HW1 Eigendigits Assignment CS391L

Avani Agarwal, EID: aa88539
Department of Computer Science
The University of Texas at Austin
avaniagarwal@utexas.edu

1 INTRODUCTION

The MNIST dataset comprises sixty thousand training images and ten thousand testing images of handwritten digits. Each image has seven hundred and eighty four pixels. For correctly classifying the training data into their respective labels pixel data has to be analysed, thus, making the task computationally expensive due to high dimensionality. One approach to solve this problem is dimensionality reduction. Dimensionality reduction not only lowers the computational cost but in some cases it may help to remove noise [1]. The fewer the dimensions the easier it becomes to interpret the data. According to the curse of dimensionality, a higher number of dimensions requires more training data. There are three ways of dimensionality reduction which are as follows: feature selection, feature derivation and clustering. In this assignment we use a feature derivation method called principal component analysis (PCA) for dimensionality reduction. We also use the K nearest neighbour algorithm for classification.

2 METHODS

Principal Component Analysis (PCA) allows us to transform the data matrix that has high dimensionality into a smaller data matrix that contains the important information extracted from the larger dataset. In PCA, the goal is to reduce the number of variables while preserving necessary information that allows machine learning models to achieve high accuracy. To perform PCA, I first subtracted the mean of the data from data points as the PCA method is sensitive to variance in initial values. Subtracting the mean from data points helps to narrow the range of initial data points. Then the covariance matrix of the dataset is calculated which determines the variation in data points from the mean, that is , variation from each other. We then apply eigenvectors and eigenvalues to calculate principal components.

The principal components calculated by applying transforms to the covariance matrix are uncorrelated. Usually the top n principal components are sufficient to extract maximum information of the dataset. The eigenvectors identify the direction of maximum variance for a high dimensional dataset. The principal components calculated can be thought of as a lower dimensional set of axes that help to observe and evaluate the differences in data points from the most optimal angle possible.By finding a lower dimensional set of axes, we try to find a direction in which the data has the largest variation. These sets of axes are used to create a subspace where the images can be reconstructed using the projected data.

2.1 Implementation

I implemented a function hw1FindEigendigits that returns the eigenvectors and the mean of the data. First I took a subset of the training data samples in matrix **A** with dimensions $\mathbf{x} * \mathbf{k}$ where **x** is the number of pixels in an image which is equal to 784 and **k** is the number of training images. I centred the data by subtracting the mean and stored the results in matrix **B** and then calculated the covariance matrix **C** using the formula

$$\mathbf{C} = \mathbf{B}^\top \mathbf{B}$$

I calculated the eigenvectors V and eigenvalues **D** of covariance matrix **C** by using the formula

$$\mathbf{V}^{-1}\mathbf{C}\mathbf{V} = \mathbf{D}$$

and then proceeded to sort the eigenvalues in descending order to extract the top n eigenvalues for training the data.The function returned the vales of mean and eigenvectors calculated for given **k** images. The mean returned was used to calculate new pixel data vthat would be used for projection.

The value of the n was calculated by considering the amount of eigenvectors required to explain the total variance of the data and by calculating the number of eigenvalues above zero for different amounts of training images **k**. The results are depicted in **section 3**. I then calculated the accuracy for using different amounts of eigenvectors whose values are greater than zero for a particular training image sample (k=1000). Then I calculated the accuracy of the model by using KNN classifier on two test samples of 5000 images each for a k=1000 using top 30 eigenvectors for varying number of neighbours considered and later for varying number of amounts of training samples by using nearest 5 neighbours.

3 RESULTS

Below are the results achieved by using the values returned by hw1FindEigendigits. The eigenvectors are multiplied with the matrix **A** to find the eigen digits. The first ten images of the MNIST dataset and the eigendigits calculated are depicted in **Figure 1**and **Figure 2** respectively.

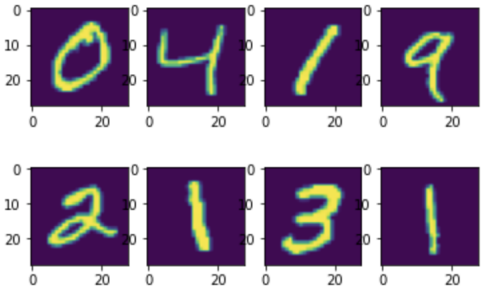

Figure 1: First ten images of MNIST dataset.

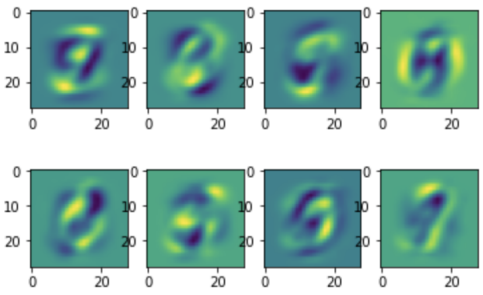

Figure 2: Displaying eigenvectors calculated for 5000 training images.

The reconstructed images using the top seventy eigen vectors are shown in **Figure 4** and their corresponding original images are shown in **Figure 3**.

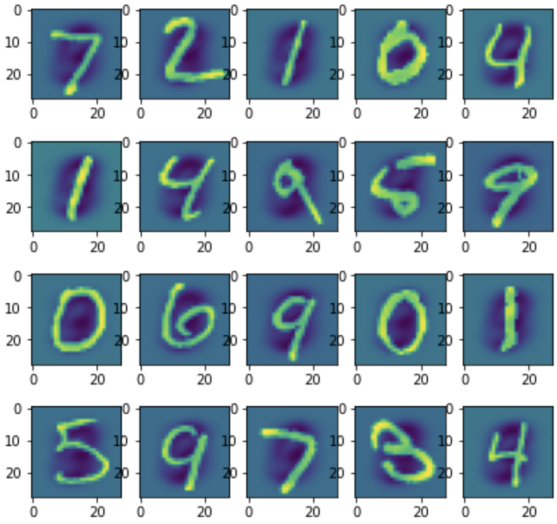

Figure 3: Displaying 20 projections of original images.

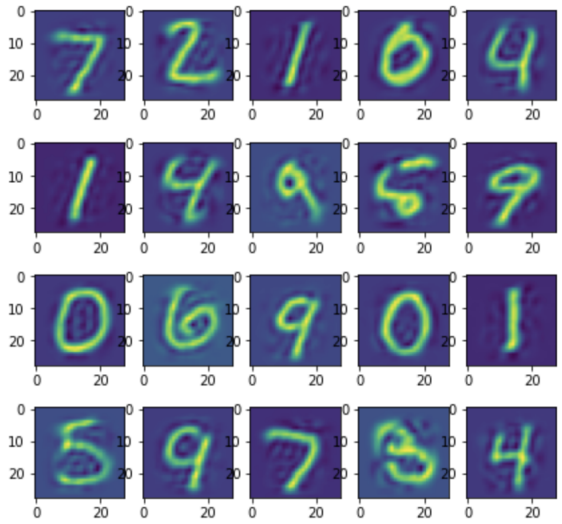

Figure 4: Displaying reconstructed images using projected data.

To optimise the solution, I analysed the number of eigendigits required to explain the total variance and the number of eigendigits with significant values greater than zero for **k** equal to 100, 500, 1000, 2000, 3000 and 5000 by plotiing graphs. The results are depicted in **Figure 5** and **Figure 6** for **k** equal to 1000 and 5000 training images. The code file submitted contains the graphs for other values of k.

For different amount of k we calculate the number of eigenvectors required to expalin the variance i.e. the part of the model's total variance that is explained by factors that are actually present and isn't due to error variance. Also we calculate number of eigenvectors whose values are greater than zero for different amounts of training images (**k**).

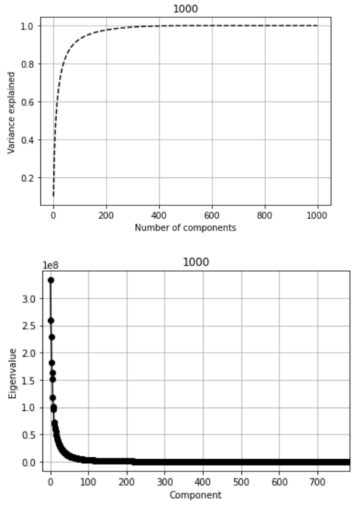

Figure 5: Displaying variance explained vs number of components and eigenvalues vs components for k = 1000.

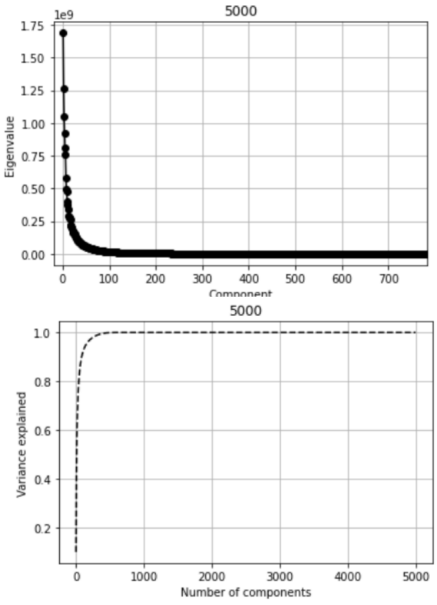

Figure 6: Displaying eigenvalues vs components and variance explained vs number of components for k = 5000.

I then calculated the number of eigenvectors that give maximum accuracy when I use a KNN classifier for **k= 1000** on test set of 5000 images. The results are depicted in **Figure 7.**

```
Accuracy for 10 eigvectors is 0.8974
Accuracy for 20 eigvectors is 0.9446
Accuracy for 30 eigvectors is 0.9529
Accuracy for 40 eigvectors is 0.9515
Accuracy for 50 eigvectors is 0.9515
Accuracy for 60 eigvectors is 0.952
Accuracy for 70 eigvectors is 0.9501
Accuracy for 80 eigvectors is 0.9479
Accuracy for 90 eigvectors is 0.9474
Accuracy for 100 eigvectors is 0.9461
Accuracy for 150 eigvectors is 0.9437
```

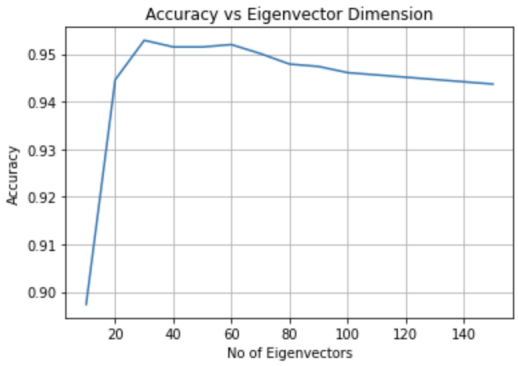

Figure 7: Displaying accuracy with top n eigenvectors for 1000 training samples.

I then proceeded to calculated how many neighbours should be taken in classifier to get optimal solution using k=1000. From explained variance graphs we know that top 30 eigenvectors are needed for maximum accuracy for k=1000. I tested this on two testing datasets of 5000 images each.The results are depicted in **Figure 8.**

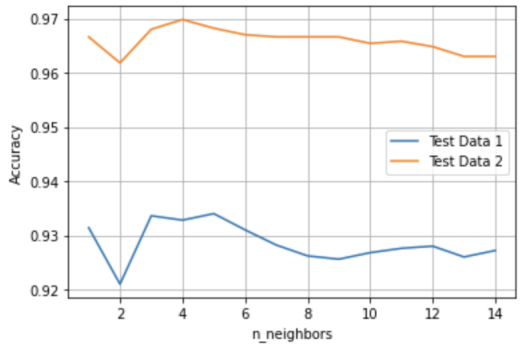

Figure 8: Displaying accuracy vs neighbours for k=1000 and using top 30 eigenvectors.

Afterwards I analysed the accuracy for the kNN classifier for different amounts on training points for k=1000 and the neighbours equal to 5 on two test sets of 5000 images each. The results are depicted in **Figure 9.**

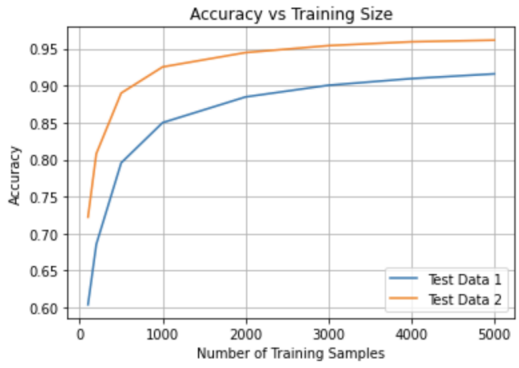

Figure 9: Displaying accuracy vs training points

4 CONCLUSIONS

For training images between 1000 to 5000, in most cases around 100 eigenvectors could be used to explain more than 90 percent of the total variance in the data. During experimentation I found that the number of eigenvectors with eigenvalues with significant value were around 100 for different amounts of training images **k** ranging from 100 to 5000. For **k=1000** kNN classifier gives an accuracy greater than 92 percent for 4 to 8 neighbours and when top 30 eigendgits are under consideration. It gives an accuracy of 97 percent when tested on first 5000 test images for neighbours equal to 4.The assignment also helped me analyse that as the dimensions are increased the images retain more information but a further increase after a certain point for example 30 incase of **k=1000**

actually decreased the accuracy of the model, but accuracy was still above 94 percent. Another key learning was that under PCA we assume that the direction of variation are straight lines which may not always be true [1].

ACKNOWLEDGEMENT

Thanks to Prof. Dana Ballard and Jirau Song (TA) for their ideas.

REFERENCES

[1] Stephen Marsland. *Machine Learning - An Algorithmic Perspective.* Chapman and Hall / CRC machine learning and pattern recognition series. CRC Press, 2009.