# **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

# **Functions used**

## **multisvmtrain.m**

Used for training an SVM structure for multiclass classification. Training is done in one-against-all manner. For every class, we train an SVM models which treats data as it is comprised of two groups, one group is the class for which SVM is trained and the other group is the ‘non class’

### **Syntax**

[models] = multisvmtrain(Train\_X,Train\_Y,options)

### **Input Arguments:**

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

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

options: This structure contains parameter values for SVM training. Some of these values are set by calling optimset command. These values are

'MaxIter' Maximum number of iterations for SVM training

'Display' Status of training we want to see in the workspace. In the GUI, this is set for iterations which means after every 500 iterations, status of training is displayed in command window.

'TolFun' Function tolerance value for training

Following ‘options fields’ must be set by the user

Stopping criteria for function

**'rbf\_sigma'**: A positive number specifying the scaling factor in the Gaussian radial basis function kernel. Default is 1.

**Polynomial Order:** Order of polynomial if kernel function is polynomial

**Kernel Function**

**Method:** A string specifying the method used to find the separating hyperplane. Choices are:

'SMO' - Sequential Minimal Optimization (SMO) method (default). It implements the L1 soft-margin SVM classifier.

‘QP’ - Quadratic programming (requires an Optimization Toolbox license). It the L2 soft-margin SVM classifier. Method 'QP' doesn't scale well for TRAINING with large number of observations.

‘LS’ - Least-squares method. It implements the L2 soft-margin SVM classifier.

An example is given below

options=optimset('MaxIter',MaxIter,'Display','iter','TolFun',TolFun);

options.StopCrit=0.01;

options.RBF\_Sigma= 0.9;

options.PolynomialOrder=6;

options.Method=’SMO’;

options.KernelFunction=’polynomial’;

### **Output Arguments:**

models: Trained SVM

## **multisvmtest.m**

### **Syntax**

[Predicted,Actual,Accuracy]=multisvmtest(X,Y,models)

### **Input Arguments**

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

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

**models:** Trained SVM

### **Output Arguments:**

**Predicted:** Labels predicted by ELM

**Actual:** Actual labels

**Accuracy:** Prediction Accuracy

# Demo

Run ‘demoMultiSVM.m’ for demo

# **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 two 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

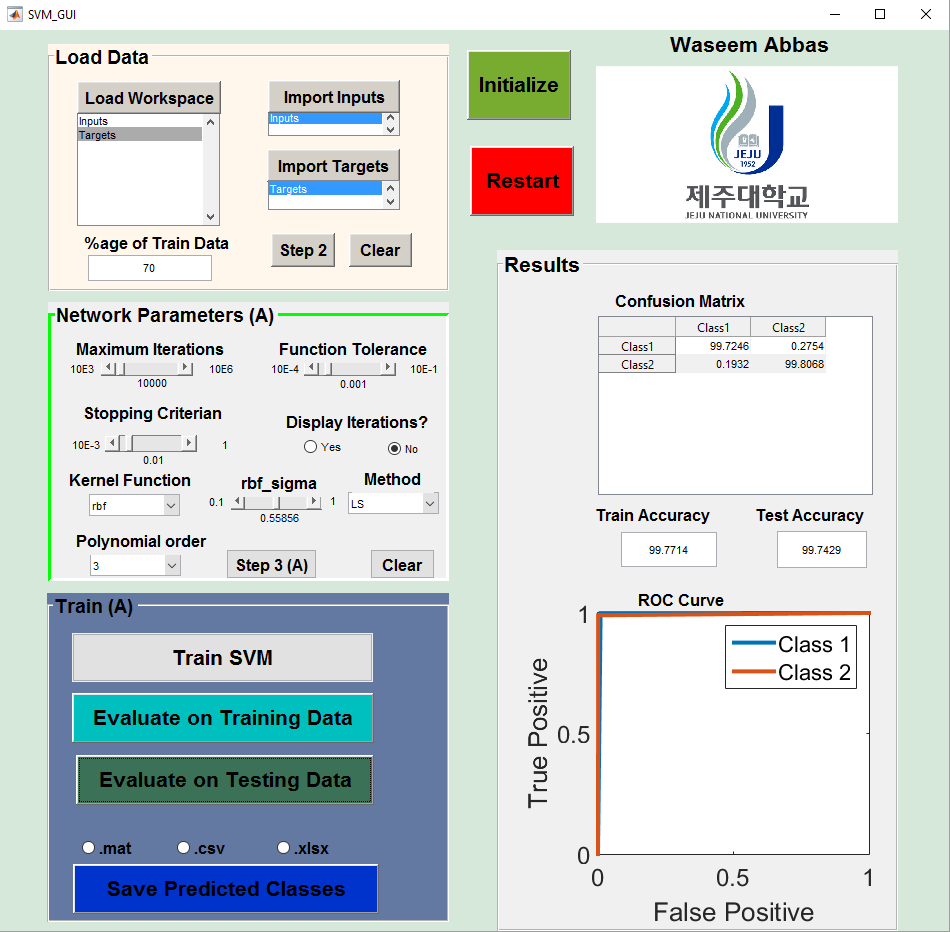
.

.

.

# **GUI**

Run ‘SVM\_GUI.m’



**4**

**3**

**2**

**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

## **SVM parameters**

This panel is used to set network parameters. For SVM, we can set maximum number of iterations, function tolerance, stopping criteria, display iterations, Kernel function, rbf sigma, polynomial order and hyper planes separating method. Once they are set, click on ‘step 3’. It will activate the next panel

## **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