MAT160 - Final Project

Ahmed H. Mahmoud

June, 11th 2017

Problem No.1

Part A: Our code consists of first computing the left K singular vectors or rank K approximation $U_k^{(j)}$ (j represents the class) using the MATLAB function `svds` for each class $X^{(j)}$ using the training data set. For each test data point y_i, we compare the data point with all computed singular vectors (10 classes) and the inference value of the data point is the class of minimal error. The error of data point y_i w.r.t class j is computed as $E_j y_i = ||y_i - U_k^{(j)}(U_k^{(j)T} y_i)||_2$. Figure 1 shows the first five left singular vectors plotted as images for the first three classes (digits 0, 1 and 2). We notice that the first left singular vector (associated with largest singular value) closely resemble the corresponding digit followed by the second singular vector.

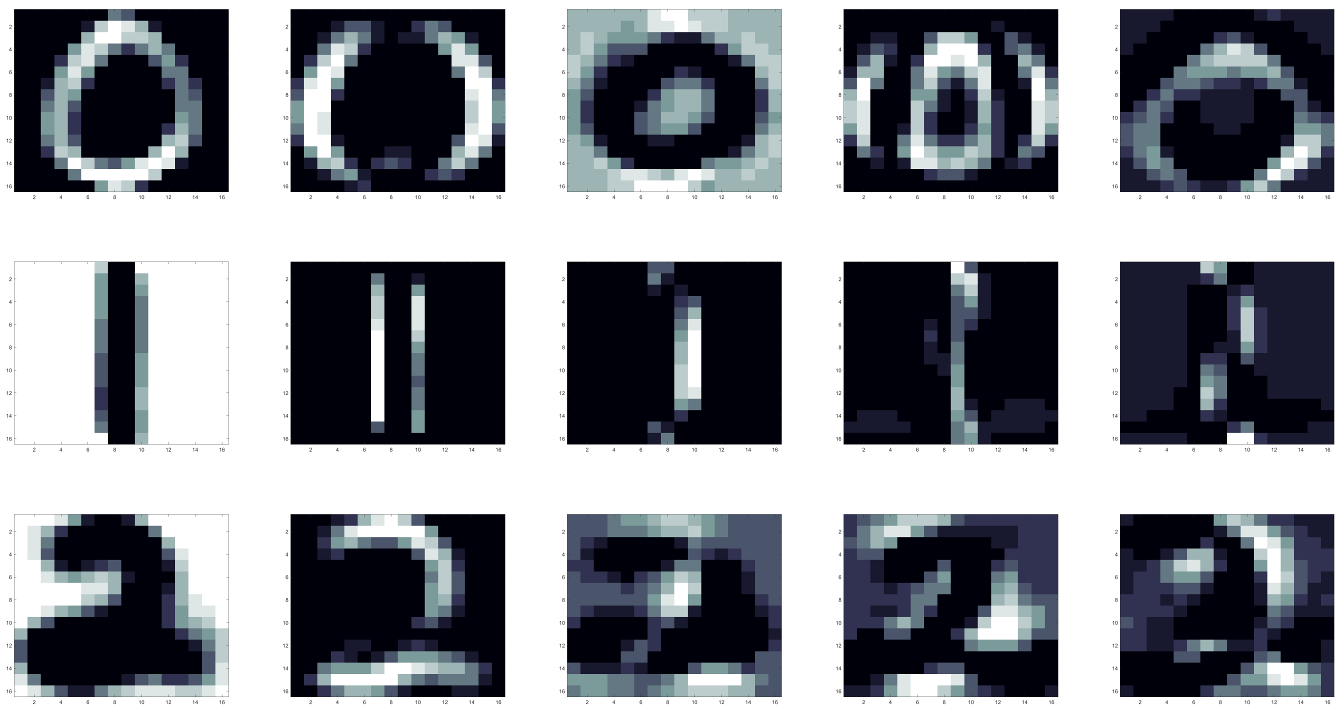
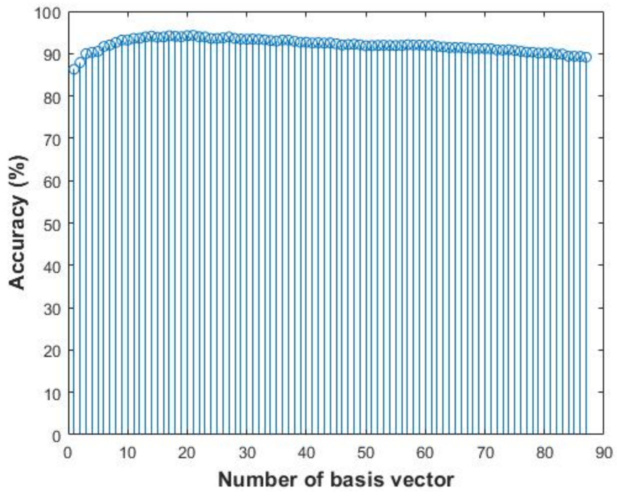


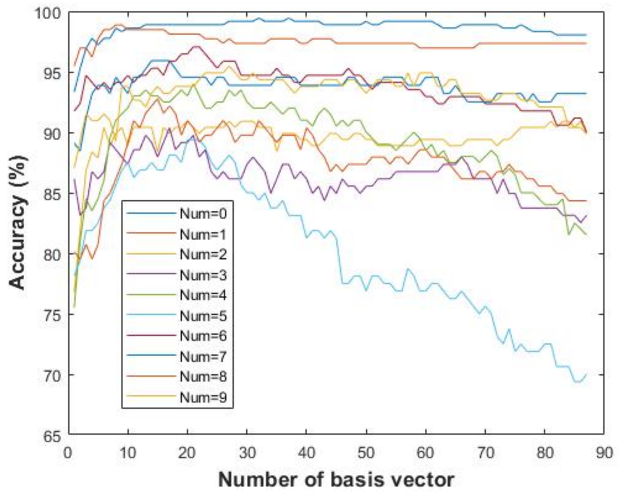
Figure 1: First left (associated with largest singular values) singular vectors for the first three classes.

Part B: We computed the accuracy as a function of the number of basis vector K. The accuracy is computed as the percentage of the number of images that has been classified correctly to all images that has been tested. Figure 2(a) shows the relation between accuracy and the number of basis vector. The maximum accuracy **94.32%** with **22** basis vector. The table to the right show the accuracy values for 5, 10, and 20 basis vector.

# Basis Vector	Accuracy
5	90.3%
10	93.2%
20	93.9 %



(a)



(b)

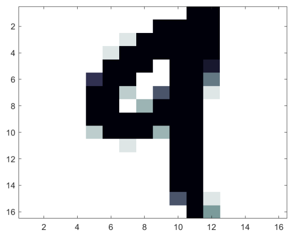
Figure 2: The accuracy of classified images as a function of the number of basis vectors K (a) for all images (b) per class/number.

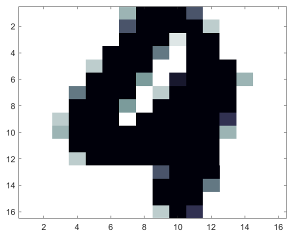
Part C & D: In order to realize if it is beneficial to use different number of basis vector for different classes, we compute per-class accuracy for number of basis vector (from 2 up to 88). Figure 2(b) shows the per-class accuracy for different basis vector. We can see for class of *Number 5*, the accuracy decreases with increasing the number of basis vectors. For other classes like *Number 0* and *Number 1* the accuracy stays same (or marginally decreases) as the number of basis increases. For class of *Number 5*, it is better to use 21 vector basis for which the accuracy is 89.3%. Table 3 shows the maximum accuracy obtained for all class along with the number of basis vector associated with the maximum accuracy. We notice that class of *Number 1* only need 9 basis vector due to symmetry and simplicity of the shape of number one, while *Number 9* need up to 28 basis vector as it is not symmetric and the most complex.

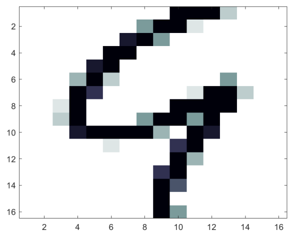
In order to investigate the class of *Number 9* more thoroughly, we draw the few instances for which the number got mis-predicated using 28 basis vector as shown in Figure 4. Most of the mis-predicated images are indeed badly written and looks likes other number like the first row in Figure 4 where the number looks more like *Number 4* more than *Number 9*.

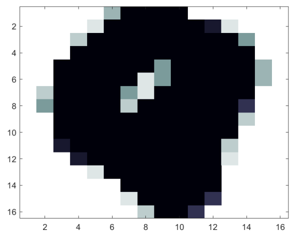
Class	Max Accuracy (%)	# Basis Vector
Number 0	99.4429	33
Number 1	98.8636	9
Number 2	90.9091	21
Number 3	90.3614	18
Number 4	94.0000	22
Number 5	89.3750	21
Number 6	97.0588	22
Number 7	95.9184	15
Number 8	92.7711	16
Number 9	95.4802	28

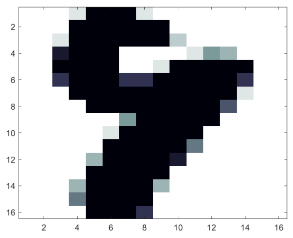
Figure 3: The maximum accuracy obtained per class by varying the number of basis vectors. For each class, we report here the maximum accuracy along with the number of basis vector associated with it.

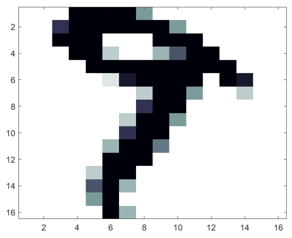

(a) pred=4

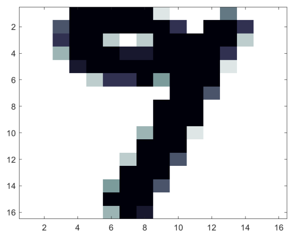

(b) pred=4


(c) pred=4


(d) pred=4


(e) pred=8


(f) pred=8


(g) pred=7

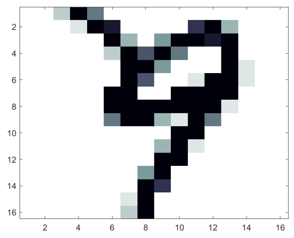

(h) pred=7

Figure 4: Using 28 basis vector where the maximum accuracy is obtained for class of *Number 9*, we draw the few instances where the images is classified incorrectly along with the wrong predication

Problem No.2

Part A: TO DO

Part B: To solve the LP problem, we used `cvx`; MATLAB-based modeling system for convex optimization. `cvx` makes it easy to write and code the problem as a mathematical equations without much conversion compared with using `linprog` within MATLAB Optimization toolbox.

The optimal value we get was **0.49375**. The value represents the average absolute error of the linear model using the provided data set. Since the score (or labels) varies from 0 to 10, the values 0.49375 looks acceptable compared with the range of the scores.

The following shows the parameters of the model

$$a = [\quad 0.083682 \quad -0.84129 \quad -0.23035 \quad 0.061643 \quad -1.608 \quad \cdots\cdots$$

$$\cdots \quad 0.0018835 \quad -0.0027255 \quad -61.517 \quad -0.041654 \quad 1.0888 \quad 0.29696]$$

$$b = 63.1391$$

Part C: We used `cvx` to solve the least-squares regression problem by replacing the 1-norm by 2-norm. The model we get is as follows

$$a = [\quad 0.025086 \quad -1.0835 \quad -0.18256 \quad 0.016374 \quad -1.8741 \quad \cdots\cdots$$

$$\cdots \quad 0.0043605 \quad -0.0032643 \quad -17.9835 \quad -0.41315 \quad 0.91647 \quad 0.2761]$$

$$b = 22.0655$$

The residual sum of squared errors (RSS) is **666.4107**. Note that RSS = $\sum_{i=1}^{n}(y_i - a^T x_i - b)^2$.

Part D: Here we implemented the LASSO model by adding a regularization term to the least-squares regression model. The regularization is $\lambda|q|_1$ where $q \in \mathbb{R}^1 2 = [a; b]$ (i.e., extends the a vector to contain b as well). We experimented with few values for λ in order to reach a value that will turn most of the elements of q to zero except four of them; those that determines the quality of wine. With $\lambda = 0.2$, we found the four features are the top four: *volatile acidity*, *sulphates*, *pH*, and *alcohol*. Our model is as follows

$$a = [\quad 0.063692 \quad -0.92564 \quad -5.3119 \times 10^{-5} \quad 1.2099 \times 10^{-5} \quad -8.1434 \times 10^{-6} \quad \cdots\cdots$$

$$\cdots \quad 0.0044334 \quad -0.0024063 \quad 0.00044866 \quad 0.38007 \quad 0.66368 \quad 0.32965]$$

$$b = 0.5012$$

Note: In order to run the code, `cvx` should be installed on your machine.