Eigen Digits Classification

Name: Kunal Lad
UT Eid: KL28697
Email: kunal.lad@utexas.edu

Introduction
Methodology
  Data Details:
  Algorithm:
Experiments
  E1: Performance variation with size of training data
  E2: Performance variation with number of principal components used
  E3: Performance variation with num of neighbors in kNN
  E4: Performance variation with distance metric
Discussion
  Conclusions:
  Future Work:
References

Introduction

The goal of this assignment is to use Principal Component Analysis (Eigen Vector decomposition) for high dimensional data and see how it works with a classifier such as kNN. Main motivation for such models is that many data sets have the property that the data points all lie close to a much lower dimensionality subspace than that of the original data space and hence we can project the data onto this subspace and use it as an approximation to get huge speed benefits.

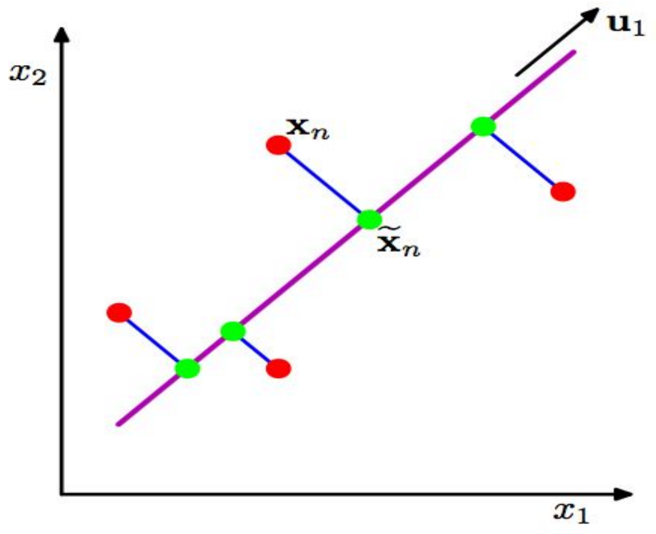

Fig 1 (Courtesy [1]): Principal component analysis seeks a space of lower dimensionality, known as the principal subspace and denoted by the magenta line,

such that the orthogonal projection of the data points (red dots) onto this subspace maximizes the variance of the projected points (green dots). An alternative definition of PCA is based on minimizing the sum-of-squares of the projection errors, indicated by the blue lines.

Methodology

Data Details:

- Training Data consists of 60000 images of size 28 x 28
- Test Data consists of 10000 images (5000 easy, 5000 hard) of size 28 x 28

Algorithm:

1. Reshape all 28 x 28 images to 784 dimensional column vectors.
2. Randomly sample k << 784 images from training data and call the matrix A.
3. Compute eigen vectors of covariance matrix of A^T
4. Computer top k eigenvectors of A by multiplying eigen vectors obtained in previous step by A.
5. Sort them by decreasing order of their corresponding eigen values.
6. Choose a subset of top v vectors such that sum of their eigen values is greater than 99% of sum of all eigen values, as the Eigen Basis and normalize it.
7. Train kNN classifier with sampled data projected onto subspace of eigen basis (after subtracting mean).
8. Predict labels for test data by performing same transformation (mean subtraction followed by projection onto eigen subspace) on test data and using above model.

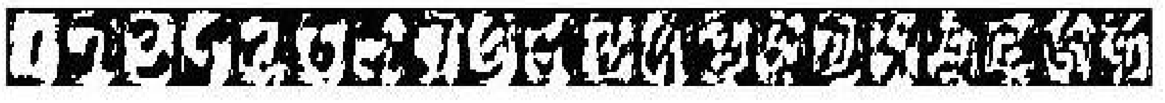
Fig 2: Top 20 Eigen Vectors

Fig 3: Original Digit Images

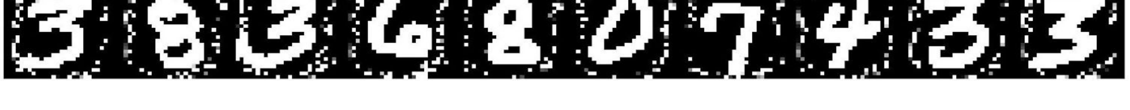
Fig 4: Reconstructed Digit Images after projecting onto eigen subspace

This reconstruction shows that reconstructed image lose some information and are not exactly similar to original image.

Experiments

This section gives details of the experiments performed in order to asses the performance of our algorithm and see how it varies with various parameters involved. For all the experiments mentioned in this section we use the following unless specified otherwise:

- Random Sampling using datasample command in MATLAB for sampling training and test data.
- 500 images for training and 2000 test images (1000 hard, 1000 easy) for evaluating the accuracy.
- Data used for finding eigen vectors is also used for training kNN classifier.
- Used fitckNN method in MATLAB for training k Nearest Neighbor classifier model with default parameters.
- Use all eigen vectors found from test data.

E1: Performance variation with size of training data

In this experiment we track algorithm accuracy against the number of samples used for training. As the plot suggests that accuracy increases sharply up to a point and then tapers off giving very small gains even with large increase in data size after ~275

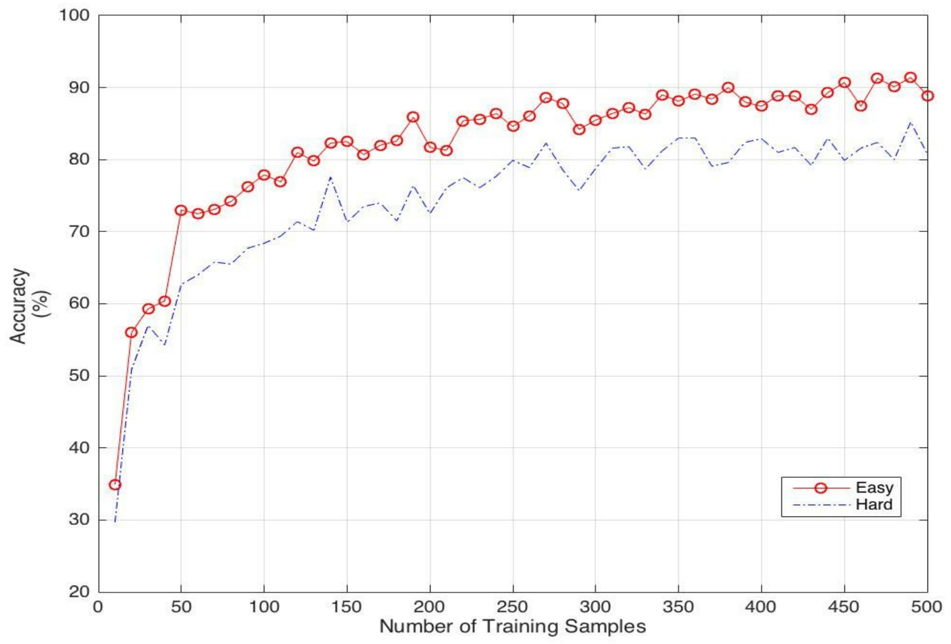

Fig 5: Plot of Accuracy Vs Size of Training Data

E2: Performance variation with number of principal components used

In this experiment we use only a fixed subset of top eigen vectors as our eigen basis. Accuracy plots show that there there is a huge increase in accuracy even with a small increase in number of eigen vectors used, but not much after around ~30. This suggests that almost all the information of original data space (dimension 784) can be captured in much smaller dimensional space (~30) and there is no significant gain in using higher dimensional space than that. Thus we can speed up our computations to a huge extent without much loss in accuracy by projecting the data to a much smaller subspace.

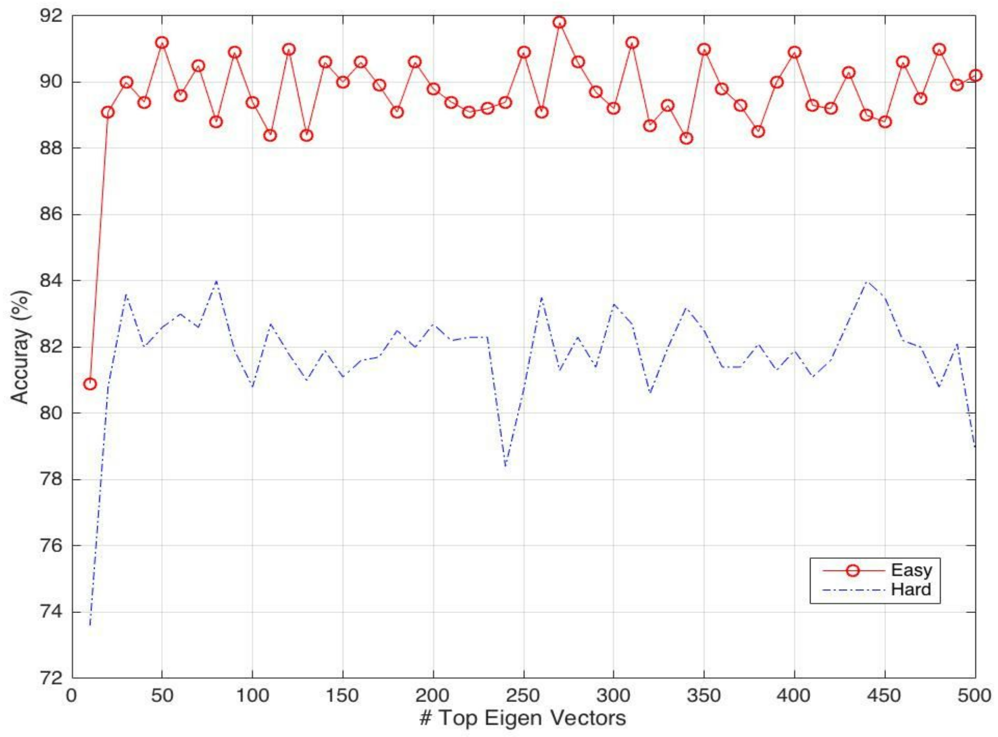

Fig 6: Plot of Accuracy Vs Number of Principal Componets used

E3: Performance variation with num of neighbors in kNN

In this experiment we vary the number of neighbors used for training k Nearest Neighbor Classifier. From accuracy plots we can see that the peak for easy test data occurs at k = 2 and the peak for hard test data occurs at k = 7. From both plots it is clear that their is no significant improvement after these peak points. Infact the accuracy decreases if the number of neighbors increase beyond a point. This is expected because if we use larger number of neighbors then points close to different class boundaries might get misclassified.

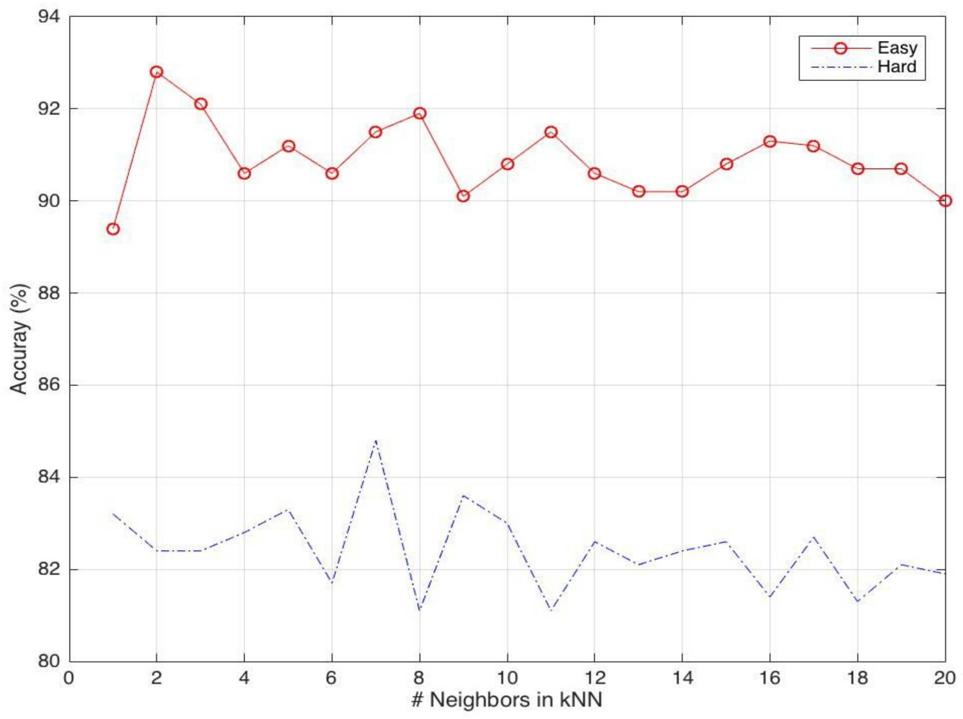

Fig 7: Plot of Accuracy Vs Number of Neighbors used in kNN Classifier

E4: Performance variation with distance metric

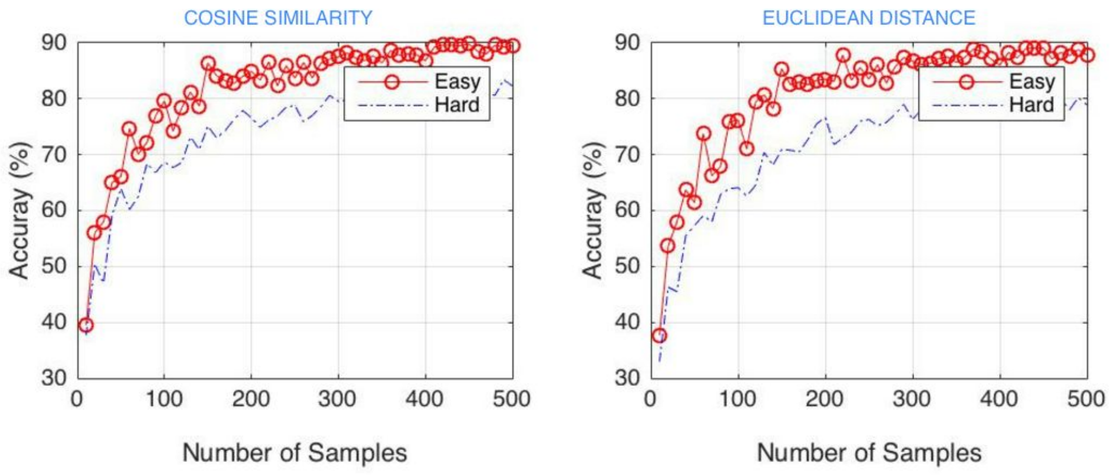

Fig 8 : Comparison of Accuracy for Cosine Similarity and Euclidean Distance metric

In this experiment we vary the distance metric used for kNN classifier. From the plots we can see that the curves for both distance measures have almost similar shape. This shows us that both the distance measures serve well for our task if we have right amount of data to train our model.

Discussion

Conclusions:

We can use PCA for dimensionality reduction of high dimensional data (if it resides close to lower dimensional subspace) and improve computational efficiency of the algorithm. All the experiments performed clearly show that the amount of data and number of principal components used are the most important metric for algorithm performance. Other parameters like distance, number of neighbors are not very important. From the experimental results we could also conclude that increasing the training data and number of principal components improves the performance only upto a point (till we have captured the dimensionality of the underlying data).

Future Work:

Currently our framework requires all the images to be of same size. But this is not a realistic assumption. We would want to train and test on data images collected from different sources which are likely to have different sizes. Hence we would like to use some Image Processing techniques to rescale all the images to a standard size before applying our algorithm. Secondly PCA is sensitive to outliers and skewed data so we would want to use some techniques to clean and prepare the data to make it more robust.

References

[1] Pattern Recognition And Machine Learning by Christopher Bishop