Sabin KC

ENEE 436

Project 1

11/9/2020

Project Report

All the experiments on the MNIST data set was done using python 3 and sklearn library. FDA classifier was manually done. The results on each part of the question are briefly explained.

1. ***Naive Bayes***

   For the Naive Bayes data was assumed to be from gaussian distribution and its parameters were estimated using Maximum lIkelihood Estimation. First the model was trained using 60,000 image samples from the MNIST dataset and then the model was used to predict all of 60,000 image samples of training data as well as 10,000 test image samples. Following are the error rates for each case.

   Error Rate(Test Data): 43.5100 %

   Error Rate(Training Data): 44.4200 %

   As we can see the error rate is very high. This is because in Naive Bayes classifier we assumed that the features are independent of each other given a class, but this is hardly true in practical life. Also we assumed the data is from gaussian distribution, and derived its parameters, which may not be the case here.

2. ***K-Nearest Algorithm***

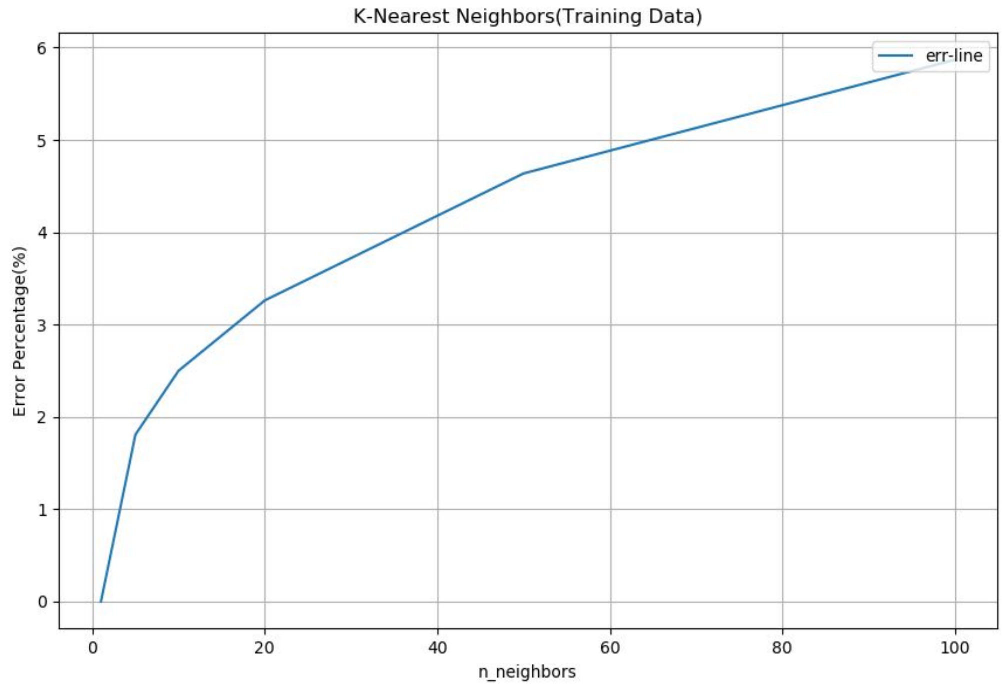
   The model was constructed using 60,000 training images where each image was treated as a point in 28 x 28 = 784-Dimensional space. These points are stored in a Ball Tree data structure to speed up the k-nearest neighbors calculation. Finally when we have the k neighbors, majority rule was used to determine the class of test sample. During the experiment the number of neighbors used were: 1, 5, 10, 20, 50, 100. In each case both training and test errors were collected. Following are the list of errors percentage according to the value of k.

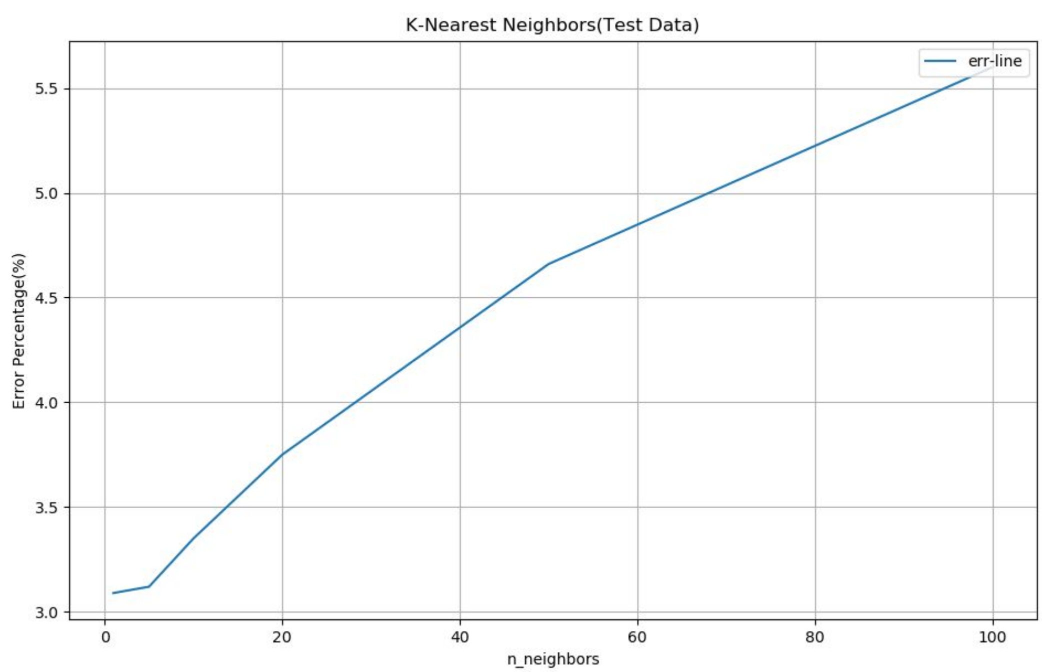
   Neighbors: [1, 5, 10, 20, 50, 100]

   Error Rate(Training Data): [0, 1.808, 2.5, 3.262, 4.637, 5.868]

   Error Rate(Test Data): [3.09, 3.12, 3.35, 3.75, 4.66, 5.6]

   The corresponding plots for training error line and test error line as the number of neighbors used are increased are:


K-Nearest Neighbors(Training Data)
err-line
Error Percentage(%)
n_neighbors
0
1
2
3
4
5
6
0
20
40
60
80
100


K-Nearest Neighbors(Test Data)
err-line
Error Percentage(%)
n_neighbors
3.0
3.5
4.0
4.5
5.0
5.5
0
20
40
60
80
100

As we can see as the amount of neighbors used increases the error rate also increases. This is expected for a training sample because when we use just 1 neighbor it will choose the sample itself as the distance will be 0 and the correct class is selected as a result every time. And as the number of neighbors increases, so does the probability of selecting wrong class. So there really isn't one value of k that gives the best result. Value of optimal K varies according to the problem and for this problem the optimal k seems to be 1.

3. ***Fisher's Linear Discriminant***

For each experiment the sample data was separated according to classes and used to train the model. Then the both training sample and test samples were used to test the accuracy of the model. Following are the error rates for each part of the question.

   a. ***Digit 0 VS Digit 9***

      Error Rate(Training Data): 0.4970 %

      Error Rate(Test Data): 1.1061 %

   b. ***Digit 0 VS Digit 8***

      Error Rate(Training Data): 1.1296 %

      Error Rate(Test Data): 0.9212 %

   c. ***Digit 1 VS Digit 7***

      Error Rate(Training Data): 0.5612 %

      Error Rate(Test Data): 1.1096 %

For this classifier first the data was projected into the optimal line obtained by the equation w = Sw^-1 (mu1 - mu2), when the scatter within the matrix was not invertible Moore-Penrose pseudo inverse was used. Then the data was projected into the line and the projected data was fitted according to MLE and univariate gaussian distribution. For the decision boundary, bayes linear discriminant was used as:

Choose wi such that gi(x) = p(x|wi). p(wi) is maximum for all i

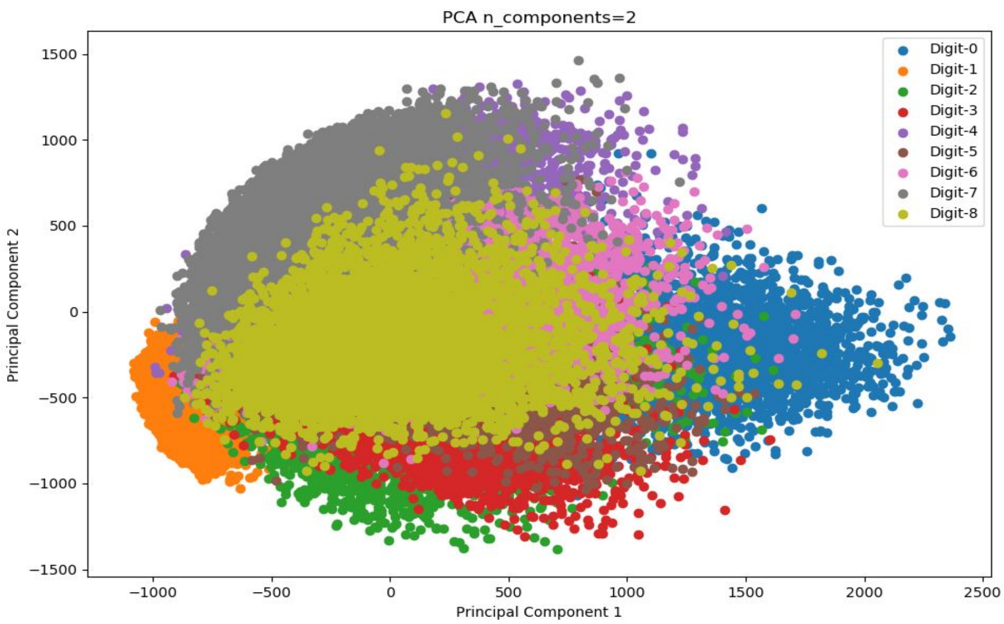
Here p(x | wi) is calculated according to the above estimated gaussian distribution.

As we can see the error rate for both training and test data are always below 2% which is a pretty good result so far.

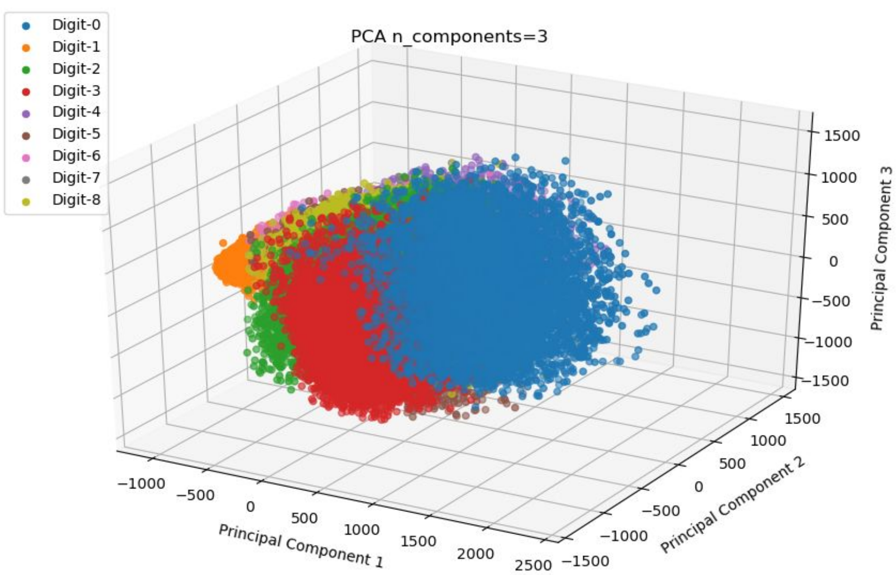
4. Principal Component Analysis (PCA)

First of all the training samples were projected into 2 and 3 dimensional spaces. The projection plots are as follows:

Projection in 2D space



Projection in 3D space



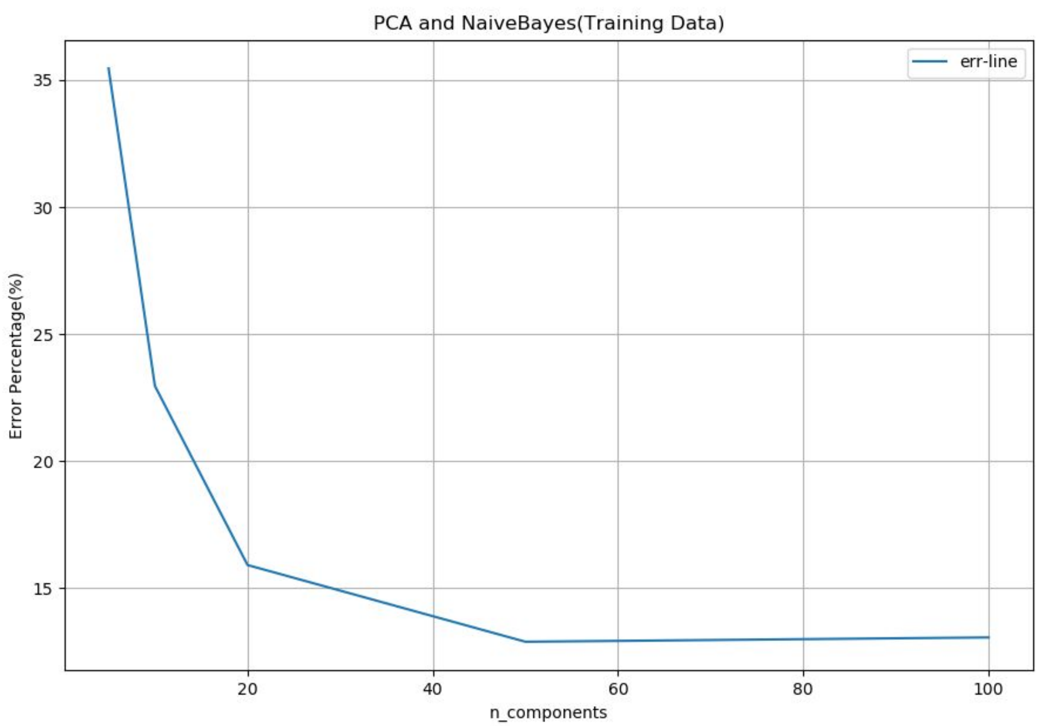
As we can see the projection is working so we proceed with experimentation. First the data was projected according to the component list as : [5, 10, 20, 50, 100] then in each projection Naive bayes and K-Nearest Neighbor model was constructed using projected training samples and then the models were tested by using both projected training and test samples. The error rate list and error curves are as follows:

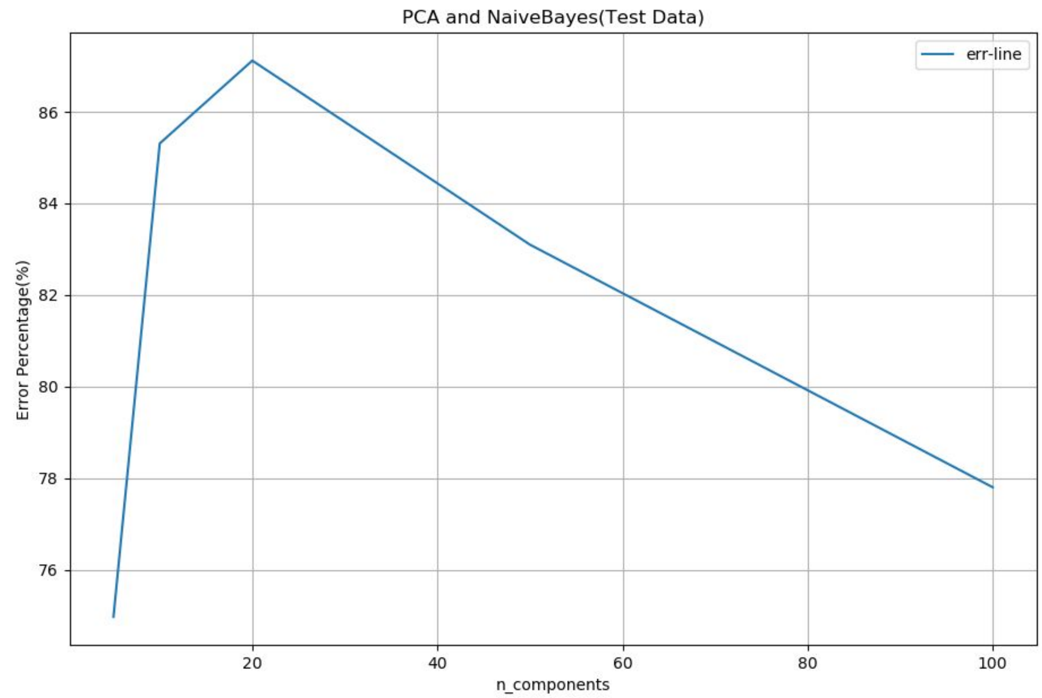
Components used: [5 , 10, 20, 50, 100]

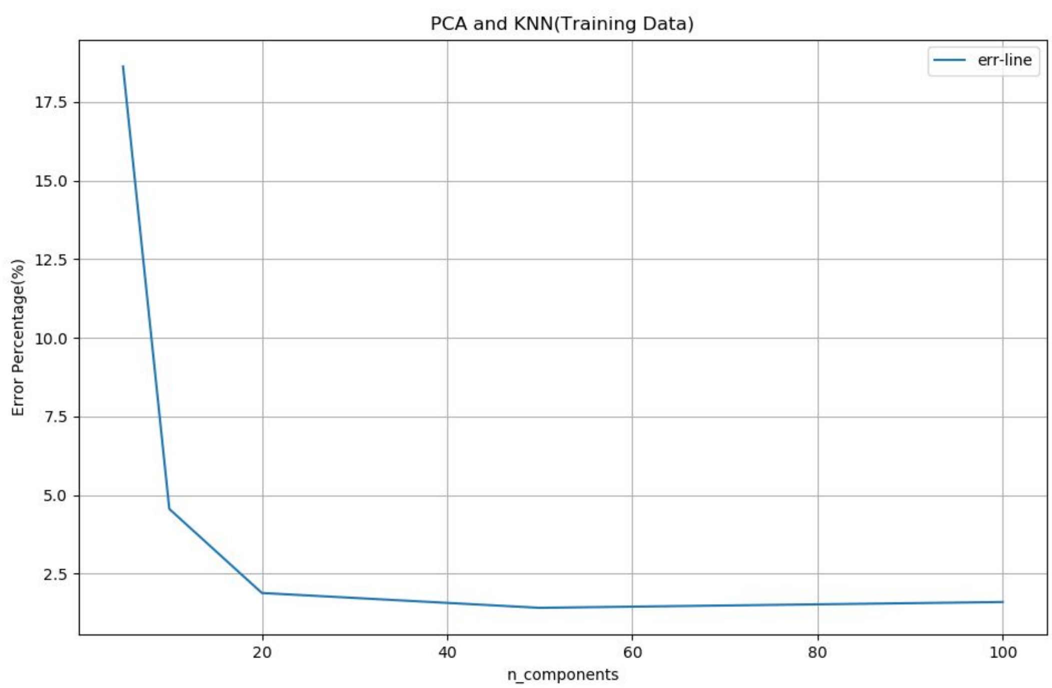
Error Rates Training Data(Naive Bayes): [35.46, 22.96, 15.908, 12.888, 13.058]
Error Rates Test Data(Naive Bayes): [74.97, 85.31, 87.12, 83.1, 77.8]

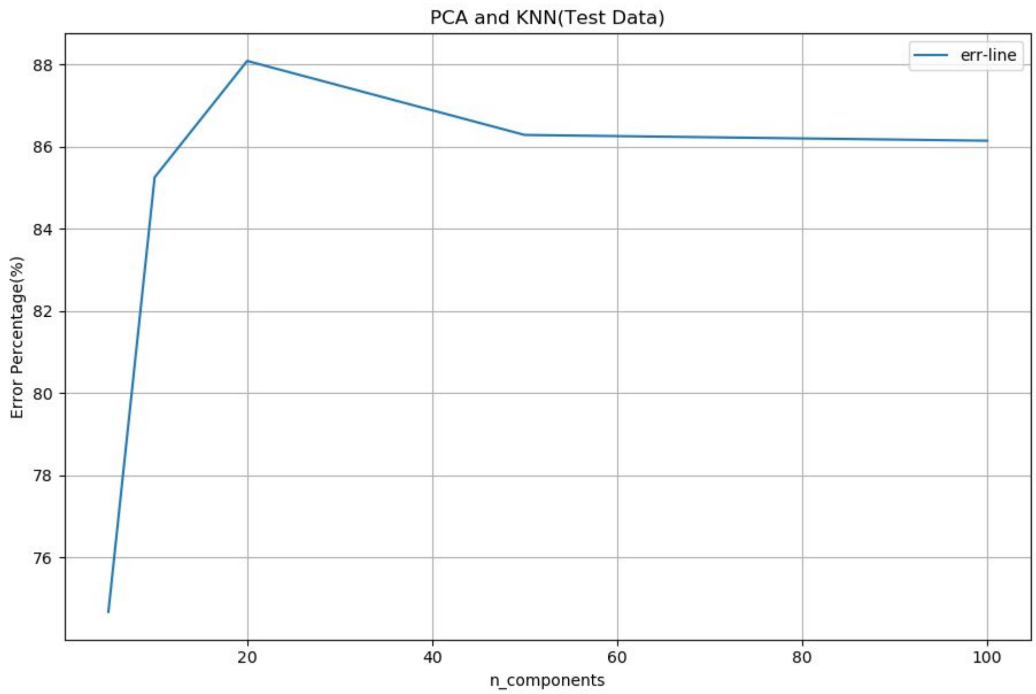
Error Rates Training Data(KNN): [18.633, 4.558, 1.885, 1.413, 1.598]
Error Rates Test Data(KNN): [74.68, 85.25, 88.08, 86.28, 86.14]

The Error Curves are as follows:




PCA and NaiveBayes(Test Data)
err-line
Error Percentage(%)
n_components
76
78
80
82
84
86
20
40
60
80
100


PCA and KNN(Training Data)
err-line
Error Percentage(%)
n_components
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20
40
60
80
100



By looking at the curves we can see that the error rates decrease as the dimensionality of projection subspace is increased. This is expected because as the more components are used for subspace more information from original data is retained compared to low dimensional subspaces. In both cases the training samples error rate is decreasing as the components are increased. On the other hand the training data's error rate seems to start low then high then start falling down again. And it seems like Naive Bayes is better than K-nearest neighbors. This may be because as the dimensionality of the data is decreased the conditional independence of the features are truly established, and the performance of Naive Bayes Classifier is improved.