HW-1: EigenDigits

Subhashini Venugopalan (EID: SV8754)
vsubhashini@utexas.edu

Abstract

This report presents the experiments and results of implementing eigenvector decomposition of variance (principal component analysis) and k-nearest neighbour classification for hand written digit recognition.

1 Introduction

This experiment focuses on the task of hand written digit recognition using eigenvector decomposition and k-nearest neighbour classification. Images of handwritten digits have a fairly high dimension i.e. the number features used to represent/define the image is quite high. In this experiment each image is of dimension 28x28 pixels. Assuming each pixel as a feature, there are 784 features/dimensions describing every digit. The main idea of principal component analysis (PCA), and in particular eigenvector decomposition is to reduce the data in a high dimensional space to a point in a lower dimensional space without losing much of it's variance. It is easier and more efficient to work with points in a lower dimension, especially since the data samples are of the order of thousands. In the lower dimension space, algorithms such as the k-nearest neighbours (k-NN) algorithm can be used to recognize the digit of a new image based on points closest to it from the observed samples.

2 Dimensionality reduction and Algorithms

A Naive Approach to classify a set of data samples using k-NN would be to consider the distance of the vectors in the original space. This can be accomplished by first obtaining the eigenvectors in the covariance matrix of the observed samples. Then, for each new data point that needs to be classified, one would project the point into this eigenspace and assign to it, the same label as its nearest neighbours. However, considering the dimensions of the data points (which is 28x28=784 in this case), such an approach would scale very poorly as the observed data samples increase.

Eigenvector Decomposition and principal component analysis (PCA) indicate that it is not necessary to use all of the eigenvectors to obtain a good classification; most of the variance is contained in a small fraction of the eigenvectors and it is in fact sufficient to just consider this small subset in order to classify the data point reasonably well. From the experiments it has been observed that even after discarding most of the eigenvectors (corresponding to low magnitude eigenvalues), the remaining eigenvectors (corresponding to higher magnitude eigenvalues) retain enough variance to classify new samples with good accuracy.

The eigenvector decomposition algorithm and k-nearest neighbours algorithm are presented below. For clarity of presentation, bold letter in lower case (e.g. $\mathbf{v}$) is used to denote a column vector and a bold capital letter (e.g. $\mathbf{A}$) is used to denote a matrix.

[1]Note that this covariance matrix will have much smaller size as compared to that of $\sum \boldsymbol{x} \cdot \boldsymbol{x}^T$

Algorithm 1 Eigenvector Decomposition Algorithm

1: Compute the mean of each feature in the training set: $\boldsymbol{\mu} \leftarrow \text{mean}(\mathbf{A})$
2: Subtract the mean from the actual values: $\mathbf{A}_{norm} \leftarrow \mathbf{A} - \boldsymbol{\mu}$
3: Find the eigenvalues of the covariance matrix of $\mathbf{A}^1$. i.e $[\mathbf{U}\ \boldsymbol{\mu}'] \leftarrow \text{eig}(\mathbf{A}_{norm}^T \cdot \mathbf{A})$
4: Get the eigenvectors of this reduced space: $\mathbf{V} \leftarrow \mathbf{A}_{norm} \cdot \mathbf{U}$
5: Sort the eigenvectors in the order of the magnitudes of the eigenvalues.
6: One can further reduce the dimension by considering only the top-n eigenvectors of this already reduced space. Call the resulting matrix of eigenvectors $\mathbf{V}$.
7: Normalize $\mathbf{V}$ to get $\mathbf{V}_{norm}$ such that $||\boldsymbol{v}_j|| = 1$.
8: Project all the data samples on to the eigenspace given by $\mathbf{V}_{norm}$: $\mathbf{Z} = \mathbf{V}_{norm}^T \cdot (\mathbf{X} - \boldsymbol{\mu})$
9: Output the reduced dimension training data $\mathbf{Z}$ (for classification)

Algorithm 2 k-Nearest Neighbours Classification Algorithm

1: Project each test data sample on to the eigenspace. $\mathbf{Z}_{test} = \mathbf{V}_{norm}^T \cdot (\mathbf{X}_{test} - \boldsymbol{\mu})$
2: In the eigenspace, use euclidean distance to find the distance of the test data sample from each of the training data samples ($\mathbf{Z}$).
3: Assign the label corresponding to the majority of the training samples lying closest to the test sample.

2.1 Projection and Reconstruction

In order to apply the classification algorithm, the training data samples and the test data samples are projected into the eigenvector space. The code to perform the projection is presented below.

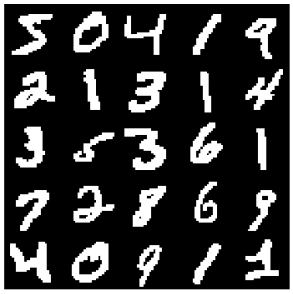
Listing 1: Project Data to Eigenspace

```
function Z = project2Eigen(V, A, mu)
%project (784 x Samples) data to (trainData x Samples) dimension eigenspace

Z = zeros(size(V,2), size(A,2));
A_norm = bsxfun(@minus, A, mu);
Z = V' * A_norm;

end
```

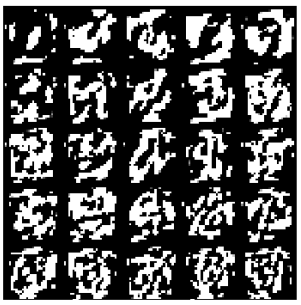
For the purpose of visualization and testing, it is helpful to reconstruct the projected vector back in to the original space. This can help in visualizing how much of the variance the eigenspace retains and also gives an idea of how the k-NN algorithm might perform.

Listing 2: Reconstruct in the original dimension

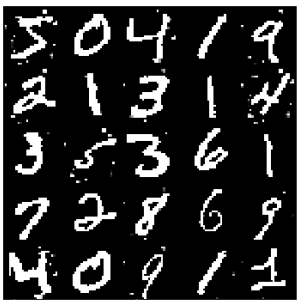
```
function X_rec = reconstruct(Z, V, K)
% Reconstruct an approximate full dimension (784) vector of the given K dimention vector Z

X_rec = zeros(size(V,1),size(Z,2));
X_rec = V * Z;

end
```



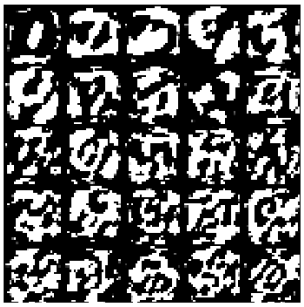
(a) 25 train samples



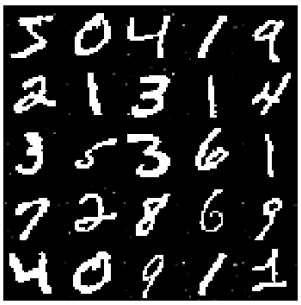
(b) Eigens from 100 samples



(c) Reconstructed from b



(d) Eigens from 500 samples



(e) Reconstructed from d

Figure 1: Eigen Digits and Reconstruction (of 25 images in the training data set)

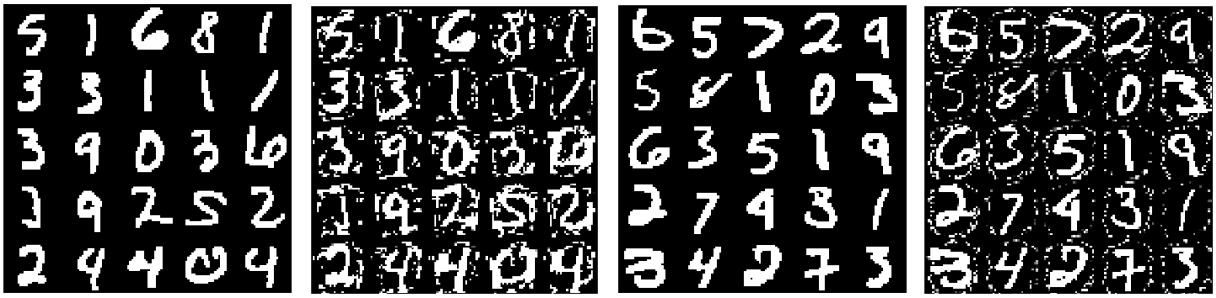
3 Experiments

3.1 Methodology

The data for the experiments used at most 5000 training images each of size 28x28 (giving 784 features for each data point). However, since the main focus of this experiment was to reduce dimension space, it was observed that as few as 1000 samples were sufficient to build a reasonably good classifier. The training data set consisted of 10000 images 5000 of which were identified to be harder to recognize and the remaining 5000 samples were considered easy. The algorithms were implemented in Octave.

Training Visualization. The figure in (1) shows the first 25 training images. It also visualizes the eigenvectors obtained using algorithm (2) on the first 100 and 500 training images. The pictures (c) and (e) show the reconstruction of the projected training sample. A couple of quick observations

- It can be observed that the reconstruction improves with an increase in the training sample.
- More importantly, one can also observe that the reconstruction is not as perfect as the original training sample although the eigenvectors were obtained from the same sample.
- From this, one can infer that the eigenvectors are from a reduced space, however, they still retain most of the variance.



(a) Test-1 25 samples (b) Test-1 reconstructed from 100 eigens (c) Test-2 25 samples (d) Test-2 reconstructed from 500 eigens

Figure 2: Eigen Digits and Reconstruction (of 25 images of the test data set)

Test Visualization. Figure (2) shows images of 25 random training samples which were first projected on to the eigenspace and then reconstructed back on to the original space. The image in (b) shows the reconstruction of a random test sample using eigenvectors obtained by training just 100 samples. Image (d) shows the reconstruction of another random test sample from the eigenvectors obtained by training 500 data points. Here are a couple of observations:

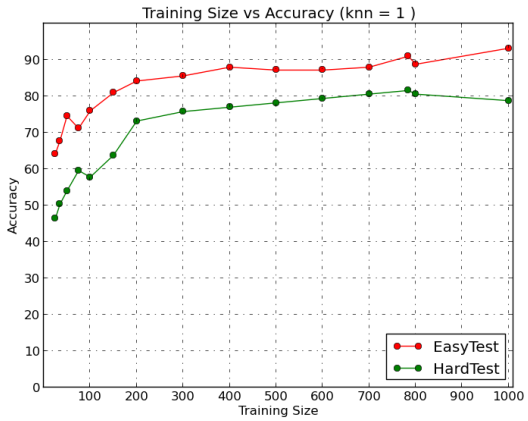
- The same eigenvectors perform fairly well to reconstruct the test data. In figure 1 the reconstruction was very good because the eigenvectors were obtained from the same data sample. This is a new test sample that was never seen in training and the eigenvectors still do a good job of reconstruction.
- Here again, the eigenvectors obtained on training 500 samples gives a better reconstruction as opposed to the one using 100 training samples.

3.2 Results

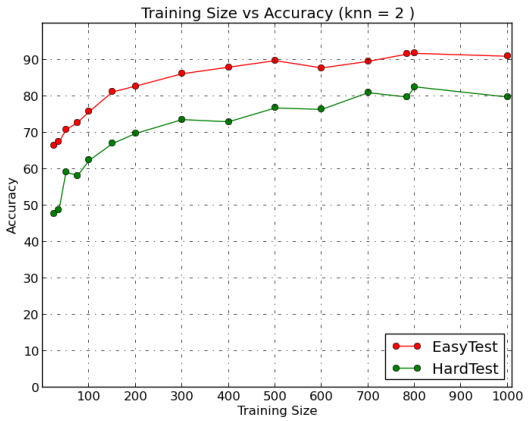
The primary task of the experiment was to test the performance of the algorithm in classifying the digits. Performance is measured in terms of accuracy of classification. After applying the k-NN algorithm, the assigned labels were compared to the actual test samples' labels to determine the accuracy of the entire classifier. Further experiments were performed to test the following aspects:

1. Performance of the classifier with change in number of training data samples. (Train data size vs Accuracy)
2. Classification accuracy when number of nearest neighbours were increased. (Number of nearest neighbours vs Accuracy)
3. Number of eigenvectors necessary to classify accurately. (top-n eigenvectors vs Accuracy)

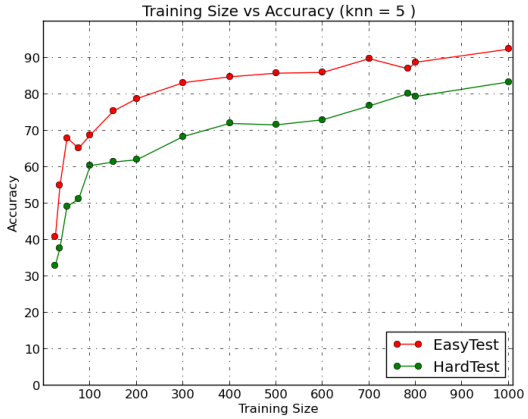
The results of the experiments are demonstrated using some plots. The measurement of classification accuracy in all experiments were obtained by averaging the accuracy values over 10 runs of the experiment. The test samples for each run were picked at random from two sets - labelled "easy" and "hard".



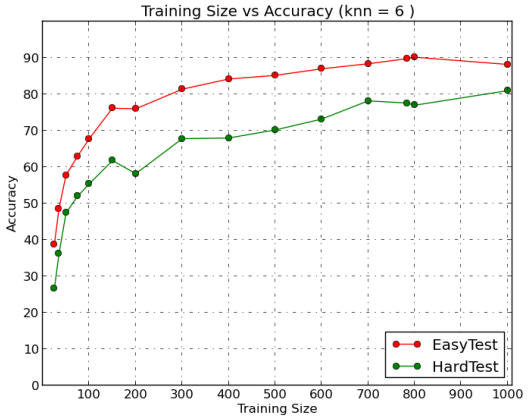
(a) 1 nearest neighbour



(b) 2 nearest neighbours



(c) 5 nearest neighbours



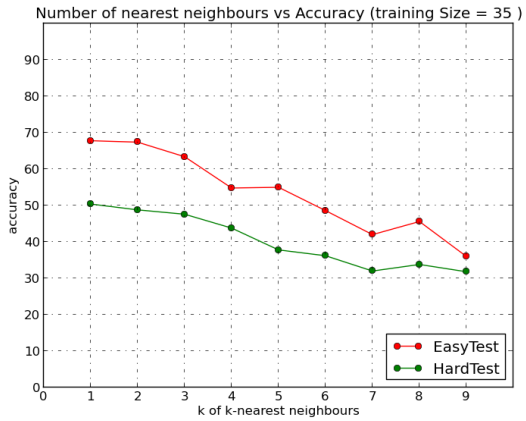
(d) 6 nearest neighbours

Figure 3: Accuracy improves with increased training data independent of k-NN

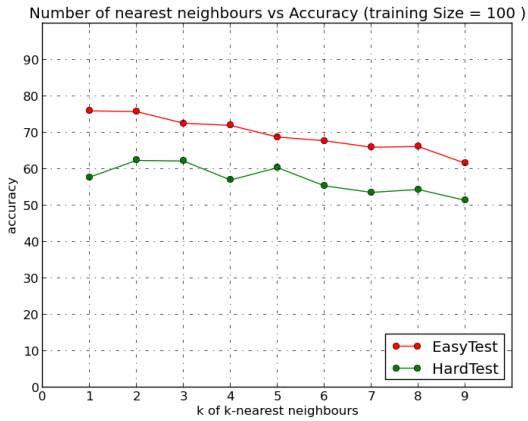
3.2.1 Training Size vs Accuracy

The graphs in figure (3) show that the accuracy improves with increase in training size.

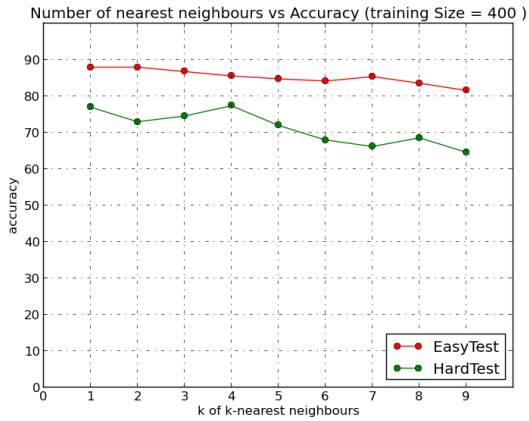
1. With just 25 and 35 training samples, the accuracy of classification is $50\% - 60\%$.
2. When the training data is increased to 400 to 600 samples, the accuracy plateaus to about 88% for the easy set and about 80% for the hard set.
3. When training data is increased to 700 and 1000 samples, the accuracy increases to 92%.
4. On a 5000 sample training set, the accuracy was 96% (on the easy test set) and 91% (on the hard test set). Note: this image has not been plotted, since it was computed in a separate run. This result was obtained on training 5000 samples using 3-nearest neighbours.



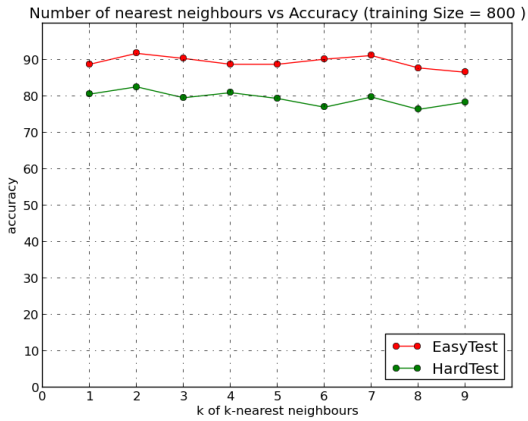
(a) Training Size = 35



(b) Training Size = 100



(c) Training Size = 400



(d) Training Size = 800

Figure 4: Increase in nearest neighbours decreases accuracy (effect is diminished with more training)

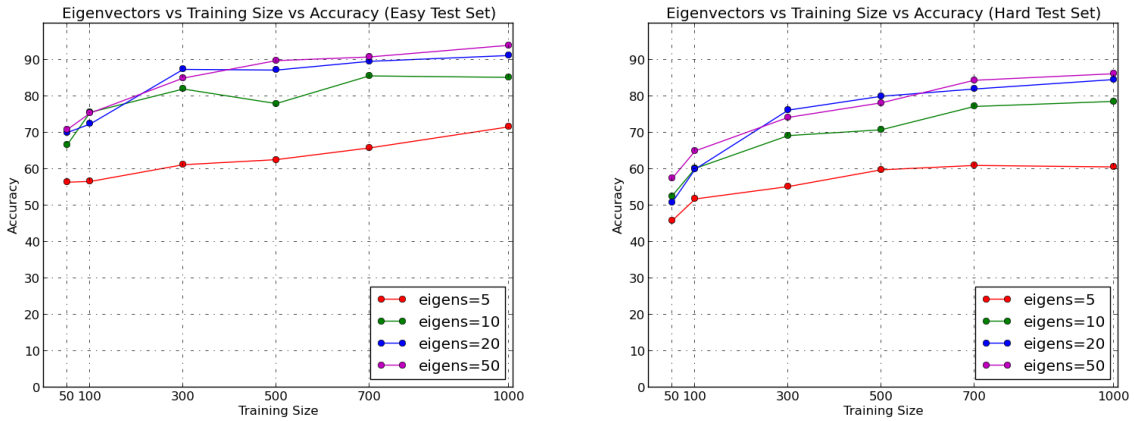
3.2.2 Number of nearest neighbours vs Accuracy

The graphs in figure (4) show that increase in nearest neighbours affects accuracy when training data is small, but does not affect accuracy as much when the training data size grows.

1. It is easy to observe that increasing the number of nearest neighbours used for classification generally reduces the accuracy.

2. In all observations, it can be noted that 1 and 2-nearest neighbours perform best. Increasing to more than that only reduces the accuracy.

3. As the training size is increased the reduction in accuracy of larger (7,8,9) nearest neighbours is diminished to a large extent. This can be observed from the curve in images (c) and (d) when training size is increased to 400 and 800, the curve gets more flatter.



(a) Easy Test Set

(b) Hard Test Set

Figure 5: Increase in nearest neighbours decreases accuracy (effect is diminished with more training)

3.2.3 Variation in number of eigenvectors vs Accuracy

The graphs in figure (5) show that increasing the number of eigenvectors used in the reduced dimension space increases the accuracy.

1. More the eigenvectors used for classification the better is the accuracy.
2. When the training size is reasonably large, one can use fewer eigenvectors.
3. The graph on the easy test set shows that even with as few as 50 eigenvectors an accuracy of 90% can be achieved using just 500 training samples. This accuracy increases to 94% when the training size is increased to 1000.
4. These graphs confirm that the top few eigenvectors retain most of the variance necessary to classify the given images.

4 Conclusion

This experiments confirm that eigenvalue decomposition and principal component analysis are extremely useful techniques in the task of dimensionality reduction for classification. The experiments also conclude that most of the variance is captured by the eigenvectors corresponding to the eigenvalues with high magnitude. As few as 50 (top) eigenvectors are sufficient to classify the images with a high accuracy of 90%. The best performance obtained in this experiment was with 5000 training samples, where an accuracy of 96.0% was achieved on the easy test set and 91.0% on the hard test set.

5 Appendix

Comments.

1. The most inefficient function in the classifier is identifying the k-nearest neighbours. All the other parts of the algorithm are vectorized and can be computed efficiently, and the overall speed of the algorithm can improve greatly if kNN could be made more efficient or replaced by some other classifier.

2. It would have been interesting to check the performance of the classifier on each individual digit (to check if some digits were harder to classify than others). But this deviates from the main focus of the algorithm.

Code To see a sample run of the code, do the following on a terminal:

1. `cd src`

2. `octave eigendigits.m`

The README file explains the contents of other folders.