HW1: Eigendigits

Manu Agarwal
UTEID: ma53767

Abstract. In this paper, we explore the Eigenvector decomposition of covariance for analysis (also known as Principal Component Analysis PCA) for hand-written digit recognition. Once we project the images of hand-written digits onto an eigenspace, we use the k-nearest neighbor method for classification of digits.

1 Introduction

This paper tries to accomplish the task of hand-written digit recognition that are supplied to us as images. The images usually have a high dimension and thus are difficult to operate upon. In fact, if we have an image with dimension mxn pixels, then it represents a digit with mn features (assuming every pixel is a feature). For the purposes of this paper, we are given 28x28 dimensional images of hand-written digits. Thus, there are 784 features that describe every digit. Principal Component Analysis (PCA) is a technique that can help us in dimensionality reduction and extracting relevant features. However, it is important to note that PCA is a lossy compression and there might be loss of information during the process. In this project, we use PCA to find the k principal eigenvectors of the covariance matrix of A (A is the matrix containing image vectors as its columns). We then project the test datapoints onto this eigenspace and finally use k-nearest neighbor algorithm to classify the digit on the basis of its k nearest neighbors in the eigenspace.

2 Related work and our approach

Several algorithms have been developed for hand-written digit recognition. The most naive is the Baseline Linear Classifier where every input pixel value contributes to a weighted sum for each output unit. The output unit with the largest sum is used as the class for the input image.

We then have the baseline nearest neighbor classifier where input images are represented as points in a feature space. The k-nearest neighbor classifier is used with Euclidean distance measure to classify the query image. The problem with this approach is the memory requirement and time taken for recognition of digits.

To address the above problem, we use PCA to find the k principal eigenvectors (the ones with the largest eigenvalues). The intuition behind this is that most of the variance is concentrated in these k principal components. In the experiments, we shall see that we are able to classify hand-written digits with good accuracy using these principal components only.

Our algorithm is as follows:

Algorithm 1 Projection of data onto Eigenspace

```
Input: Matrix $A$ containing image vectors as its columns.
Output: Reduced dimensional training data $Z$.
1: Calculate the mean of each feature in the training set $mean\_vec = mean(A)$.
2: Subtract the mean from the matrix $A$. $A\_normalized = A - mean\_vec$
3: Compute the eigenvalues and eigenvectors of the matrix $A^TA$. $[V\mu] = eig(A\_normalized^T A\_normalized)$
4: Calculate the eigenvectors of $\Sigma$. $V\_sigma = A\_normalized.V$;
5: Sort the eigenvectors in descending order of the corresponding eigenvalues and take the first $k$ of them. Let us call the resulting eigenvector matrix as $V\_final$.
6: Normalize $V\_final$ such that $|V\_final_j| = 1$ to get $V\_normalized$.
7: Project the training data onto the eigenspace for training. $Z = V\_normalized^T.A\_normalized$.
```

Algorithm 2 Classification using k-nearest neighbors

```
Input: Reduced dimensional training data $Z$.
Output: Labeling of the test data
1: Project the test data onto the eigenspace. $test\_A = V\_normalized.(test\_A - \mu)$
2: Use Euclidean distance measure to find the distance of the test sample from each of the training sample
3: Assign the label of the majority of the $k$ training samples that are closest to the test sample.
```

3 Experiments

We were supplied with 60000 labelled training samples and 10000 test samples. Out of the 10000 test samples, the last 5000 training samples were cleaner and easier to recognize than the first 5000. We found that using 500-1000 samples for training was quite sufficient to achieve significant accuracy while recognizing test samples. Our main motive of this exercise was to show that we can reduce dimensionality of our feature space and still achieve significant levels of accuracy. Hence, we refrained from using more than 784 samples for training purposes.

Figure 1 shows the first 36 training examples and their reconstructed versions using 100 and 500 samples to create eigenvectors. Observe that the reconstructed digits do not look exactly like the originals. This tells us that the eigenvectors thus computed are from a reduced space, none the less, they capture most of the significant information.

Figure 2 shows the first 36 test examples and their reconstructed versions using 100 and 500 training samples to create eigenvectors. We see that the computed eigenvectors perform reasonably well in terms of reconstructing the test samples. As we use more training samples, the reconstructed versions of test samples become clearer.

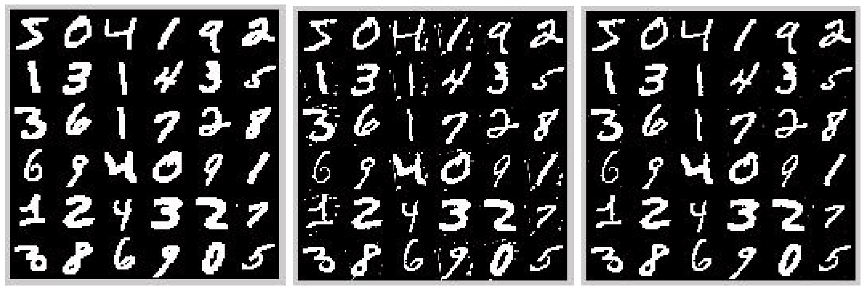

Fig. 1. Left most figure: First 36 training samples, Center figure: Reconstruction of the first 36 training samples using 100 samples to create eigenvectors, Right most figure: Reconstruction of the first 36 training samples using 500 samples to create eigenvectors

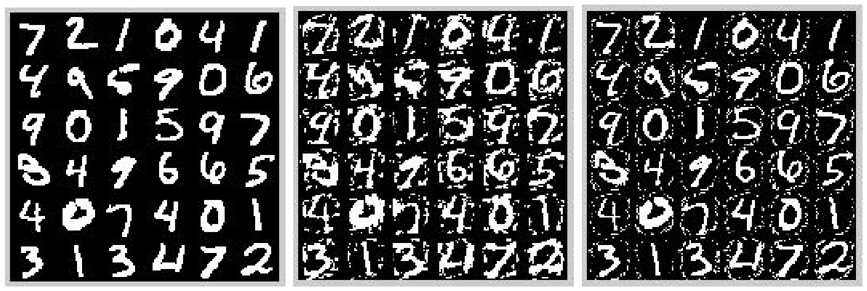

Fig. 2. Left most figure: First 36 test samples, Center figure: Reconstruction of the first 36 test samples using 100 training samples to create eigenvectors, Right most figure: Reconstruction of the first 36 test samples using 500 training samples to create eigenvectors

3.1 Performance as we increase the number of training samples

Figure 3 and Figure 4 represents the increase in accuracy as we increase the number of training samples used. We use 1-nearest neighbor algorithm here. Figure 3 uses the Euclidean distance metric while Figure 4 uses the cosine distance metric.

We observe that the accuracy increases as we increase the number of samples for training. Easy samples reach an accuracy of almost 75% with just 100 training samples. The rate of increase in accuracy diminishes as we increase the number of training samples beyond 400. We are able to achieve an accuracy close to 90% when using 700 samples.

We also observe that the cosine distance metric performs slightly better than the Euclidean distance metric used to compute distances in the k-nearest neighbor classification algorithm.

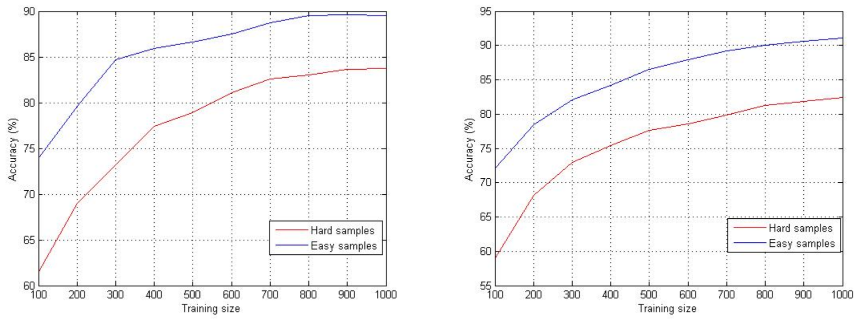

Fig. 3. Increase in performance with increase in the training size. Here we have used Euclidean distance metric while computing distances in k-nearest neighbor algorithm. The first figure uses 1-nearest neighbor and the second figure uses 5-nearest neighbor

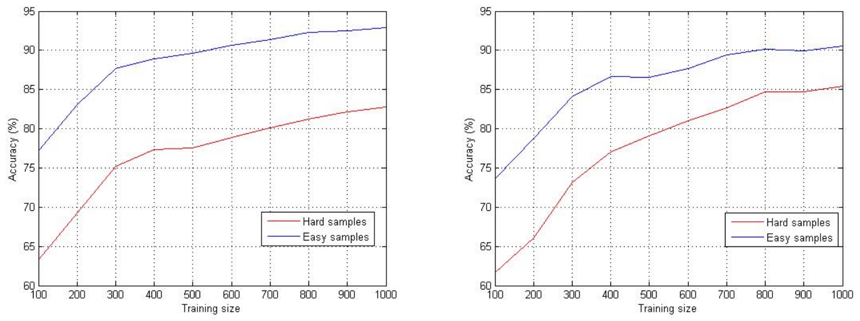

Fig. 4. Increase in performance with increase in the training size. Here we have used cosine distance metric while computing distances in k-nearest neighbor algorithm. The first figure uses 1-nearest neighbor and the second figure uses 5-nearest neighbor

3.2 Accuracy vs k in k-nearest neighbors

Figure 5 shows the variation in the accuracy as a function of k in k-nearest neighbors. We observe that the accuracy goes down as we increase the value of k. The best performance is shown by 1 and 2-nearest neighbors. However, the effect of increasing k becomes less as we increase the number of training samples. This is observable from the second figure in Figure 5 where the curve flattens out (observe the scale on the y-axis is different between the two figures of Figure 5).

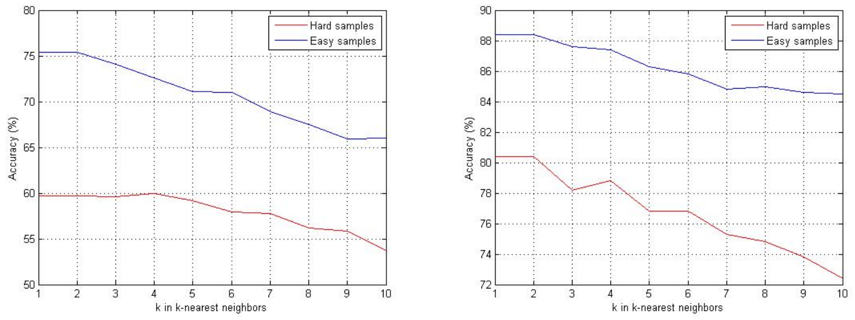

Fig. 5. Change in accuracy as a function of k in k-nearest neighbor algorithm. The first figure uses 100 training samples and the second figure uses 500 training samples

3.3 Accuracy vs the number of top eigen vectors used

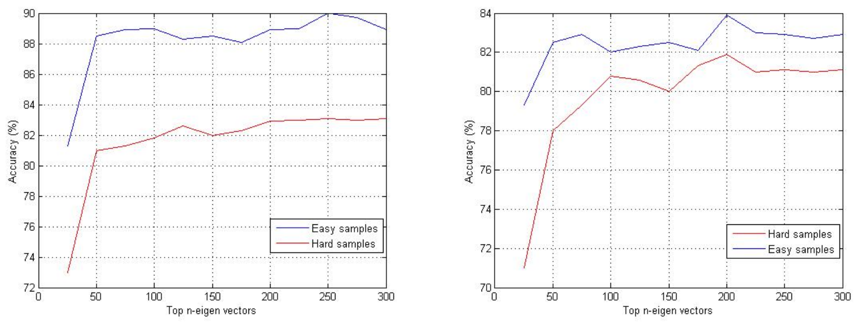

Fig. 6. Change in accuracy as a function of the top n eigenvectors. The first figure uses 1-nearest neighbor and the second figure uses 5-nearest neighbor.

Figure 6 shows the number of top eigen vectors vs accuracy. The first figure uses 1-nearest neighbor algorithm while the second figure uses 5-nearest neighbor algorithm. We see that the performance increases sharply as we increase the number of eigen vectors used from 25 to 50. However, then the performance saturates. It shows that most of the variance is contained in the top 50 eigen vectors.

4 Conclusion

The above experiments show that PCA is a very effective technique for dimensionality reduction. When using PCA for hand-written digit recognition, most of the variance is captured in the top 50-100 eigenvectors. In fact, using 100 eigenvectors gives an accuracy of almost 90% on the easy test samples. We also tried testing our approach for 8000 training samples which achieved an accuracy of 93.2% on the hard samples and 97.4% on the easy samples.

References

1. Pattern Recognition and Machine Learning by Christopher Bishop
2. Resources on the Machine Learning course homepage
   http://www.cs.utexas.edu/ dana/MLClass/446outline.html