ECE 8560 Takehome 2

Rohan Dani (rdani)

March 16, 2017

1 Confusion Matrix and P(error)

The true pattern was indicated to be 3-1-2-3-2-1 repeated 15000/6 times. Comparison between true pattern and takehome 1 answer was made using confusion matrix. Since the trace of confusion matrix gives the number of cases which were correctly classified, the formula used was as follows.

$$P(\text{error}) = \frac{\text{number of cases} - \text{cases corrrectly classified (trace)}}{\text{number of cases}}$$

The probability and confusion matrix are as follows (directly from MATLAB output window)

```
p_error_1 =
    0.0881

confusion_matrix =
        4536         313         151
         581        4393          26
         210          40        4750
```

2 Ho Kashyap Hyperplanes

Before we start to work on the training data, it is important to format the training data such that it has a column of ones before the data or after. In this case a column of ones before the data columns were used.
Considering the pairwise classification for the first case, class 1 was made positive and class 2 and 3 combined were made negative. For case 2, class 2 was made positive and class 1 and 3 combined were negative and so on. Instead of starting with arbitrary a and b, a was initialized to be the pinverse of b and b was initialized as matrix of ones. The algorithm and terminology used was as follows:

Repeat until $k < k_{max}$

- $e^{(k)} = Ya^{(k)} - b^{(k)}$
- $b^{(k+1)} = b^{(k)} + \eta[e^{(k)} + |e^{(k)}|]$
- $a^{(k+1)} = (Y^TY)^{-1}Y^Tb^{(k+1)}$
- $k = k + 1$

Through trial and error, learning rate(η) was chosen as 0.3 and k_{max} was chosen as 1000. The hyperplanes resulting from this algorithm is as follows:

```
a1 =
   -2.2835
    0.0129
   -0.0060
   -0.0539
   -0.0316

a2 =
    1.7270
   -0.0160
    0.0025
   -0.0250
    0.0407

a3 =
    0.5667
    0.0019
    0.0057
    0.0945
   -0.0072
```

3 Classifier performance and P(error) estimate - Ho Kashyap

The test data (with appended column of ones) is then multiplied to the hyperplanes to generate a classification strategy. We classify the case as class 1 if (test $* a1$) is maximum and so on. We get the following confusion matrix through this strategy.

```
confusion_matrix =
        3805         724         471
        1080        3920           0
         447          18        4535

p_error =
    0.1827
```

4 Classifier performance and P(error) estimate - k-NNR

The nearest neighbor algorithm considers the Euclidean distance of the test data points with the training data points. An efficient brute force method was used in this case.
For 1-NNR the first row was used to classify the data. The confusion matrix was found to be as follows:

1-NNR

```
confusion_matrix_1 =
        4100         632         268
         654        4264          82
         289          63        4648

p_error_1 =
    0.1325
```

For 3-NNR the first 3 rows were considered and if any class was classified twice for a test data point, that class was assigned to the point. The confusion matrix was found to be as follows:

3-NNR

```
confusion_matrix_2 =
        4283         491         226
         627        4310          63
         247          54        4699

p_error_2 =
    0.1139
```

For 5-NNR the first 5 rows were considered and if any class was classified thrice for a test data point, that class was assigned to the point. The confusion matrix was found to be as follows:

5-NNR

```
confusion_matrix_3 =
        4368         431         201
         616        4338          46
         243          51        4706

p_error_3 =
    0.1059
```

As we can see the probability of error slowly decreases as the number of NNRs increase.

5 PCA results

For PCA we remove the features with minimum variance since they do not add a lot of information for classification. The steps followed for PCA are as follows:

- The training data is mean normalized i.e. the mean is made zero by subtracting the mean from all data points in training data
- The relative variance is found by finding the covariance matrix
- Singular value decomposition is used to arrange the eigenvalues in descending order. The higher the eigenvalue, higher the variance.
- The eigenvectors corresponding to the two highest eigenvalues are considered and multiplied with training data to obtain a reduced set of training data, the test data is reduced similarly
- Like in takehome 1 we use a Bayesian classifier to classify the reduced set of test data using reduced set of training data

The confusion matrix and probability of error is as follows:

```
confusion_matrix_3 =
        4016        1063        1350
         381        3561          71
         603         376        3579
```

The probability of error is

```
p_error_3 =
    0.2563
```

6 Comparison and analysis

Method	**P(error)**
Bayesian	8.81%
Ho-Kashyap	18.27%
1-NNR	13.25%
3-NNR	11.39%
5-NNR	10.59%
PCA	25.63%

The Bayesian classifier is the best classifier with error percent as 8.81%.

Ho-Kashyap gives an error rate of 18.27% which is significantly worse than Bayesian classifier, the reason could be that the data might not be linearly separable.

k-NNR gives error rate of 13.25%, 11.39% and 10.59% for 1, 3 and 5 respectively however this takes up half a minute of run time to execute which is very slow, an alternative to brute force method is needed for shorter run time.

PCA gives an error percent of 25.63% which is understandable because we are working with half the data than before. However it is very fast so a compromise between computational speed and accuracy is needed.