PROGRAMMING ASSIGNMENT #1

Recognition of Handwritten Digits using Naive Bayes and KNN

Amrita Sundari V

1. Introduction

This programming assignment aims at recognition of handwritten digits using two classifier algorithms namely Naive Bayes and K-Nearest neighbour.

2. Getting familiarized with the data set

The Data set consist of two folders consisting the training and testing data. The training set consists of 1934 text files with the digit label as the name of each label. The testing set consists of 946 text files labelled the same way as the training set.

I have implemented the codes Loading_dataset.py to vectorize the dataset and subplots.py to display a binary image for each of the digits from the training data set to get an idea of the given sets.

I have converted each txt file into an array with 1024 entries, by unravelling the numbers in the txt file and combined all the txt files to create a single matrix of dimension 1934 x 1024 where each row contains the entries of each txt files. Testing set is also created in the same way and the matrix is of the dimension 946 x 1024. *Figure1* shows the result of displaying the binary images of each digit.

3. Implementation of Naive Bayes Classifier

I have used the Scikit learn package for modelling the Naive Bayes Classifier. The Naive Bayes classifier is very simple since it just computes the probability in a frequentist approach by just counting.

Testing and Training error rates and the corresponding errors are shown in the *Table 1*.

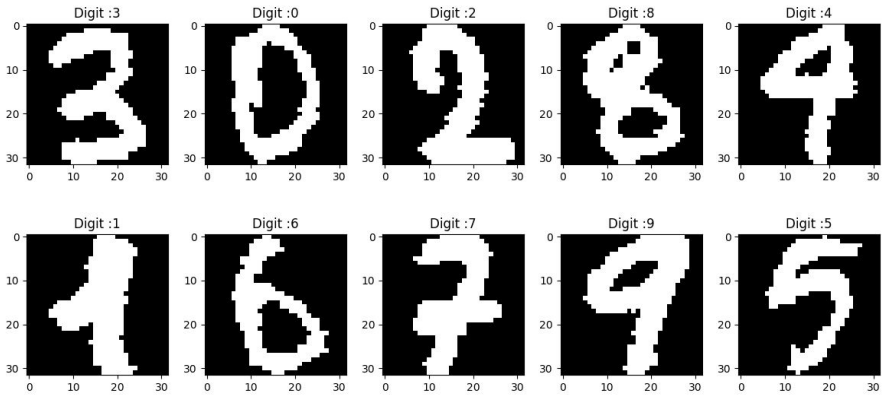

Figure 1.

	TESTING DATA	TRAINING DATA
ERRORS	252	403
ERROR RATES	0.2663	0.2083

Table 1: Testing and Training error rates of Naive Bayes Classifier

4. Implementation of K-Nearest Neighbor Classifier

The K-Nearest Neighbor Classifier takes the labels of the first k neighbors into account and classifies the new vector according to the most common label among the K neighboring points.

The Testing error rates and the Training error rates for KNN models with k= 1-10 have been plotted and is shown the figure 2.

As we can see from the figure, the testing error rates is minimum for k=3 in this region. Therefore k=3 works the best for this dataset.

The difference between the two rates is that, as we decrease the model complexity, the training error rates rises. But the testing error rates decreases up to a certain point and increases later on.

If we choose the model based on the training error rates we always end up choosing the more complex model, which in this case k=1. This is because as we increase the model complexity (decrease the value of k) the model is prone to overfitting. I.e the model should be able to predict all the training data points without any errors. This is why the training error rate at k=1 is zero and it increase as the model complexity reduces.

Similarly when it comes to the testing error rates, we can see that as we increase the model complexity (right to left in the plot; k decreases) the testing error falls. However, once we pass a certain point it starts to increase. The reason being, at higher complexity, the model fits the training data exactly and it does a poor job when it comes to predicting new data.

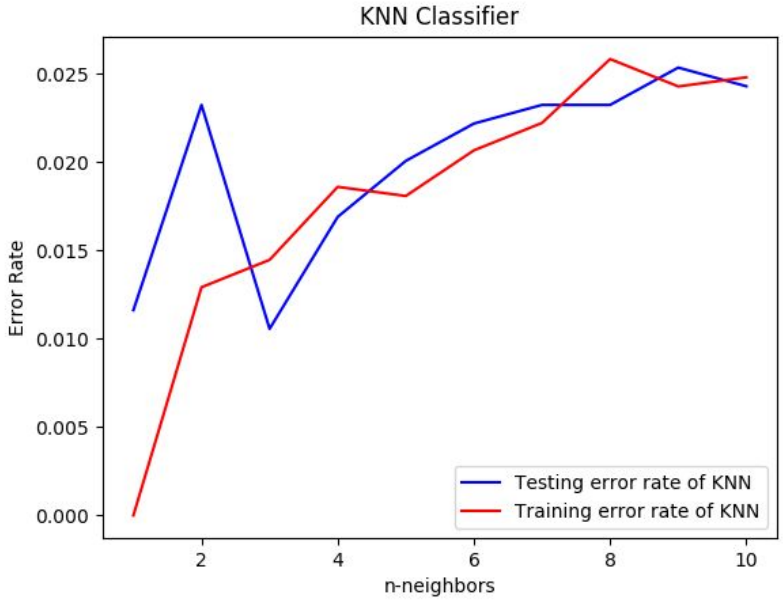

Figure 2: Testing and Training error plots of KNN classifier

5. Model Averaging

The main aim of the averaging of models is to combine the predictions of several classifiers in order to improve the generalisability over a single estimator. I have used the averaging technique called majority voting in three different ways.

Hard Majority voting: It makes the final predictions based on the maximum number of votes from the members of the averaged model.

Hard Weighted Majority voting: The predictions are based on the weighted votes from the members. I.e each member classifier will be given a certain weights and the final prediction will take these weights into account while averaging.

Soft Class probabilities: In contrast to the majority voting , this predicts the label as the argmax of the sum of predicted class probabilities.

5.1 Model Averaging of KNN:

The testing rates of different models that is k=1, average of k=1,2 , average of k = 1,2,3 and so on upto the average of all from k=1 to 10 and the testing rate of single knn models from k=1 to 10 has been plotted as shown in the figure 3.

As we can see the average models seem to work well when compared to a single KNN classifier.However, the best average model performs very similarly to the model KNN with k=3. This is because combined models are highly correlated. The performance of the average model is very high when the models are diverse i.e the predictions are independent. But since the models we used are all KNN models, the combined model did not significantly increase the performance.

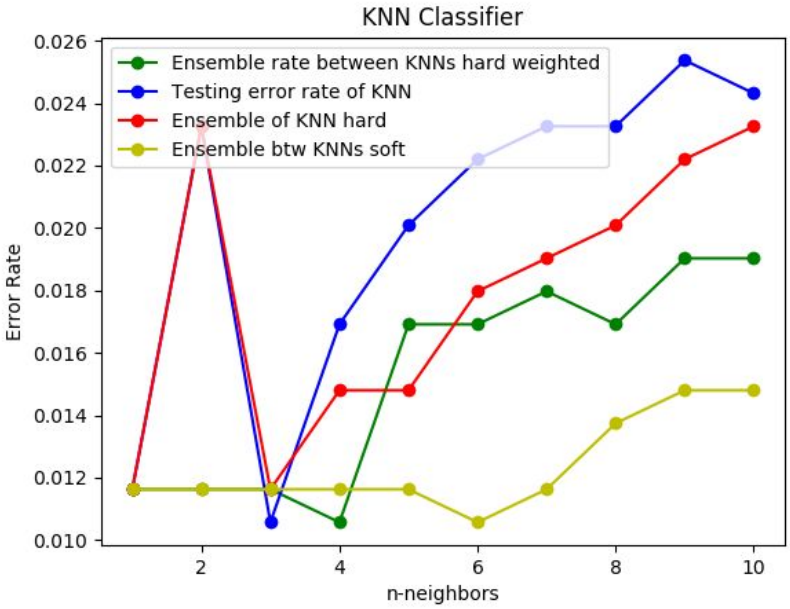

Figure 3

5.2 Model Averaging of KNN and Naive Bayes

I have also performed a model average of Naive Bayes and KNN with different k values and the performance is better than that of the individual KNN or NaiveBayes model as we can see from figure 4.

To compare the classifiers, the Training and Testing error rates of Naive Bayes and KNN(k=3) are as shown in the following table 2.

6. Pros and Cons of the Classifiers

6.1 K-Nearest Neighbour

Pros:

- It is very simple to implement and there is no assumptions required to model any type of data.
- Flexible to feature/ distance choices.Naturally handles multi class problems.

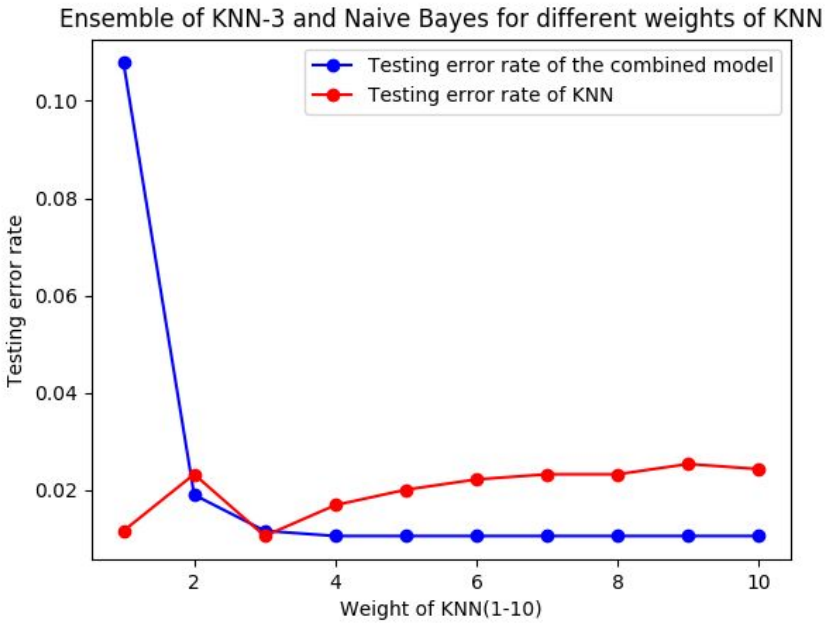

Figure 4

	Naive Bayes Classifier	K-Nearest Neighbor (k=3)
Testing Error Rate	0.2663	0.010571
Training Error Rate	0.2083	0.014478

Table 2.

Cons:

- The Main disadvantage of the KNN algorithm is that it is a lazy learner i.e, it doesn't learn anything from the training data but uses the training data every time a prediction is made.
- Since it calculates the distance and sorts the training data every time a prediction is made, the computation can become very slow for large training data set.
- It is computationally expensive to find the optimal k value for a large training set.

6.2 Naive Bayes

Pros:

- Computationally fast and it also performs well in the multi class problems.
- If the independence between the features condition is met, it converges faster when compared to other classifiers with less training samples.

Cons:

- The main disadvantage of this classifier is that it makes a strong assumption that the features are independent. But surprisingly it doesn't perform significantly bad even in the case where the assumption does not hold.
- Data scarcity can be a problem, since it estimates the likelihood value by a frequentist approach and this can result in probabilities like 0 or 1, which in turn leads to worse results . In such cases we can smoothen our probabilities somehow or can impose some prior on the data .

Possible ways to improve the accuracy of Naive Bayes classifier

- We can use other distributions to model the likelihood of the data and then computing the parameters of the distribution with the training data set. For example we can use Gaussian distribution to model the data. Gaussian NB assumes the likelihood to be a gaussian distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$. The parameters ($\boldsymbol{\mu}$, $\boldsymbol{\sigma}^2$) are estimated using maximum likelihood.

- Another way to improve the accuracy is to remove the redundant features in the data set. This can be done using dimensionality reduction techniques such as Principle Component Analysis(PCA) which is discussed in the next section.

7. Principal Component Analysis

Principal component analysis technique is used to reduce the dimension of the vectors by mapping the data along the axis of maximum variance. The feature corresponding to the maximum variance is called as the principal component. By doing this we can improve the efficiency of the classification by removing the unnecessary information from the data and model it with only the features that is important.

The important task is to choose the number of principal components that has to be accounted in order to retain the complete information. For this I have computed the error rates of both the classifiers i.e Naive Bayes and K-Nearest Neighbor with respect to different number of principal components chosen.

The following plot (figure 5) shows the variation of the error rates.

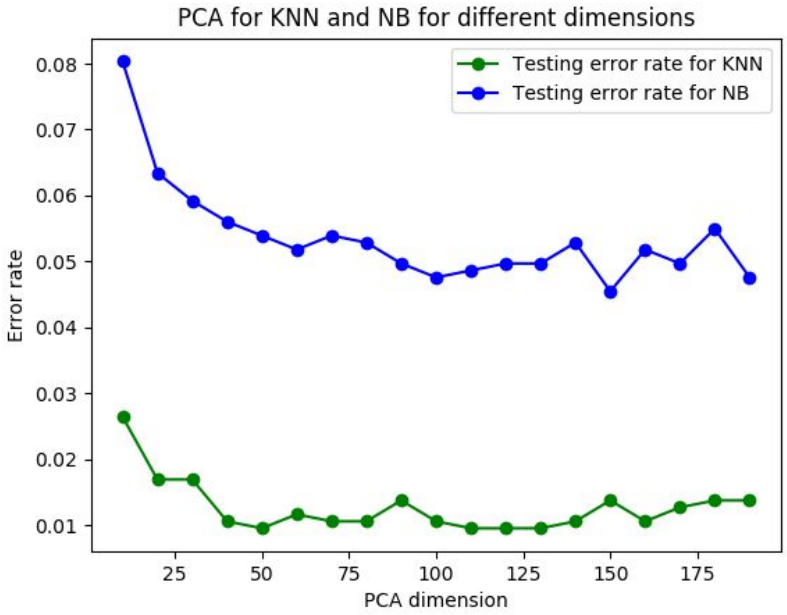

Figure 5

From figure 5, we can see the error rates reaches the minimum around the dimensions in the range 100-130. Thus we can reduce the dimension of the data from 1024 to 120 without losing useful information from it.

	Naive Bayes before/after PCA	KNN before/after PCA
Testing Error rates	0.2663/0.04569	0.010571/0.009514

Table 3.

The testing error rates for Naive Bayes and KNN after applying PCA are found to be 0.04569 and 0.009514. Thus the accuracy has been increased in both the classifiers after applying PCA as shown in the table 3

8. Conclusion

Thus we have discussed about the various ways to model a classifier to predict the digit recognition task and also about the pros and cons of the various techniques.