Predicting Hand-written Digits using Principal Components Analysis (PCA)

Yue WU

CS 391L

2019

February

1 Introduction

This homework aims at using principal components analysis (PCA) to detect hand-written digits automatically. We will use a certain amount of training images included in a training set of 60000 images to derive an algorithm that can detect the digits shown in a test set of 10000 images. x denotes the number of pixels in each images. In this case, x equals to 28*28, which is 784.

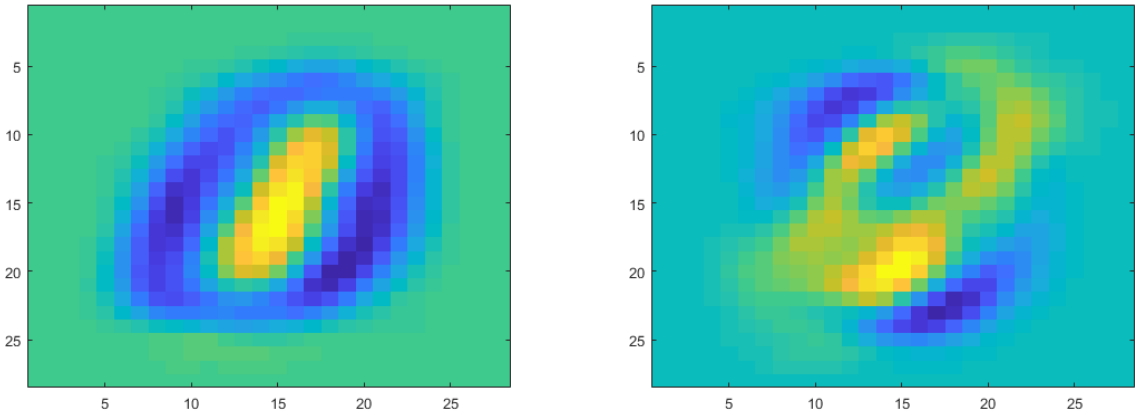
2 Methods

In the implementation, we first calculated the covariance matrix C (x-x or k-k depending on which one is smaller) and mean value vector m (with length x) using k training images stored in matrix A (x-k) (Equation 1), and then we find the eigendigits of the covariance matrix C and store the eigenvectors in matrix V by sorting their corresponding eigenvalues in a descending order. The process is shown in the code namely "hw1FindEigendigits.m". Note that in order to accelerate the calculation, usually we first calculated the eigendigits of the smaller system, and then transfer the eigendigits into the larger system. In our case here we do not need this step because the sizes of the images are small, so we are already

doing calculations in a small system.

$$C = A * A^T \tag{1}$$

The eigenvectors indicate key features found by analyzing k training images. Showing here in Figure 1are the first and third eigenvectors with yellow denoting positive values and blue denoting negative values. Each image can be seen as a weighted combination of these eigenvectors.



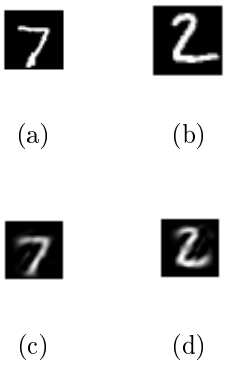
(a) (b)

Figure 1: (a) (b) The first and third eigenvectors with yellow denoting positive values and blue denoting negative values.

With eigenvectors stored in V and the mean vector m calculated using the training set, we were able to project test images into the eigenspace and find out how each image corresponds to the features preserved in the eigenvectors. In order to test if our eigenvectors are working correctly, we displayed the first two images in the test set, projected them to the eigenspace using Equation 2, and then reconstructed the two images using Equation 3. Here we used 2000 training samples and the first 30 eigenvectors. As is depicted in Figure 2, the images reconstructed from the eigenspace are not the same as the original images, but we are still able to tell the main feature of the original images from the reconstruction.

$$I' = transpose(I - m) * V \tag{2}$$

$$I'' = V * transpose(I') + m \tag{3}$$



(a) (b)

(c) (d)

Figure 2: (a) (b) Original test images. (c) (d) Reconstructed images.

After that, we used the Matlab function *fitcknn* to classify training images into ten classes using the Euclidean distance metric. We set the number of neighbors to be 5. By applying this classification algorithm to the test set, we were able to predict content in each test image and calculate how the accuracy of the prediction changes with different numbers of training samples and different numbers of eigenvectors used.

3 Results

By applying numbers of training samples ranging from 1000 to 30000, and applying numbers of eigenvectors ranging from 5 to 100, we calculated the accuracy changing shown in Figure 3. As the number of training samples increases, accuracy increases as well. For the same number of training samples, we observe maximum accuracy when the number of eigenvectors used lies between 25 to 45. The optimal number of eigenvectors used increases with an increase in the number of training samples. The optimal number of eigenvalues used for different numbers of training samples as well as the corresponding accuracy are shown in Table 1.

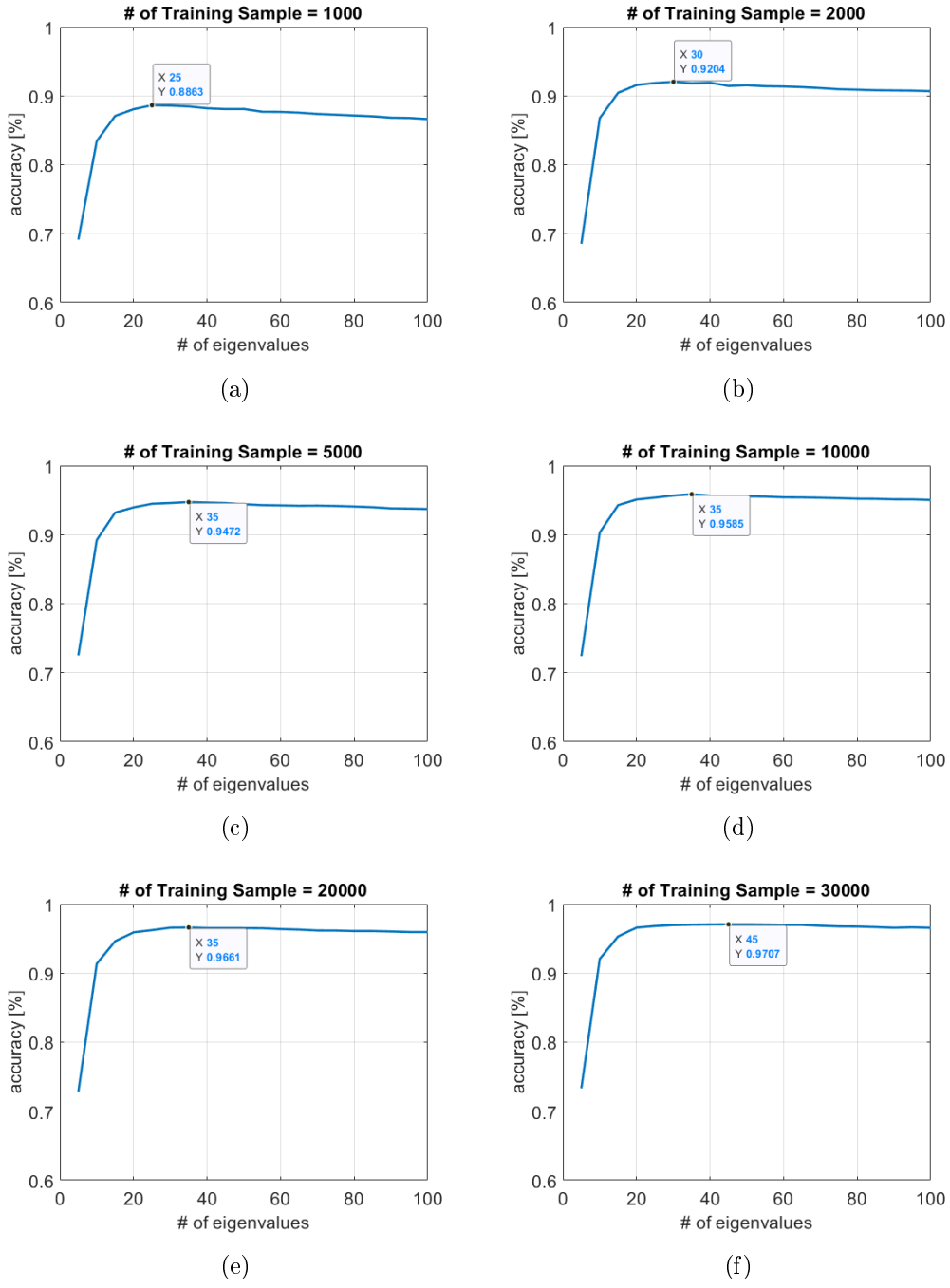


Figure 3: Accuracy changes with different numbers of training samples: (a) 1000 samples, (b) 2000 samples, (c) 5000 samples, (d) 10000 samples, (e) 20000 samples, (f) 30000 samples, and different numbers of eigenvectors used: from 5 to 100.

4 Conclusion

In order to achieve better performance, we should balance between accuracy and the time cost by changing the number of training samples. More training samples make the procedure

		# of Training Sample					
		1000	2000	5000	10000	20000	30000
Optimal # of Eigenvalues	25	0.8863					
	30		0.9204				
	35			0.9472	0.9585	0.9661	
	40						
	45						0.9707

Table 1: The optimal number of eigenvalues used for different numbers of training samples as well as the corresponding accuracy.

slower but can increase the accuracy of the prediction. Once we decide the number of training samples we are going to use, we need to find the optimal number of eigenvectors in order to achieve higher accuracy. We will not have enough features to help us decide which class to put the new image into if we use too little eigenvectors, so, the accuracy can be very low with too little eigenvectors. However, too many eigenvectors will disturb the prediction as well because eigenvectors with small eigenvalues are nearly useless. Therefore, it is critical for us to test and find out the optimal number of eigenvectors to use.