**Data mining**

**Assignment – 4**

**Task 1:**

**Method:** We used PCA features resulted from previous assignment as input matrix to algorithms. We divided data set into training (60%) and testing data(40%), then trained the decision tree, support vector machine and neural network with it. We verified the results with testing data and measured accuracy with parameters such as recall, precision, F1, and AUC.

Definition of parameters :

**Accuracy**

To check the accuracy of the classifier, a confusion matrix is constructed with the expected output class and predicted output class.

Precision = True Positive / True Positive + False Positive

Recall = True Positive / True Positive + False negative

F Score = 2\* True Positive/ 2\*True Positive + False Positive + False Negative

1. **Decision tree (Classification):**

Classification decision tree gives responses in the leaf node in form of true or false.

**Input Predictor**: There are 3 predictors used(x1, x2, x3), which take the value of PCA features that we obtained in the previous task.

**Dimensions of feature matrix:**

Training Data: 94 x 3

Testing Data: 64 x 3

**Class-Name (Labels):** 1 (Eating actions) and 0 (Non eating action)

**Response**: 94 x 1

**Machine configuration:**

We provided three features which are acting as nodes, in some case one feature in enough to classifying eating with non-eating actions while in other cases the algorithm uses all three features. Below is the example of the first user:

Number of Branch nodes: 2 (x1 and x2)

Number of leaf nodes: 3

Decision tree created properties for the first user:

Decision cut value for first node: 2.545500000000000e+03

Decision cut value for second node: 1.423800000000000e+03

We have included Decision cut values of each user in DecisionTreecutPoint.csv file.

Below is the confusion matrix :



1. **Support vector machine :**

**Input Predictor**: There are 3 predictors used(x1, x2, x3), which take the value of PCA features that we obtained in the previous task.

**Dimensions of feature matrix:**

Training Data: 94 x 3

Testing Data: 64 x 3

**Class-Name (Labels):** 1 (Eating actions) and 0 (Non eating action)

**Response**: 94 x 1

**Machine configuration**:

KernalFunction: Linear ( two class learning separates data by hyperplane)

Standardize : True

Classification score is consistent with:

**F(x) = (x/s)*’β* + *b***

**Parameter values for the First user are below.**

Beta has the length equal to the number of predictors i.e. 3

-1.12579318375014

0.713362449897163

-1.19367402119581

Value of b: 0.046666263776054

Value of s (KernalParameters.scale) :1

The beta values are calculated so that it minimize ||β|| such that for all data points (xj,yj),

Yjf(xj ) > 0



1. **Neural Network:**

**Input Predictor**: There are 3 predictors used(x1, x2, x3), which take the value of PCA features that we obtained in the previous task.

**Input predictor dimension**: 156 x 3

**Training vs Testing Ratio**: 60/40

**Hidden layer size**: 12

**Outputs**: [0,1] if the output is less than 0.5 than it is characterized as non-eating actions and if output is more than 0.5 then characterized as eating action.

**Algorithm**: Scaled conjugate analysis: Scaled conjugate analysis can train any network as long as its weight, net input, and transfer functions have derivative functions. Back propagation is used to calculate derivatives of performance with respect to the weight and bias variables X.

**Task 2:**

1. **Decision tree (Classification):**

Classification decision tree gives responses in the leaf node in form of true or false. In this task, we need to classify if the output is from a Eating or Non-Eating class.

**Input Predictor**: There are 3 predictor used(x1, x2, x3), which take the value of PCA features that we obtained in previous task.

**Dimensions of feature matrix:**

Training Data:

For training the data of 10 users are taken. Shape is 80 x 3

Testing Data:

For testing purpose, the rest of 23 users data is taken.

**Class-Name (Labels):** 1 (Eating actions) and 0 (Non eating action)

**Machine configuration:**

Number of Predictors nodes: 3 (x1 and x2 and x3)

Number of total nodes: 77

**Method**

PCA is applied to the data of 10 users. Then this feature set is used to build Decision Tree Classifier. To create the predictor, we have used the fitctree inbuilt method of Matlab. The predictor is used to classify the rest of the 23 users data. To achieve this job, we have “predict” method, which takes the decision tree predictor and the testing data and output the result.

**Accuracy**

To check the accuracy of the classifier, a confusion matrix is constructed with the expected output class and predicted output class.

Precision = True Positive / True Positive + False Positive

Recall = True Positive / True Positive + False negative

F Score = 2\* True Positive/ 2\*True Positive + False Positive + False Negative

Below is the example of a confusion matrix for one user in decision tree:



1. **Support Vector Machine**

Support vector machine will give a partition boundary, which helps to partition the two classes in this case as Eating and Non eating actions.

**Input Predictor**: There are 3 predictors used(x1, x2, x3), which take the value of PCA features that we obtained in the previous task.

**Dimensions of feature matrix:**

Training Data:

For training, the data of 10 users are taken.

Testing Data:

For testing purpose, the rest of 23 users data is taken.

**Class-Name (Labels):** 1 (Eating actions) and 0 (Non eating action)

**Machine configuration**:

KernalFunction: Linear ( two class learning separates data by hyperplane)

Standardize : True

Classification score is consistent with:

**F(x) = (x/s)*’β* + *b***

**Parameter values for the First user are below. We have included parameters for all other users in svmParameter.csv**

Beta has the length equal to the number of predictors i.e. 3

-3.00709591851976

-0.711009255554207

0.302288987509145

Value of b: -0.736293214324382

Value of s (KernalParameters.scale) :1

The beta are calculate so that it minimize ||β|| such that for all data points

**Method**

PCA is applied to the data of 10 users. Then this feature set is used to build SVM Classifier. To create the predictor, we have used the fitcsvm inbuilt method of Matlab. The predictor is used to classify the rest of the 23 users data. To achieve this job, we have “predict” method, which takes the SVM predictor and the testing data and output the result. As the data is not in multi-dimensional, we have selected Linear SVM.

**Accuracy**

To check the accuracy of the classifier, a confusion matrix is constructed with the expected output class and predicted output class.

Precision = True Positive / True Positive + False Positive

Recall = True Positive / True Positive + False negative

F Score = 2\* True Positive/ 2\*True Positive + False Positive + False Negative

Below is the example of a confusion matrix for one user in SVM:



1. **Neural Network**

**Input Predictor**: There are 3 predictors used(x1, x2, x3), which take the value of PCA features that we obtained in the previous task.

**Input predictor dimension**: 156 x 3

**Training vs Testing Ratio**: 30:70

**Hidden layer size** : 12

**Outputs**: [0,1] if the output is less than 0.5 than it is characterized as non-eating actions and if output is more than 0.5 then characterized as eating action.

**Algorithm**: Scaled conjugate analysis: Scaled conjugate analysis can train any network as long as its weight, net input, and transfer functions have derivative functions. Back propagation is used to calculate derivatives of performance with respect to the weight and bias variables X.

**Method**

Input data of all users are divided into testing and training sets. The input neurons are fed into the network. Network has 1 input, 12 hidden layers and 1 output layer. The output of the activation function is in between [0,1]. We have fixed a threshold to classify out input into eating and non-eating categories.

**Result: For both the tasks Neural Network performed better than Decision tree and Support Vector. The AUC was observed better in user dependent task (close to 0.99).**

References : Matlab documentation

<https://www.mathworks.com/help>