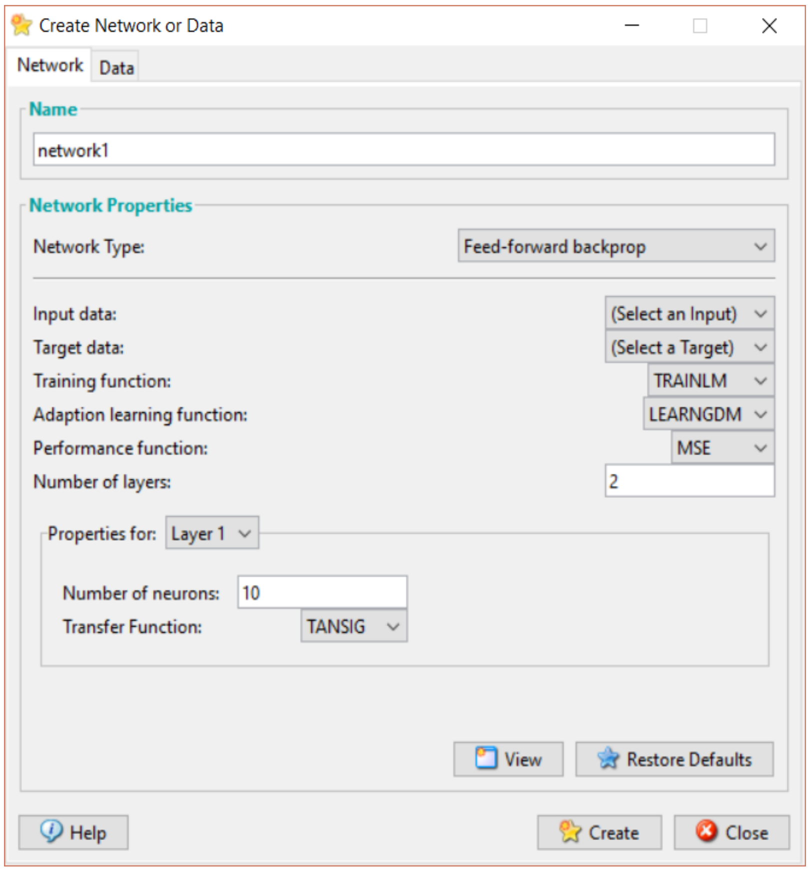


Figure 13: Main Window with Data Imported

Now, select "New" from the toolbox window to open the tool for creating the neural network.


Create Network or Data

Network | Data

Name

network1

Network Properties

Network Type: Feed-forward backprop

Input data: (Select an Input)
Target data: (Select a Target)
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

Figure 14: Network Creation Window (NN Toolbox)

This is the window where the basic parameters of the neural network can be modified. For this initial exercise, other parameters are left unchanged. "Input Data" and "Target Data" are selected based on the previous data import.

Rename the network to a suitable name. This is the name that will be used to test the network later. Select Create to create the network. Select the network in the main toolbox window and click Open.

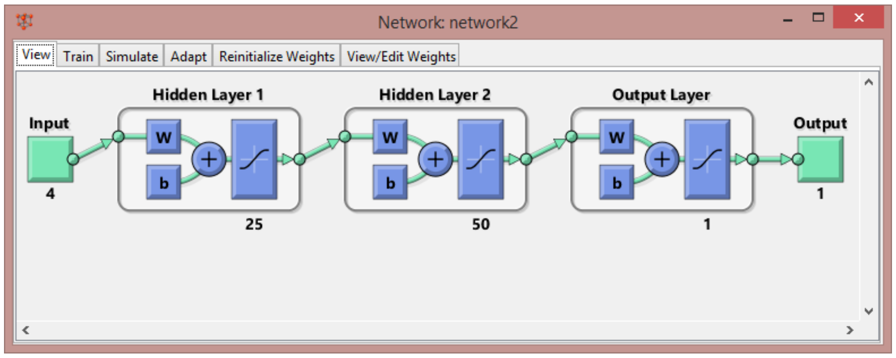
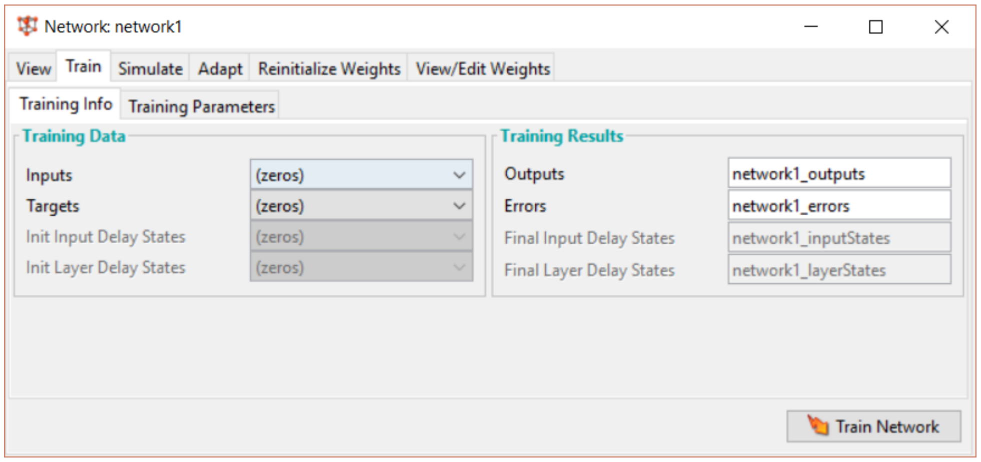


Figure 15: Network Architecture

The window above will open, showing the schematic of the created network. Notice the layers and neurons shown reflect the number of layers in the in the creation of the network previously.

Switch over to the “Train” tab. Select both the Input and Target data before clicking “Train Network”.



Training Data

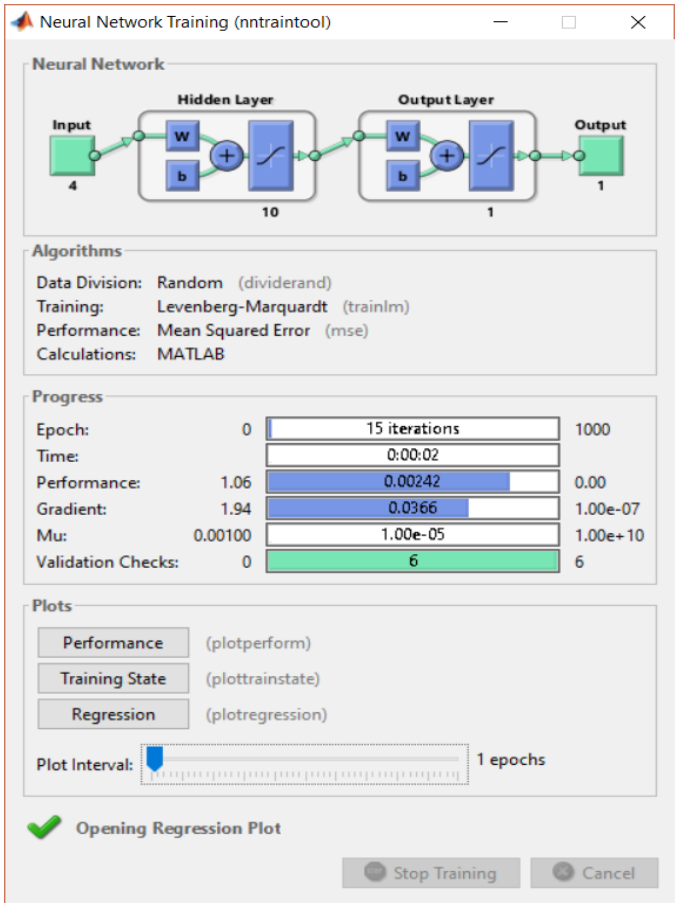
Inputs	(zeros)
Targets	(zeros)
Init Input Delay States	(zeros)
Init Layer Delay States	(zeros)

Training Results

Outputs	network1_outputs
Errors	network1_errors
Final Input Delay States	network1_inputStates
Final Layer Delay States	network1_layerStates

Figure 16: Network Training

This window will appear, showing the training progress.



Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MATLAB

Progress

Epoch:	0	15 iterations	1000
Time:		0:00:02	
Performance:	1.06	0.00242	0.00
Gradient:	1.94	0.0366	1.00e-07
Mu:	0.00100	1.00e-05	1.00e+10
Validation Checks:	0	6	6

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)
Plot Interval: 1 epochs

Opening Regression Plot

Figure 17: Network Training Window (NNToolbox)

Train the network (by clicking Train Network) five to six times and check the **Regression** in the window above. The regression line should be from lower left corner to upper right corner.

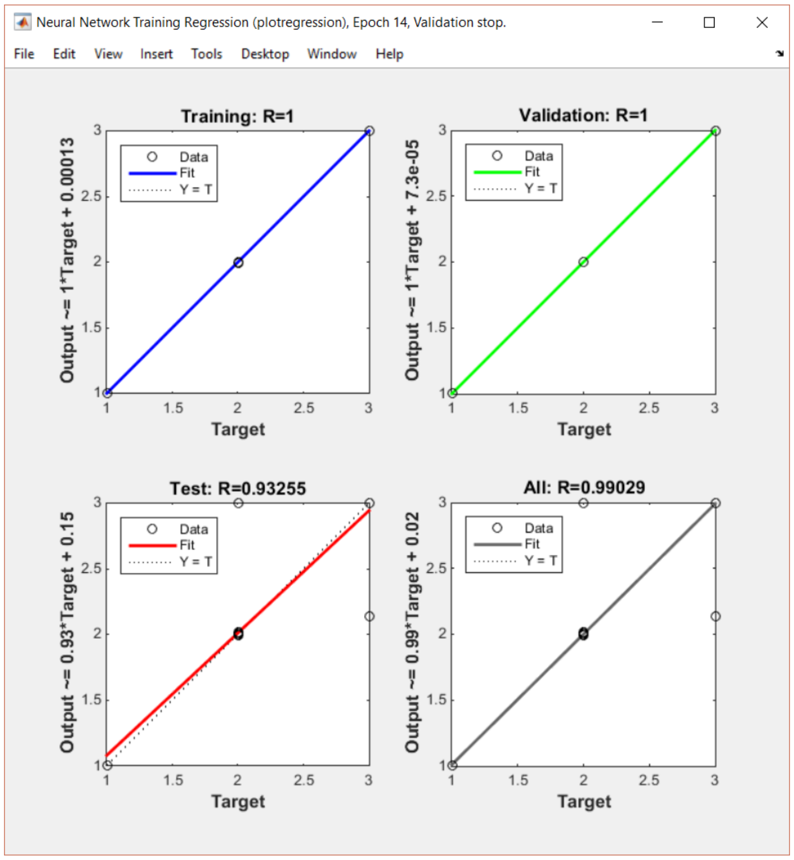
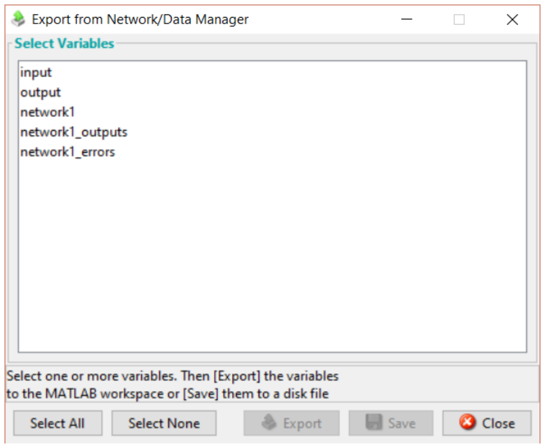

Neural Network Training Regression (plotregression), Epoch 14, Validation stop.
File Edit View Insert Tools Desktop Window Help
Training: R=1
Output ~= 1*Target + 0.00013
Validation: R=1
Output ~= 1*Target + 7.3e-05
Test: R=0.93255
Output ~= 0.93*Target + 0.15
All: R=0.99029
Output ~= 0.99*Target + 0.02
Data
Fit
Y = T
Target

Figure 18: Neural Network Training Regression

Once done with training, export the trained network to the workspace by selecting network and clicking Export.



input
output
network1
network1_outputs
network1_errors

Select one or more variables. Then [Export] the variables to the MATLAB workspace or [Save] them to a disk file

Figure 19: Export Window

Select the network name to export and export to Workspace.

PART 5: Simulating the Trained Network

The trained network can now be used with the test data to determine the efficiency of training on the network. The code below shows the way to test the network.

Test Neural Network (Trained)

```
testres = sim(iristest1,test);
testres = round(testres);
results = testres == test_out
```

The sim command is used to simulate the neural network with the data from **test** variable. The results are then rounded to obtain integer values which corresponds to the type of Iris flower from dataset. These results are compared with actual output of the test data stored in **test_out** variable.

Comparison can be performed in various ways. In this exercise, both "testres" and "test_out" are plotted using the following commands.

Check Differences between Trained Output and Original Output

```
figure('Name','Output Comparison')
hold on
plot(test_out,'b');
plot(testres,'r-*');
% Cleaning up graph presentation
axis([1 15 0 4])
set(gca,'YTick',[1 2 3]);
set(gca,'YTickLabel',['iris-seto';'iris-vers';'iris-virg'])
set(gca,'XTick',[1:15]);
```

```
xlabel('Sample No.');
ylabel ('Iris Species');
legend('Actual Result','Trained Result')
```

Note that the codes given above are just one way to plot. Self-experimentation is encouraged.

```
>>> set(gca,'YTick',[1 2 3]);
```

```
>>> set(gca,'YTickLabel',['iris-seto';'iris-vers';'iris-virg'])
```

```
>>> set(gca,'XTick',[1:15]);
```

These three commands are used to modify the values of the graph to reflect the species of Iris corresponding to the integer values mentioned previously.

The results for both a well-trained network and a badly trained network are shown below.

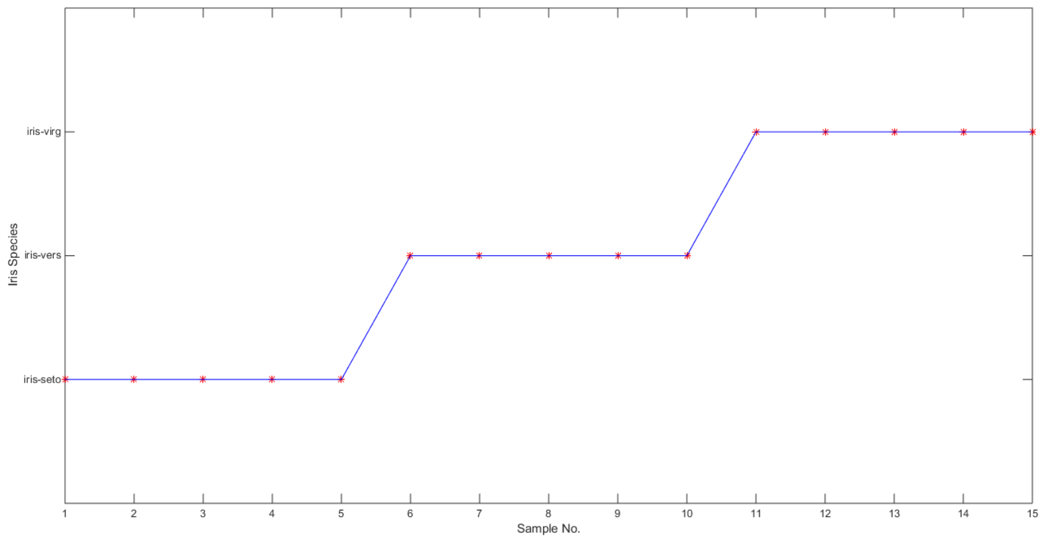


Figure 20: Properly Trained Data Comparison

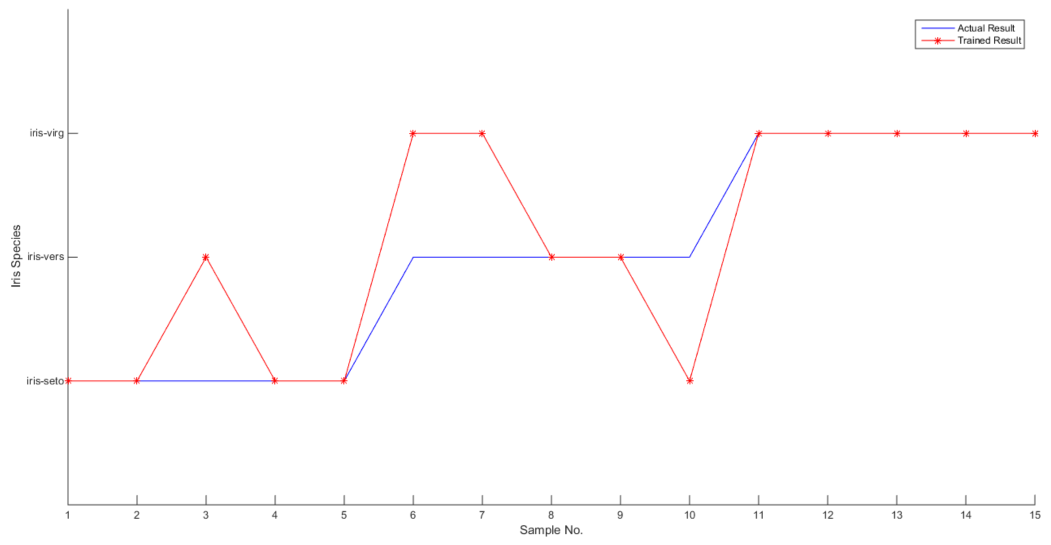


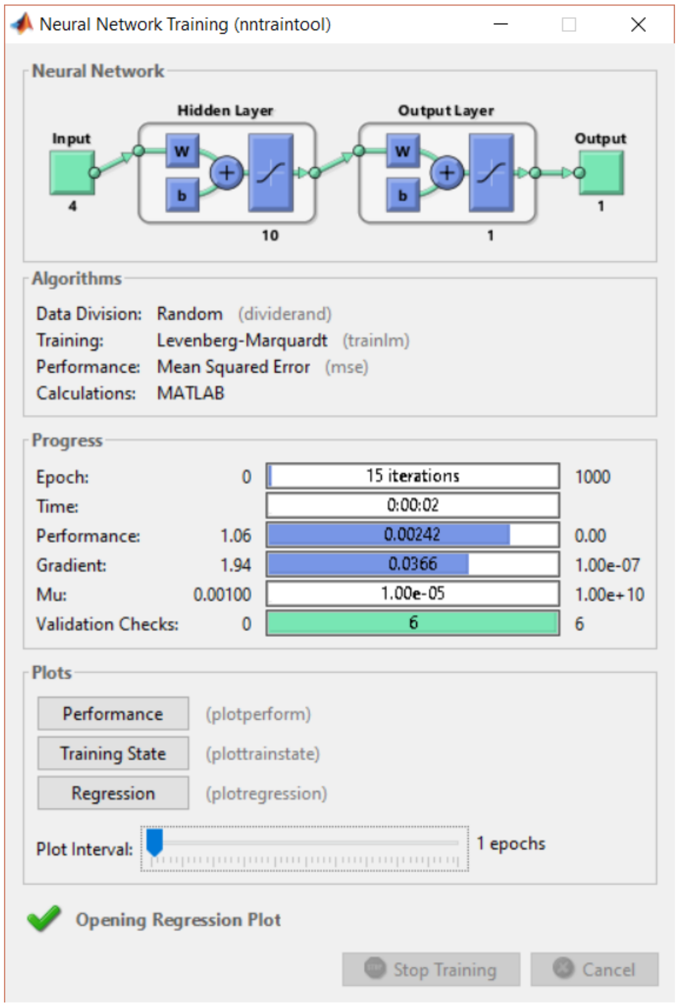
Figure 21: Insufficiently Trained Network

The first figure shows a network which can accurately describe the types of Iris flowers based on the data provided. If discrepancies exist between the actual result and trained result, then the network needs more training. Repeat the steps involving creation and training of network (found in earlier parts of this exercise).

PART 6: Further Experimentation

Many parameters of the networks can be changed via the Neural Network Toolbox. Students are encouraged to try changing the various parameters (namely number of neurons, number of layers, training function) and compare between them including their performance.

Hint: Look through the previous pages to find details on where to change parameters. The **Performance** of the system can be gauged from the training progress window as shown.



Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MATLAB

Progress

Epoch:	0	15 iterations	1000
Time:		0:00:02	
Performance:	1.06	0.00242	0.00
Gradient:	1.94	0.0366	1.00e-07
Mu:	0.00100	1.00e-05	1.00e+10
Validation Checks:	0	6	6

(plotperform)
(plottrainstate)
(plotregression)

Plot Interval: 1 epochs

Opening Regression Plot

Figure 22: Network Training Window (NNToolbox)

Conclusion

This exercise identifies the main process in which to use MATLAB Neural Network Toolbox to solve/analyze a dataset. This same process can be used with various other datasets and in other applications besides analyzing data.