ARJUN KOLUMAM RAMAKRISHNAN

AKOLUMA

ECE 8560

TAKEHOME #2

Introduction

This report explains in detail about the performance of the Bayesian classifier that we designed in takehome1 on test data. Also dimensionality reduction using PCA and the performance of Bayesian classifier on the reduced dimension test data is discussed. Apart from the Bayesian classifier, performance of k-Nearest Neighbor Algorithm and hyperplanes designed by Ho-Kayshap procedure on the test data given are discussed in this report.

Assessing Classification Result from Takehome #1

The true pattern of test data in takehome #1 was revealed to be 3-1-2-3-2-1 repeated 2500 times.

Probability of error

On comparing the true pattern with our classified answer we get a probability of error of 0.0881 or 8.81%. It is calculated used the following formula

$$P(error) = \frac{15000 - trace\ of\ confusion\ matrix\ of\ test\ data}{Total\ number\ of\ data\ points\ in\ test\ data}$$

Confusion Matrix of Test data

The confusion matrix obtained is shown below;

	Class1	Class2	Class3
Class1	4536	313	151
Class2	581	4393	26
Class3	210	40	4750

From the confusion matrix, it is seen that the performance of the classifier is good overall i.e. at least 4000+ features classified properly out of 5000 features in all the classes.

Separating Hyperplanes

Separating hyperplanes are developed under the assumption that the data is linearly separable. Hyperplanes are found for a pair of classes such that the resulting hyperplanes separates the data into those two classes. We consider classes pairwise between class 1 and combining class 2, class3, between class 2 and combining class1, class 3 and between class 3 and combining class2, class 1. The classes are multiplied by one and minus one respectively and then entered into the Y matrix. This is done to signify that the bounded pair of classes lies on the negative side and the other on the positive side. The Ho-Kayshap procedure for developing separating hyperplanes is as follows,

Start with arbitrary values for a and b greater than zero. Initialize initial value of b to zero.

Repeat the following steps till $error^{(k)} \geq 0$

- $Error^{(k)} = Ya^{(k)} - b^{(k)}$
- $b^{(k+1)} = b^{(k)} + \eta[error^{(k)} + |error^{(k)}|]$
- $a^{(k+1)} = ((Y^T Y)^{-1}) * Y^T * b^{(k+1)}$
- $k = k + 1$

The learning rate η is fixed between 0 and 1, in our case 0.5 has been chosen by trail-and-error method. Using the above method, we get three hyperplanes with the weight parameter "a" as the following;

For the hyperplane between class 1 and class 2, class3,

a = [-1.0177, 0.0041, -0.0017, -0.0173, -0.0112]

For the hyperplane between class 2 and class 3, class 1,

a= [0.2793 -0.0050 0.00041 -0.0077 0.01418]

For the hyperplane between class 3 and class 1, class 2,

a = [-0.2617, 0.0009, 0.0013, 0.0250, -0.0030]

The training data and test data are then multiplied with these "a" parameters for the three hyperplanes and based on which value is the greatest we assign the data point to the corresponding class. The confusion matrix obtained for the training data is,

	Class1	Class2	Class3
Class1	3962	672	366
Class2	1155	3845	0
Class3	608	13	4379

The probability of error for classification of training data is 0.1876 or 18.76%

Hyperplane Classifier Performance on Test Data

	Class1	Class2	Class3
Class1	3965	678	357
Class2	1129	3871	0
Class3	563	9	4428

The probability of error for classification of test data is 0.1824 or 18.24%

k-NNR Classification

The nearest neighbor algorithm was implemented for one, three and five neighbors. The implementation was as follows- find the distance of a single point in test data from every point in training data and then sort this data in ascending order and store the indexes of the first five smallest distances (since the maximum number of nearest neighbor we check is five). The formula for finding Euclidian distance is as follows,

$$distance = \sqrt{(x_{4i-}y_{4i})^2 + (x_{3i} - y_{3i})^2+(x_{2i} - y_{2i})^2+(x_{1i} - y_{1i})^2}$$

This process was then repeated for all point in test data and we obtain a matrix of 15000 rows and 5 columns. For 1-NNR, the first column of the matrix is considered i.e. the index of one nearest neighbor and if the index was between 1 and 5000, the test data point belonged to class 1. If it were between 5000 and 10000 it would belong to class 2 and if it were between 10000 and 15000 it would belong to class 3. The probability of error was found to be 0.1325 or 13.25% and the confusion matrix is as follows,

	Class1	Class2	Class3
Class1	4101	631	268
Class2	654	4264	82
Class3	289	63	4648

For 3-NNR, the index of three nearest neighbor for a test point were considered and they were each classified as specified above. The class which occurs more frequently (either 1, 2 or 3) was then taken as the class for that particular test point. This was achieved using the "mode" function in MATLAB. The probability of error was found to be 0.1139 or 11.39% and the confusion matrix is as follows,

	Class1	Class2	Class3
Class1	4305	482	213
Class2	659	4293	48
Class3	265	42	4693

For 5-NNR, the index of five nearest neighbor for a test point were considered and they were each classified as specified din 1-NNR. The class which occurs more frequently (either 1, 2 or 3) was then taken as the class for that particular test point. This was achieved using the "mode" function in MATLAB. The probability of error was found to be 0.1057 or 10.57% and the confusion matrix is as follows,

	Class1	Class2	Class3
Class1	4368	431	201
Class2	614	4340	46
Class3	243	51	4706

This method is made more computationally effective by not finding a distance matrix. Instead we sort the 15000 distance for each point in test data as soon as it computed and store the indices of the first five minimum distances as it is the only important data that we require to proceed further.

PCA

Principal Component Analysis (PCA) helps to remove insignificant features which have no effect in classification of test data. This reduces the test and training data from four-dimensions to two-dimensions. This reduction is done by the following steps-

- The mean normalized training data(i.e. training data centered at origin) is obtained by subtracting the mean of the input data from each element of the input data
- Covariance of the mean normalized input data is then found using the "cov" function in MATLAB. This covariance matrix gives the relative variance of each point in the data.
- "svd" is applied to this covariance matrix and this function arranges the Eigen value in descending order. The larger the eigen value, larger is the variance.
- We require maximum variance for easier classification and hence the first two eigen vectors corresponding to the two maximum eigen values are taken, transposed and multiplied with the training data to get the reduced training data of dimensions 2 x 15000. The dimension of the test data is reduced in a similar manner.
- With the reduced training data, a Bayesian classifier is then trained and it is used to classify the test data. The confusion matrix and probability of error of the new Bayesian classifier for both the training data and test data is as follows,

	Class1	Class2	Class3
Class1	3981	393	626
Class2	1006	3639	355
Class3	1337	77	3586

Probability of error (training data) = 0.2529 or 25.29%

	Class1	Class2	Class3
Class1	4016	381	603
Class2	1063	3561	376
Class3	1350	71	3579

Probability of error (test data) = 0.2563 or 25.63%

We can see that the probability of error of Bayesian classifier for reduced dimension test data (25.63%) is at least three times more than that of Bayesian classifier of normal test data (8.82%). This can be attributed to the loss of features in PCA. Hence PCA is more computationally effective whereas it has a high probability of error rate.

Method	Probability of error(%)
Bayesian on full dimension test data	8.81
Bayesian on reduced dimension test data	25.63

Comparing the results

By comparing the probabilities of error of test data for different methods, it is seen that Bayesian classifier has the lowest probability of error of 8.81%. The second best classifier is k-NNR algorithm. It has a probability of error of 13.25% for 1-NNR, 11.39% for 3-NNR and 10.57% for 5-NNR. But this method takes up most of the computational time of our program and is computationally inefficient even though we have suggested methods to improve it. Hyperplanes are very less efficient in classification as they have an error probability of 18.24% which is considerably higher than the other two methods. This could be because the data is non-separable and also because they are a non-parametric method. PCA is the poorest classifier, having an error probability of 25.63% and this is a bit on the higher side i.e. one fourth of the entire data is being classified incorrectly. The reason could be because many significant features were also lost while reducing the dimensions. This is also apparent from the deviation of probability of error of a test data with all four-dimension which is 8.81%. Though this method executes faster than other methods and utilizes less memory it reduces the possibility of classifying any given data properly. The table below illustrates how the various methods stack up against one another.

Method	Probability of Error of test data (%)
Bayesian classifier	8.81
5-NNR	10.57
3-NNR	11.39
1-NNR	13.25
Ho-Kayshap	18.24
PCA (Bayesian on reduced dimension)	25.63

Conclusion

Thus we have implemented the various methods as specified and have performed comparisons on the different types of classifiers.