Eigendigits Formalism for Classification of MNIST Dataset

Bharath Masetty

bmasetty@utexas.edu

1 Introduction

Principal component analysis (PCA) is a mathematical procedure that converts a set of possibly correlated observations into a set of linearly independent observations using orthogonal transformation. In this report, we are going to appreciate this aspect of PCA using the eigen digits formalism. Eigen values and eigen vectors are evaluated for the MNIST handwritten digits dataset and the raw data is projected onto the selected number of eigen vectors to produce a image vectors of reduced dimension. Then we use k-nearest neighbours classification algorithm to classify the projected image vectors. We analyse the change in accuracy with varying number of training images and varying number of eigen vectors. In the end, we compare the classification accuracy with and without PCA to further understand the use of this method.

2 Methods

2.1 Principal Component Analysis

Dealing with high dimensional data often poses problem related to space and time complexities. Principle component analysis is essentially a **feature extraction** method used to reduce the dimensionality of the data and enables us to consider only the features of relatively higher significance. Consider k samples of data distributed in an n-dimensional space. Then construct an n-dimensional ellipsoid that covers all the k samples in our distribution. The largest axis of this ellipsoid is the direction which has the largest variance and vice-versa. By identifying the size of all the axes of this ellipsoid, we get the information about the distribution of our samples in the direction of each axis. The vectors in the direction of the axes are called the **eigen vectors** and the magnitude of each axis is called the **eigen**

value. The eigen vectors are nothing but the feature vectors and the eigen value is quantity that represents the importance of the corresponding feature or eigen vector. In this way, by choosing the top M feature vectors and projecting our n-dimensional data onto these M vectors, we will be able to transform the data into low dimensional space without loosing too much information. Following is the step by step procedure for executing PCA for the MNIST image data.

PCA Algorithm for MNIST

1. Using the training data T of size n x k, calculate the mean image vector (n x 1).
2. Normalize T' the training data by subtracting the mean from T to get a distribution of size n x k.
3. Calculate the co-variance matrix Σ= T'T'^{t} / (k-1).
4. Evaluate the eigen values and eigen vectors of Σ.
5. Sort the eigen values and eigen vectors in descending order of the magnitude of the eigen values to get V.
6. Reduce V to decided dimensions by truncating it to the accordingly. Let's call it $V_{reduced.}$
7. $V_{reduced}$ is the new basis of our system to project the data.

Projecting & Reconstructing Images

1. Let assume we choose top 50 eigen vectors. Then the shape of $V_{reduced}$ is 784 x 50.
2. Obtain projected images U (50 x 10000) by $V_{reduced}{}^{t}$T'.
3. Reconstruct images U_{rec} of size (784 x 10000) by $V_{reduced}$U.
4. Reshape each column of U_{rec} to 28 x 28 and plot the image.

2.2 K-Nearest Neighbours Classification

The governing principle of the KNN algorithm is that similar things are closer to each other. This is a supervised learning methodology which is both used for classification and regression. In this report, we use this method to classify the MNIST images. Following is the step by step implementation of the KNN algorithm.

KNN Algorithm

1. Load the training data and initialize the number of neighbours (k).
2. For each point in the testing data, calculate its distance to all points in the training data and arrange them in increasing order of distances.

Note: A lot metrics can be used to calculate the distance, in this report, we used euclidean distance.

3. Select the k closest (first k) neighbouring training points and get their labels.

4. Return the mode of the k-training labels observed.

The above algorithm involves calculating distance from the testing point to every training points. The time and space complexity of this calculation is linearly dependent on the dimensions of the data points. For this reason, using PCA to reduce the dimensionality of the data would result in a faster classification.

3 Results

Figure 1 shows the variation in accuracy when the raw data is classified using the K-nearest neighbours classification algorithm with K changing from 1 to 10. Note that we are using the raw data in this case where each image is a vector of 784 dimensions.

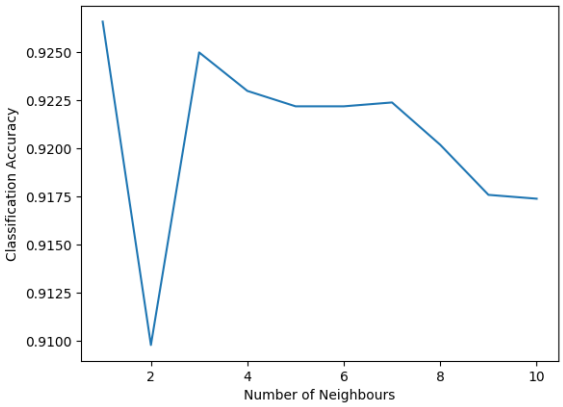

Figure 1: Training Size = 10000, Testing Size = 5000

Now, we implement the principal component analysis (PCA) to obtain the Eigen values and Eigen vectors of the training data. From the training data of size 784 x 10000, we obtain a co-variance matrix of size 784 x 784. Using this co-variance matrix, we can obtain 784 Eigen values and 784 Eigen vectors. Figure 2 shows the first 20 raw images from the test set, Figure 3 shows the first 20 Eigen Vectors and Figure 4 shows the first 20 test images which are projected and reconstructed using first 50 Eigen vectors. Upon projecting a given image onto a set of eigen vectors, we are reducing the dimensions of the image. Even if we choose only the first 50 out of 784 eigen vectors, it can be seen from figure 4 that the important information in the projected images is not lost.

We next compare the classification accuracy of the projected images for different condi-

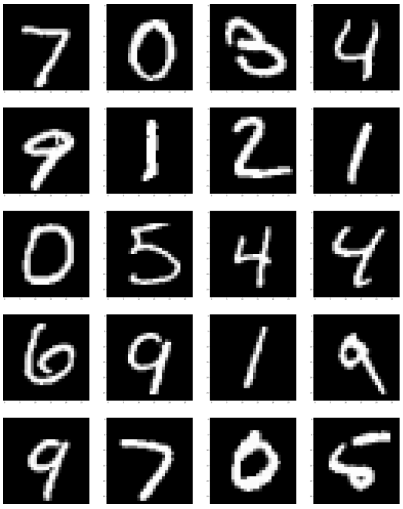

Figure 2: Raw Test Images

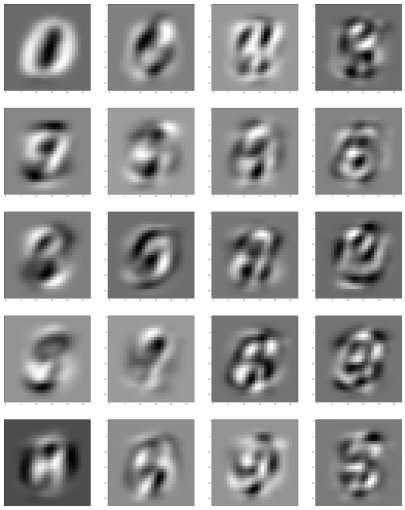

Figure 3: Eigen Vectors

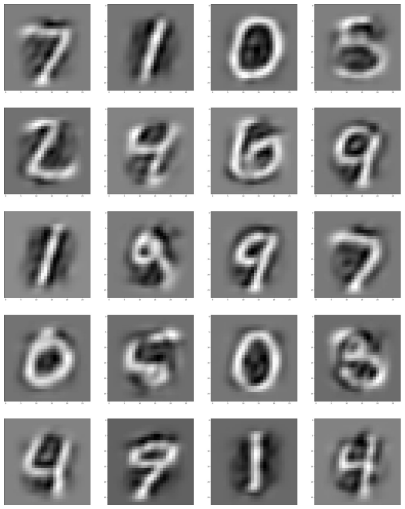

Figure 4: Reconstructed Images

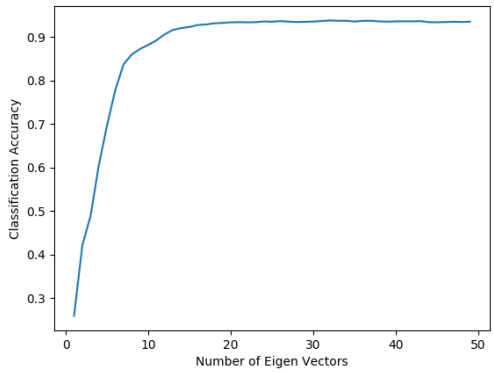

Figure 5: Training Size=10000, Testing Size=5000, k=8

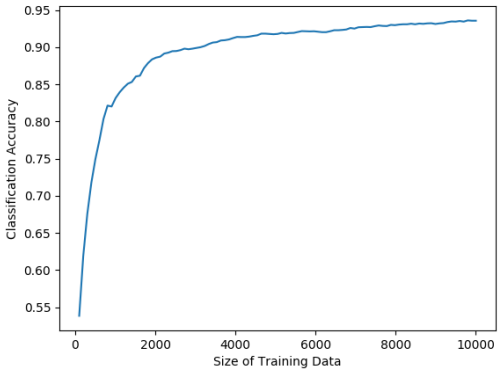

Figure 6: Eigen Vectors=50, Testing Size=5000, k=8

tions. Figure 5 shows the variation of KNN accuracy with varying number of eigen vectors. As expected, the accuracy remains fairly constant after 20 eigen vectors. This means that most of the information necessary to classify the image can be captured by the first 20 eigen vector features. Figure 6 shows the variation of KNN accuracy training size. In this case, the accuracy seems to increase with increasing training size and due to time complexity, the size of the training set in all the test cases is chosen to be 10000.

Finally, to further appreciate the elegance of PCA, an comparison of **variation of accuracy** and **computation time** against **number of KNN-neighbours** is presented in Figure 7. The size of the training and testing sets is the same as in Figure 5. It can be

observed that there is a **noticeable increase** in classification accuracy and a **drastic decrease** in computation time when PCA is used. This result, in one shot explains all the advantages offered the PCA. In either case, the computation time does not vary with increasing neighbours because the number of euclidean distance calculation depend on the size of the training set and dimensions of the data, but not on the number of neighbours.

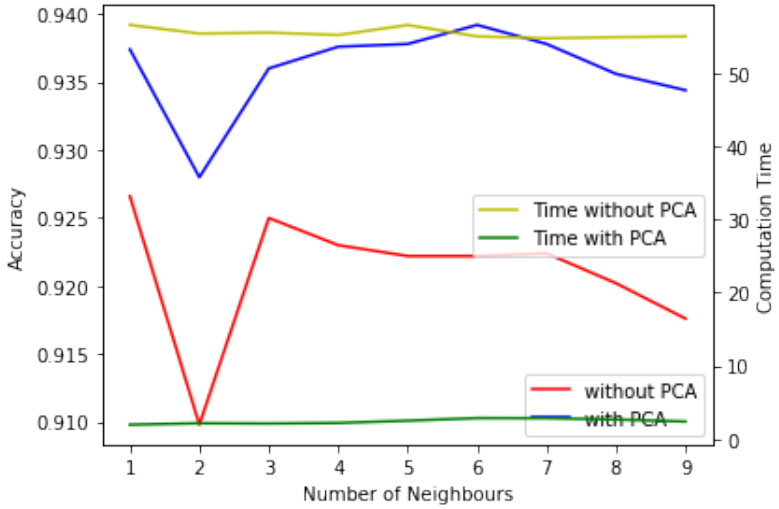

Figure 7: Effect of Accuracy and Computation time

4 Conclusion

In this report, we analysed the importance of dimensionality reduction using principal component analysis to produce a set of linearly independent feature vectors. We have shown how important information of the data can be elegantly maintained while reducing its dimensionality by projecting the data onto the first few set of eigen vectors. We have also seen an almost asymptotic increment of accuracy with increasing training size and increasing eigen vectors. Finally, we observed how PCA is useful to drastically reduce the computation time while increasing the classification accuracy through the comparison shown in Figure 7.

5 References

1. Blog post on KNN.
2. Blog post on PCA.
3. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.