Assignment 3

CSE 572:DATA MINING

Group Number 11

Sanay Devi

Koushik Kotamraju

Nikita Bahl

Shubhangi Gupta

Abhishek Kumar

For this assignment we have chosen the following group data to perform the classification exercises :

DM2, DM7, DM11, DM13, DM23, DM26, DM30, DM32, DM34, DM37

* **Steps for dataset creation:**

We selected each group to perform the following steps. Let us take DM 11 as an example for this process :

1. As we did for assignment 2, we generated a folder called ‘processed\_data’ which had 10 .csv files for each action. Each of these files was a vertical concatenation of data for each action done 20 times. An example file for the same is Action\_ABOUT.csv which contained 4641X40 cells. This was done by running the file called ‘processData.m’.
2. We generated all the features and performed PCA analysis on the ‘processed\_data’ folder using the file called ‘pca\_file\_generator.m’ with source directory as processed\_data. Later we multiplied the PCA output with the PCA input for each of the 10 actions separately and generated files named such as ‘pca\_ABOUT.csv’ in a separate folder called ‘pca\_data’. The pca for each of these files generated a 121X121 table which was multiplied with the pca input containing number\_of\_rowsX121 table for each action.
3. Now we executed the file called ‘model\_dataset.m’ from source directory pca\_data which concatenated all the pca\_ABOUT.csv, pca\_CAN.csv, etc file data (i.e. all the 10 action files pca data) into one from the pca\_data folder. 10 copies of this was made such that each file was named in the format all\_data\_ABOUT.csv (this is an example) which had labels assigned to it in the 122nd column such that if the file was for ABOUT action, 1 was assigned to the row or else 0.
4. Lastly, in order to get the shuffled dataset for each action, we read data form each of the files like all\_data\_ABOUT.csv files and randomly shuffled the rows to generate out final dataset which were named such as shuffle\_ABOUT.csv. This involved the execution of ‘shuffle.m’ file with source as pca\_data folder.

* **For Decision Trees :**

On each of the shuffle\_ABOUT.csv type of files, we executed the ‘decision\_tree.m’ file with source directory as pca\_data to generate the results for each of the 10 actions and calculated the recall, specificity, accuracy, true positive rate, false positive rate and precision values. The results are stored in separate pdf files for each group.

* **For SVM:**

We run SVM.m on each of the shuffled data files for each action for all the groups. We have 10 actions for each of 10 groups. We read these files one by one and get our dataset into variable X and the output classifier variable in Y. We split the initial data “data” into xData yClassLabel. Then we split the dataset xData into training(xtrain) and test data(xtest) sets using Cross Validation Partition as 60% training and 40% test. The corresponding classifiers are stored in ytrain and ytest. We train the svmModel on this training data using fitcsvcm function. After the training, we test the model by predicting the output using the SVM Model and our test dataset. Predicted results are stored in yPredict.

Confusion Matrix is calculated for each instance. Using this Confusing Matrix, performance metrics are calculated as follows:

confusionMatrix = confusionmat(ytest,yPredict);

trueNegative=confusionMatrix(1,1);

falsePositive=confusionMatrix(1,2);

falseNegative=confusionMatrix(2,1);

truePositive=confusionMatrix(2,2);

total=trueNegative+truePositive+falseNegative+falsePositive;

accuracy=(truePositive+trueNegative)/total;

precision=truePositive/(falsePositive+truePositive);

recall=truePositive/(truePositive+falseNegative);

f1Score=2\*(precision\*recall)/(precision+recall);

Performance metrics are displayed for each user each action and stored in the corresponding PDF Files. Accuracy of our file will always show hundred because we have correctly classified each Action with the correct class variable. For example, in our shuffled About.csv file of Group 11, all records for Action About are correctly labelled as one. To check our code in the testing phase we tried tweaking the values of the shuffled Action About files by misclassifying some of the records for Action About, now when we ran *SVM.m* on this, our values for accuracy, precision, recall and f1score all decreased.

After running Environment Testing we found out that the xlsread function used on a Windows Environment is not available of Mac OSX. So, in MacOSX environment csvread function is being used. For this a comment has been put in the SVM.m file indicating to the users in Windows environment to use the alternative code.

* **For Neural Networks:**

To train the machine using Neural Networks, we have used Neural Network Toolbox from Matlab.

• Fitting app Wizard (Input-Output and curve fitting) was used for building and training the network.Its equivalent built in function is nftool.

• The app will build a neural network to map between a data set of numeric inputs and a set of numeric targets.

• The Neural Fitting app helps to select data, create and train a network, and evaluate its performance using mean square error and regression analysis

Neural Networks.

A two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (fitnet), can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.

The network has been trained with LevenBerg-Marquardt algorithm with 20 neurons in the hidden layer.

Following Actions were passed as inputs to the Neural Network and the below mentioned observations were recorded.

ABOUT,AND,CAN,COP,DECIDE,DEAF,FATHER,FIND,HEARING, GOOUT.

* Following code splitted the data into 60% training and 40% testing data

%Split training Data into 60% Training and 40% Test

cv=cvpartition(length(data),'holdout',0.40);

xtrain = xData(cv.training,:);

ytrain = yClassLabel(cv.training,1);

xtest = xData(cv.test,:);

ytest = yClassLabel(cv.test,1);

inputNN = xtrain.';

outputNN = ytrain.';

net = feedforwardnet(20);

net = configure(net,inputNN,outputNN);

* The network was trained as below

net = train(net,inputNN,outputNN);

neuralNetwork = net(inputNN);

yNeuralPredicted = net(xtest.');

yNeuralPredicted = round(yNeuralPredicted');

* The confusion matrix was calculated to get precision,accuracy and f1 score

% Compute the confusion matrix

confusionMatrixNN = confusionmat(ytest,yNeuralPredicted);

trueNegativeNN=confusionMatrixNN(1,1);

falsePositiveNN=confusionMatrixNN(1,2);

falseNegativeNN=confusionMatrixNN(2,1);

truePositiveNN=confusionMatrixNN(2,2);

totalNN=trueNegativeNN+truePositiveNN+falseNegativeNN+falsePositiveNN;

accuracyNN=(truePositiveNN+trueNegativeNN)/totalNN;

precisionNN=truePositiveNN/(falsePositiveNN+truePositiveNN);

recallNN=truePositiveNN/(truePositiveNN+falseNegativeNN);

f1ScoreNN=2\*(precisionNN\*recallNN)/(precisionNN+recallNN);