Digit Recognition

Nayan Singhal, ns28844

February 9, 2017

1 Introduction

Hand Digit Recognition is a well renowned problem, and people have come up with a lot of ways to solve the problem. One of the ways to solve the problem is through statistical rules, but it will be very hard because every person has different handwriting style and we can't generalize the rules for all the users. The other way to solve the problem is through Machine Learning in which the system will automatically learn different handwriting and can identify them. But most of the time, the feature vector is highly correlated and very hard to distinguish in that coordinate axis. So, it's beneficial if we can transform them into some other coordinate frame so that they are separable in that coordinate frame.

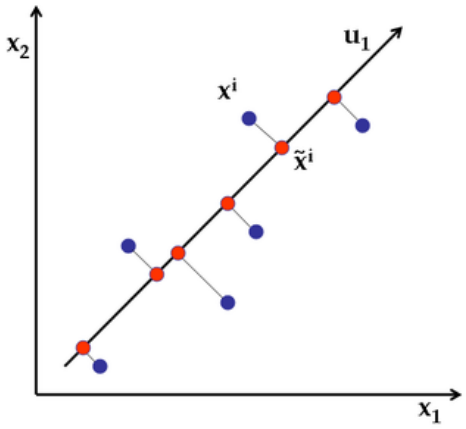

Figure 1: credits: Upenn ML lecture. Principal Component Analysis is a linear dimension technique which tries to project into a lower dimensional space known as principal subspace, which is shown by line u1. It tries to project the points (blue dots) into the principal space shown in red points so that it can preserve as much variation as possible.

The process of finding the best coordinate frame is called Principal Component Analysis. PCA converts the set of vectors that is correlated and very hard to distinguish in that coordinate frame into the set of vectors that are uncorrelated to each other. The number of principal components is less or equal to the number of the original vectors. The first principal component has highest

variance and all other succeeding components have variance smaller than the previous ones. The other advantage of using PCA is that it helps in fast training of the model as our dataset is projected onto some lower dimensional space.

2 Methodology

We have 60,000 train images and 10,000 test images in which each image is of size 28*28.

1. Reshape the image of size 28*28 into a vector of size 786*1.
2. Select some k random images <<n (number of pixels for each image i.e. 786) and call it as matrix A.
3. Calculate the mean for the k images.
4. Compute the Eigen Vectors for the matrix (A'*A) .
5. Compute the Eigen Vectors for the covariance matrix by multiplying the Eigen Vectors obtained in step 4 with the matrix A.
6. Select a subset of top v Eigen vectors such that the sum of their Eigen Values is 99% greater than the sum of all the Eigen Values, which we choose it as a basis vector for coordinate transformation by normalizing them.
7. We trained our KNN classifier by subtracting the mean digit image and projecting all of the training examples into v Eigen coordinate frame.
8. Finally, we predict the label for the test data by subtracting the mean digit image and projecting them in Eigen Coordinate frame and calculate the accuracy.

Figure 2: 10 random original digit images.

Figure 3: 10 original images after mean digit subtraction. Image 5 (digit 9) is little tricky to understand in real image but after mean subtraction, it's more clearer and easily recognizable

3 Experiment

To better understand the PCA algorithm, we did the bunch of experiments. This section will give you the details of of the experiments which we have done in order to asses the performance of the algorithm.

Figure 4: Top 20 Eigen Vectors generated from the random train samples.

Figure 5: Effect of Eigen Vector projection on images: Reconstruction of Digit 8 image as shown in first image of Figure 2, using Eigen Vectors from range [50-700]. Image generated from 50 Eigen vector is very hard to recognize but the one generated from 700 Eigen vectors is clean and easily recognizable

3.1 Performance variation with Eigen Vectors

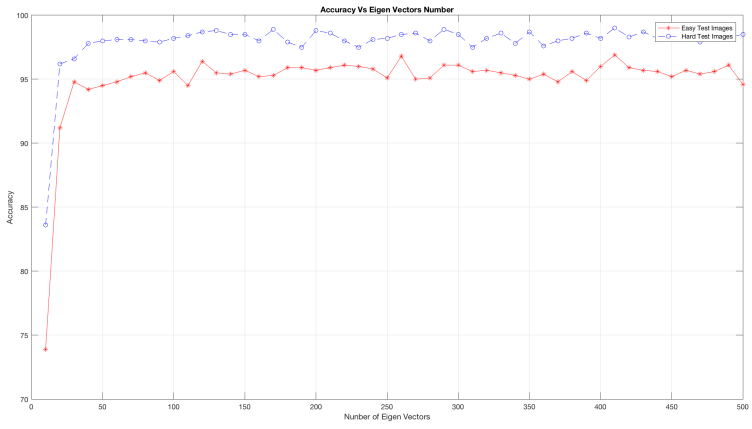

Figure 6: Accuracy vs Eigen Vector Number Generated.

In this experiment, we fix the number of test sets to 60,000 images and vary the number of Eigen Vectors as our basis vectors. From the figure 6, it can be seen that our model gains a significant increase in the accuracy with a small increase in the number of Eigen Vectors. But, there is no significant gain in accuracy while using more Eigen Vectors (can be seen from the range of 300-

500). This suggests that all the data can be easily represented in the lower dimensional space. And, it will boost up the performance if we use less number of Coordinate frames to represent the same image.

3.2 Performance variation with Sample Size

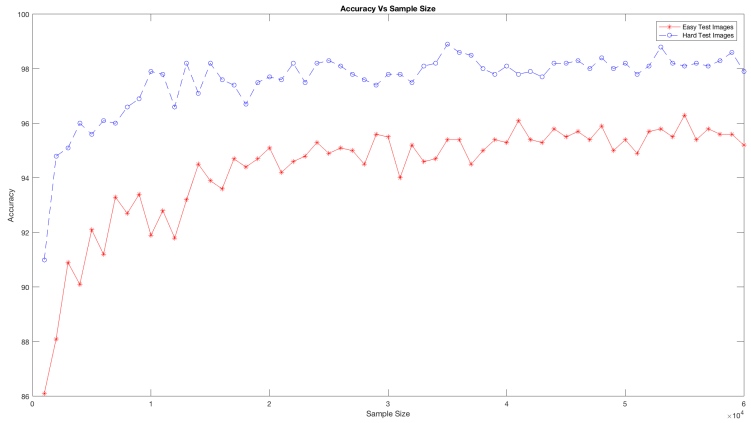

Figure 7: Accuracy vs Train Sample size.

In this experiment, we fix the number of Eigen Vectors (125) needed to represent the data and vary the sample size from [1000-60,000] stepping by 1000 each time. As we can see from the figure 7, the accuracy of the model increases as we increase the sample number , but it doesn't give much boost when we increase our train data from [3000-6000]. This suggests that increasing the train data doesn't always guarantee the increase in the accuracy.

3.3 Performance variation with Number of Eigen Vectors

In the previous experiments 3.2, we have seen that if we fixed our Sample size to 260 for creating the Eigen Vectors and train image to 35,000, we can achieve an accuracy of 98.8%. But do we really need all 260 Eigen Vectors to achieve this high accuracy? Can we use lesser number of Eigen Vectors for projection? To answer the question, we vary the number of Eigen Vectors from [1-260] and plot the accuracy curve as shown in figure 8.

From the plot 8, we can see that accuracy increases as we increase the number of Eigen Vectors but it becomes saturated in the range of [50-300]. We didn't see much increase in the accuracy of the algorithm and if we select some 27 Eigen Vectors, we still get accuracy of 98.6%. This shows that we don't need all the Eigen Vectors to represent the image, instead we can select lesser number.

3.4 Performance Variation with Number of Neighbors in KNN classifier

In this experiment, we vary the number of the neighbors in KNN classifier by fixing the Eigen Vector (125) and Training Sample size (35000). As, one can

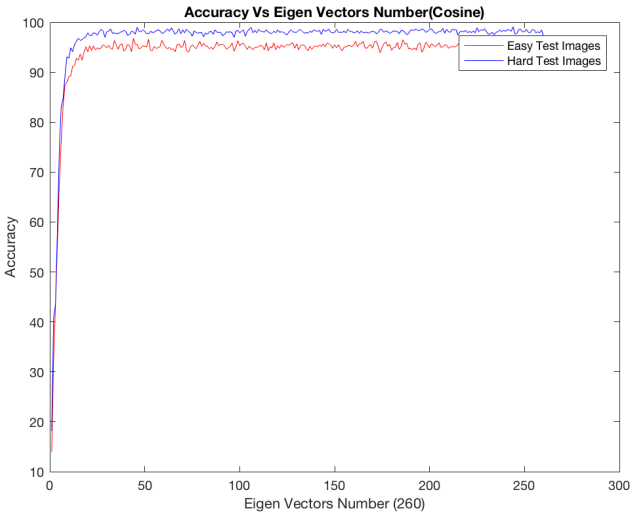

Figure 8: Accuracy vs Eigen Vector Number

see from the figure 9 that accuracy increases to a certain point but it decreases after that. As for the easy test cases, we got the highest accuracy at (k = 9) and for Hard test cases (k = 5). And this is expected because as we increase the neighbor of neighbors, it may possibly that the points closer to different class boundaries may get mis-classified.

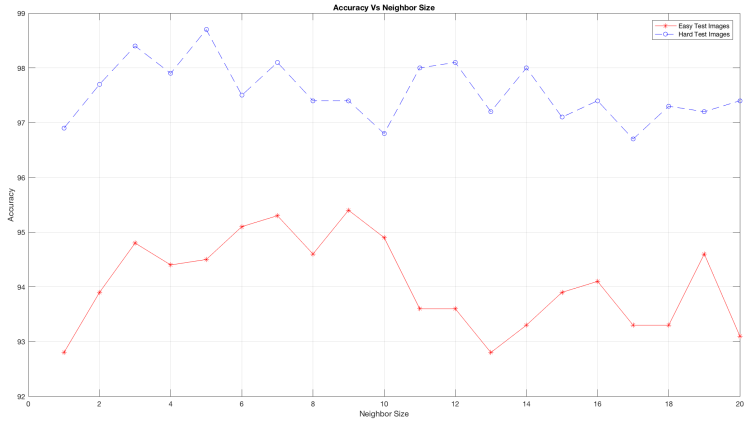

Figure 9: Accuracy vs Nearest Neighbor Number in KNN

3.5 Performance Variation with different metrics in KNN

The accuracy of the model depends on the metric used in the calculation of distance between different examples. In this experiment, we calculate the accuracy for the model by choosing 'cosine' and 'Euclidean' distance metric and varying

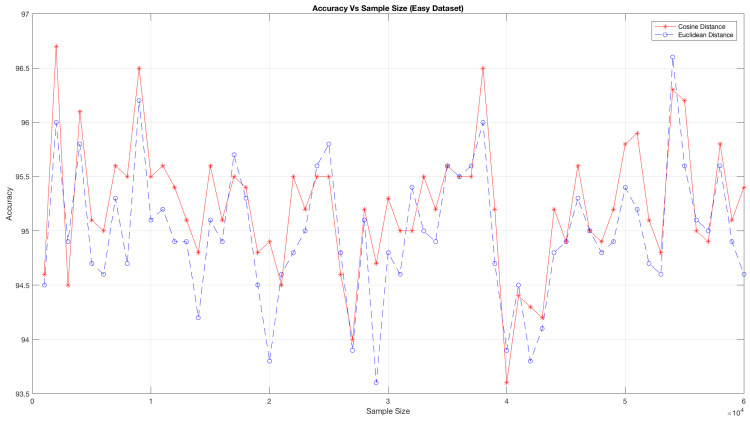

Figure 10: Accuracy vs Sample size for easy dataset using cosine and euclidean distance metric.

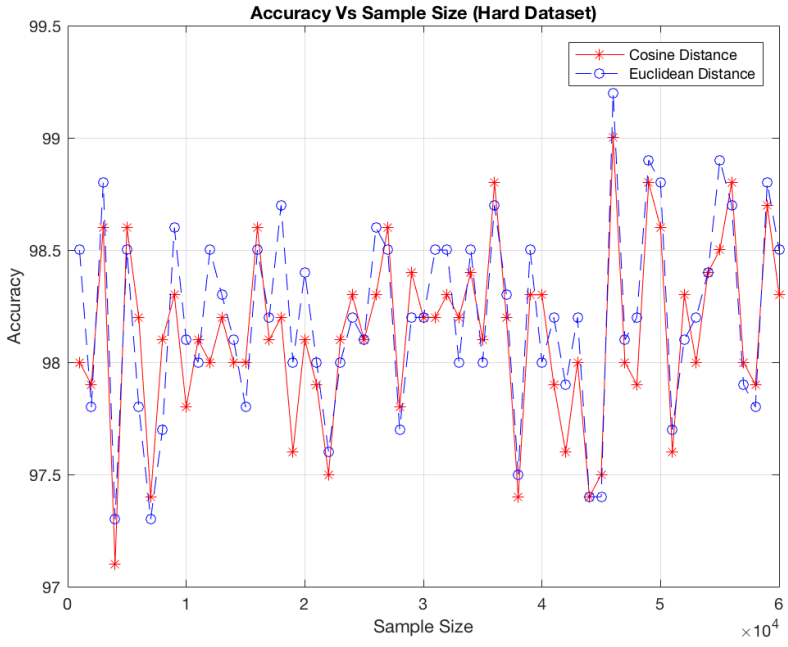

Figure 11: Accuracy vs Sample size for hard dataset using cosine and euclidean distance metric.

the sample size. From the figure 10, we can see that cosine metric perform really well in almost all the sample size for the easy test cases.

For the hard test case also, cosine distance performs very well and gives a very good accuracy as compared to some points where the Euclidean distance gives good accuracy. From these two graphs 10 11, we can say that metric plays a significant role in training the classification model and we should choose our metric very carefully.

4 Conclusion

From the experiments, we can say that Principal Component Analysis can be used for dimensionality reduction and can decrease the computational time. All the experiments clearly show that the number of training images, number of Aegean Vectors, number of neighbors in KNN increase the performance upto certain extent but it becomes saturated after it. Also, distance metric also played a significant role in the performance of the algorithm.

5 Discussion

Since our implementation of PCA requires all images to be of fixed size. However, in real life scenario, we can have train and test images from different sources which are not of the same size. Hence, we need to apply image processing techniques to convert them into fixed size images and have to make sure that we won't lose important information.

References

[1] Bishop, Christopher M. (2006). Pattern Recognition and Machine Learning. Springer: New York.