Machine Learning Assignment 2

Huihuang Zheng, huihuang@utexas.edu

hz4674 Spring 2016

1 Using Code

To run my code, open *src/main.m*, change the *DATA_DIR* to where you put your MNIST .mat data, then run the *src/main.m* with Matlab. It will output accuracy with different number of training images and different number of eigenvectors. In addition, my code will write original images and reconstruct images of few eigenvectors into *src/out/* folder. My code was well commented so see them in details.

2 What did I do

I did experiments about how the number of training images and the number of eigenvectors we used can influence the results of PCA digits classification. I will say my steps in this section.

2.1 From disk data file to matrix

In file *src/main.m*, I firstly read data from disk file (see *src/readData.m*). Secondly, since the raw data is $28*28*1$ images and in the training set we have 60000 images of hand writing digits. Due to the requirement of the assignment that training images should be less then data dimension, we should pick few images for training. I implemented two kinds of data picking (see *src/pickData.m*), one is picking first k data. The other is pick random k data. Then, I convert these $28*28*1*k$ 4-d data into $784*k$ 2-d matrix A (see *src/imageFeature.m*).

2.2 Get Eigenvectors and Mean Vector

(See *src/hw1FindEigendigits.m*)Now, since every column of A is a vector representing a image, it's easy to get mean image by just averaging columns. It's also easy to get eigenvalues and eigenvectors by calling Matlab function *eig()*. Suppose A is the matrix after subtracting

mean with size $x * k$ where x is the dimension of each data and k is how many training data we picked. Because $k < x$, the formal way of computing eigenvalues and eigenvectors of AA^T, whose size is $x * x$, will be slower than computing eigenvalues and eigenvectors of A^TA, whose size is $k * k$. And any eigenvector v and corresponding eigenvalue u of A^TA, it satisfies that $A^TAv = uv$. Multiply both sides with A, we got $AA^T(Av) = u(Av)$, which means the Av and u are eigenvector and eigenvalue of AA^T. So in this assignment, I used this way to speed up computing eigenvalues and eigenvectors.

2.3 Test

After we get the eigenvectors and mean vector, we can predict what digit the test image is. I do prediction in the following way. First, I project both training images and testing images into eigenvector space, which means I pick largest n eigenvectors. Those eigenvectors are in a matrix V of size $x*n$. Suppose test images are reshaped as a matrix B of size $x*k'$, then I subtract every column of B with mean vector. After subtracting mean, let $P_B = B'V$. The P_B is the project of B with each row vector is the project of one testing image. Do same thing to training image and I get project of training data P_A. Second, I calculate Euclidean distance between every projected test data point and every projected training data point. For one test data, I predict the image has same digit label as the label of training image with minimum eigenvector-space projected Euclidean distance.

3 Experiment Result

I run experiment from 250 to 600 training data with an increment of 50. Used 10 to 80 eigenvectors with an increment of 10. In this section I will show my experiment result and discuss some results.

Following figure shows the mean image I got from 600 training data. The training data has digits in about center of the image.

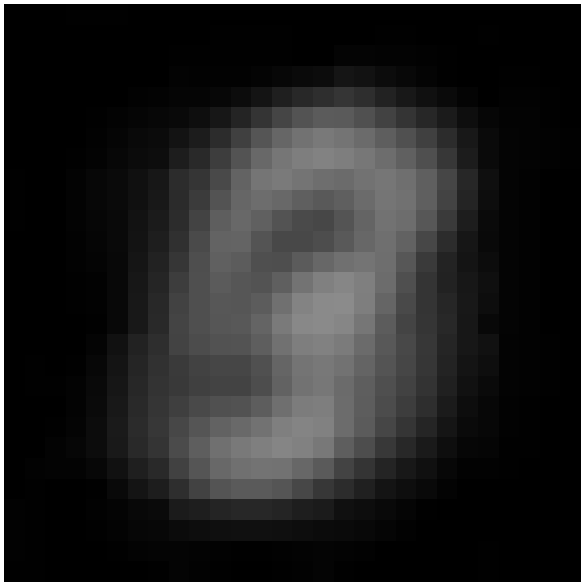

Fig.1 Mean Image

Following figure shows trend of accuracy with number of training images and number of eigenvectors.

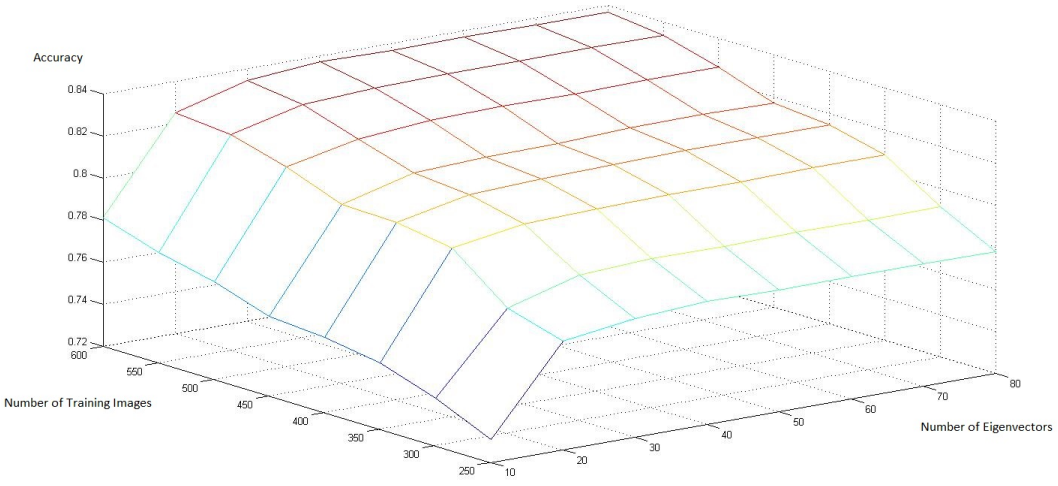

Fig.2 Accuracy Figure

Following table gives detail accuracy of above figure.

Accuracy		Number of Eigenvectors							
		10	20	30	40	50	60	70	80
Number of Training Images	250	0.7309	0.7714	0.7761	0.7783	0.7778	0.7781	0.7782	0.7779
	300	0.7425	0.7793	0.7889	0.7909	0.7904	0.7913	0.7911	0.7913
	350	0.7514	0.8001	0.8054	0.8066	0.8073	0.8074	0.8082	0.8083
	400	0.7557	0.8042	0.8116	0.8129	0.8137	0.8146	0.8146	0.8144
	450	0.7578	0.8052	0.8141	0.8153	0.816	0.8174	0.8177	0.8172
	500	0.7664	0.8149	0.8223	0.8252	0.8257	0.8256	0.8264	0.8262
	550	0.7726	0.8224	0.8307	0.8325	0.8332	0.8334	0.8338	0.8335

Table 1. Detail Accuracy.

We can find even though accuracy increases with number of training data and number of eigenvalues, when the number of eigenvalues is high than 30, it cannot improve accuracy significantly. Using 80 eigenvectors even causes performance worse when with more than 400 training data. So I suggest with 30 eigenvalues can significantly predict what the digit is and with very few computation complexity (versus original dimension of 784 for each image).

Also, I reconstruct images from project space. I show reconstructed images from project space: left column is original image and right column is corresponding reconstruct image from eigenvector space.

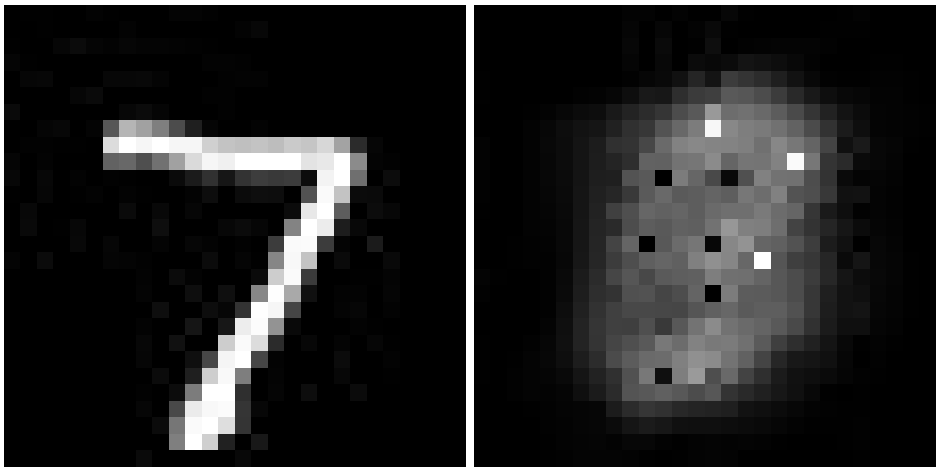

Fig 3. With 200 training data and 10 eigenvectors.

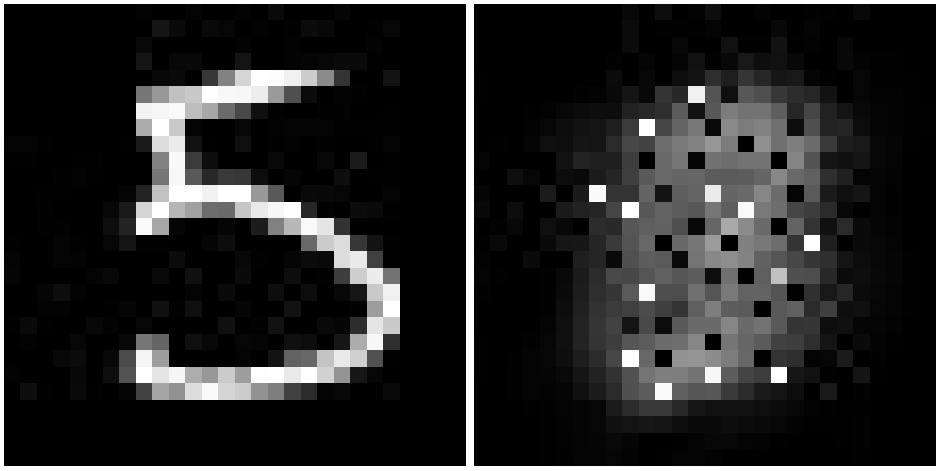

Fig 4. With 200 training data and 40 eigenvectors.

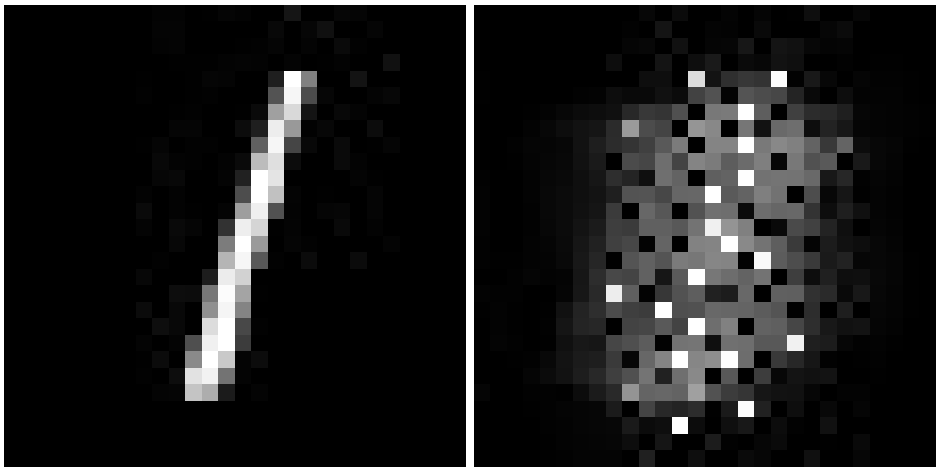

Fig 5. With 200 training data and 80 eigenvectors.

From these images, we can see in the eigenvector space, the image performs in different ways. It seems like blur in images due to add mean back. But we can see some pixels with high intensity and outline the digits.