# **Typographic Conventions**

Functions (script) names starting with Capital letters correspond to “GUI functions”

Functions (script) names starting with small letters correspond to “core mathematical functions” used to develop the network

All variable names start with capital letters

All structure names start with small letters

# **Functions used**

## **train\_mlp.m**

Used for training the MLP

### **Syntax**

[mlp MSE] = train\_mlp(Train\_X, Train\_Y, Hidden, MaxIterations, LearningRate, Momentum,Tolerance)

### **Input Arguments:**

Train\_X: Training inputs (Dimensions: Number of entries Number of features)

Train\_Y: Training targets (Dimensions: Number of entries Number of classes)

Hidden: Number of layers and number of neurons in each hidden layer

MaxIterations: Maximum number of training runs

LearningRate: Learning rate of MLP

Momentum: Learning momentum of MLP

Tolerance: Error tolerance of trained MLP

### **Output Arguments:**

mlp: Trained network (weights of each layer)

MSE: Training error (for each training run)

## **test\_mlp.m**

Used for evaluating MLP

### **Syntax**

[Output CC] = test\_mlp(mlp, Inputs, Targets)

### **Input Arguments:**

Inputs: Inputs (Dimensions: Number of entries Number of features)

Targets: Targets (Dimensions: Number of entries Number of classes)

mlp: Trained network (weights of each layer)

### **Output Arguments:**

Output: Output of network

CC: Correlation between network outputs and original targets

## **update\_mlp.m (Internal function)**

Used for updating MLP connection weights. This function is called inside train\_mlp

### **Syntax**

[mlp] = update\_mlp(mlp, Inputs, Targets)

### **Input Arguments:**

Inputs: Inputs (Dimensions: Number of entries Number of features)

Targets: Targets (Dimensions: Number of entries Number of classes)

mlp: Trained network (weights of each layer)

### **Output Arguments:**

mlp MLP structure with updated weights

### **Description**

A **multilayer perceptron** (MLP) is a primitive deep network. It comprises of feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. It consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation. MLP uses a supervised learning technique called backpropagation for training the network MLP is a modification of the standard linear perceptron and can distinguish data that are not linearly separable.

### **Demo**

To run the code without GUI, follow these steps

1. Load Train Inputs, Train Targets, Test Inputs and Test Targets
2. Specify tolerance, learning rate, momentum and maxim iterations
3. Call ‘train\_mlp’ function
4. For evaluating training data, call ‘test\_mlp’ by passing Train Inputs and Train Targets as input arguments
5. For evaluating testing data, call ‘test\_mlp’ by passing Test Inputs and Test Targets as input arguments

# **Data format**

Let’s say we have total M =1000 data entries and 5 classes. Let’s say each class is described by N=10 features, to . We can place entries of the classes in one matrix randomly.

Input data will look like as following

.

.

.

For target data, there should be M (2 in this case) column, each column belongs to a class. Input and target data will look like

Input Data Target Data

Class 1 Class2 Class3 Class4 Class5

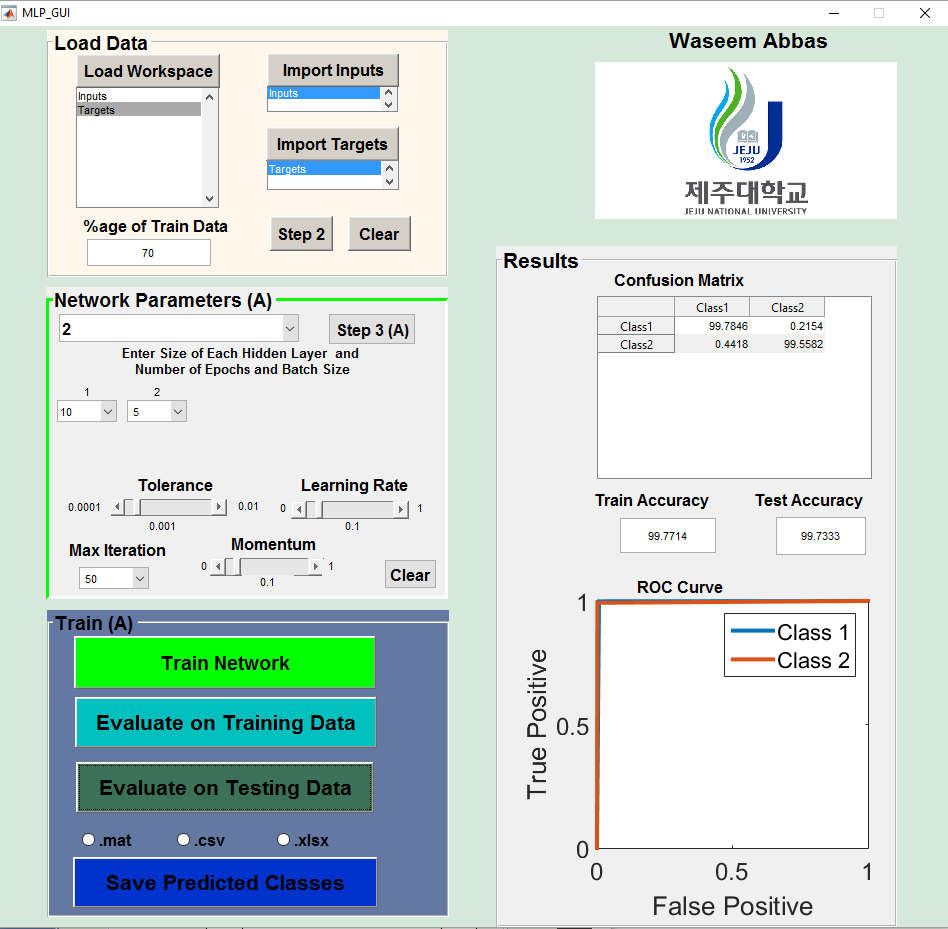
.

.

.

### GUI

Run ‘MLP\_GUI.m’



**2**

**3**

**4**

**1**

There are 4 parts of the GUI, Load Data, Network Parameters, Train and Results

## **Load Data**

This panel is used to load data from workspace. To import in puts, first click “load workspace” and then select the inputs. Then click on ‘import inputs’ button. Similarly, select targets and then click ‘Import Targets’. Once inputs and targets are imported, click on ‘step 2’. It will activate the next panel

## **Parameters**

This panel is used to set network parameters. Use following steps to properly initialize network parameters

1. Select number of hidden layers from drop down menu
2. Let’s say you have selected 4 hidden layers, you will see 4 popup menus created below. Each correspond to a hidden layer. Select number of neurons for each hidden layer
3. Select maximum iterations from corresponding drop down menu
4. Using corresponding sliders, specify the values of learning rate, momentum and tolerance
5. Once you have completed step a-d, click on ‘step 3’. Network parameters will be saved and next panel will be activated

## **Train**

This panel is used to train and evaluate the network. First, hit ‘Train Network’ button. When network is trained, it can be evaluated on train data and test data. The results can also be saved.

## **Results**

In this panel, confusion matrix and ROC curve can be observed

# CONVOLUTIONAL NEURAL NETWORKS