ECE M146 Homework 8 Solution
Introduction to Machine Learning
Instructor: Lara Dolecek
TA: Zehui (Alex) Chen, Ruiyi (John) Wu

Please upload your homework to Gradescope by June 8, 11:59 pm.
Please submit a single PDF directly on Gradescope
You may type your homework or scan your handwritten version. Make sure all the work is discernible.

1. In class, you learned that the direction that maximizes the variance of the projection onto a one-dimensional space is the eigenvector corresponds to the largest eigenvalue of the data covariance matrix $S = \frac{1}{N}X^TX$, where $X = \begin{bmatrix} x_1^T - \bar{x}^T \\ \cdots \\ x_n^T - \bar{x}^T \end{bmatrix}$. Formally, the solution to the following maximization problem

$$\max_{u_1} u_1^T S u_1 \quad \text{subject to } \|u_1\|^2 = 1,$$

is the eigenvector corresponds to the largest eigenvalue of S.

In this exercise, we use proof by induction to show that the linear projection onto an M-dimensional subspace that maximizes the variance of the projected data is defined by the M eigenvectors of the data covariance matrix S corresponding to the M largest eigenvalues. Now suppose the result holds for some general value of M and show that it consequently holds for dimensionality $M+1$. To do this, first set the derivative of the variance of the projected data with respect to a vector u_{M+1} defining the new direction in data space equal to zero. This should be done subject to the constraints that u_{M+1} be orthogonal to the existing vectors $u_1, \cdots, u_M$, and also that it be normalized to unit length. Use Lagrange multipliers to enforce these constraints. Then make use of the orthonormality properties of the vectors $u_1, \cdots, u_M$ to show that the new vector u_{M+1} is an eigenvector of S. Finally, show that the variance, i.e., $u_{M+1}^T S u_{M+1}$, is maximized if we choose u_{M+1} to be the eigenvector that corresponds to the $M+1$-st largest eigenvalue λ_{M+1}, assuming the eigenvalues have been ordered in decreasing value.
Solution: We use a Lagrange multiplier λ_{M+1} to enforce the unit norm constraint $u_{M+1}^T u_{M+1} = 1$. We use Lagrange multipliers $\eta_1, \cdots, \eta_M$ to enforce the constraints that u_{M+1} is orthogonal to $u_1, \cdots, u_M$. The Lagrangian is then:

$$L(u_{M+1}, \lambda_{M+1}, \eta_{1,\cdots,M}) = u_{M+1}^T S u_{M+1} + \lambda_{M+1}(1 - u_{M+1}^T u_{M+1}) + \sum_{i=1}^{M} \eta_i u_{M+1}^T u_i).$$

Setting $\nabla_{u_{M+1}} L(u_{M+1}, \lambda_{M+1}, \eta_{1,\cdots,M}) = 0$, we get

$$0 = 2Su_{M+1} - 2\lambda_{M+1}u_{M+1} + \sum_{i=1}^{M} \eta_i u_i.$$

Left multiplying with u_j^T and using the orthogonality constraints, we see that $\eta_j = 0$ for $j = 1, \cdots, M$. We therefore obtain

$$Su_{M+1} = \lambda_{M+1} u_{M+1}.$$

This shows that the new vector u_{M+1} is an eigenvector of S. Left multiply both sides with u_{M+1} and use the normalization constraint, we have

$$\lambda_{M+1} = u_{M+1}^T S u_{M+1}.$$

Then the eigenvector should be the one that corresponds to the $M + 1$-st largest eigenvalue.

2. Suppose you have four data points: $x_1 = [2, 2, 0]^T, x_2 = [0, -2, 2]^T, x_3 = [-2, 0, 0]^T$ and $x_4 = [0, 0, -2]^T$. Use what you learned in PCA to find the 2-dimensional projection of these data points that maximize the sum variance. You should be able to solve this question by hand.

Solution: The data have a mean of $[0, 0, 0]^T$. We first calculate the data covariance matrix:

$$S = \frac{1}{4}\left[\begin{bmatrix} 4 & 4 & 0 \\ 4 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & -4 \\ 0 & -4 & 4 \end{bmatrix} + \begin{bmatrix} 4 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{bmatrix}\right] = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}.$$

We next find the eigenvalues of S to be $2 + \sqrt{2}, 2$ and $2 - \sqrt{2}$ with eigenvectors $[-0.5, -\frac{\sqrt{2}}{2}, 0.5]^T$, $[\frac{\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2}]^T$ and $[-0.5, \frac{\sqrt{2}}{2}, 0.5]^T$, respectively.

We choose the eigenvectors that correspond to the largest 2 eigenvalues to be the subspace we wish to project on. Define s_i to be projected data for x_i. The transformation for x_i can then be written as:

$$s_i = \begin{bmatrix} -0.5 & -\frac{\sqrt{2}}{2} & 0.5 \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{bmatrix} x_i.$$

We finally get

$$s_1 = \begin{bmatrix} -1 - \sqrt{2} \\ \sqrt{2} \end{bmatrix}, s_2 = \begin{bmatrix} 1 + \sqrt{2} \\ \sqrt{2} \end{bmatrix}, s_3 = \begin{bmatrix} 1 \\ -\sqrt{2} \end{bmatrix}, \text{and } s_4 = \begin{bmatrix} -1 \\ -\sqrt{2} \end{bmatrix}.$$

3. One application of PCA is compression. Suppose we want to compress a data vector $x_n \in \mathbf{R}^D$ into M dimensions. We can write the PCA approximation to a data vector x_n in the form:

$$\tilde{x}_n = \sum_{i=1}^{M} (x_n^T u_i) u_i + \sum_{i=M+1}^{D} (\bar{x}^T u_i) u_i$$
$$= \bar{x} + \sum_{i=1}^{M} (x_n^T u_i - \bar{x}^T u_i) u_i,$$

where $\bar{x}$ is the mean vector of $\{x_1, \cdots, x_N\}$ and $\{u_1, \cdots, u_D\}$ are the eigenvectors (corresponding to the largest to smallest eigenvalues) of the data covariance matrix:

$$S = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T.$$

In this exercise, you are given part of the MNIST dataset that has handwritten 4 in it. The data is in the file *MNIST4.csv* which contains a matrix of size 400×784. Each row of the matrix represent an image of size 28×28 where each element represents the intensity of each pixel in gray scale. The 400 images are shown in Figure 1.

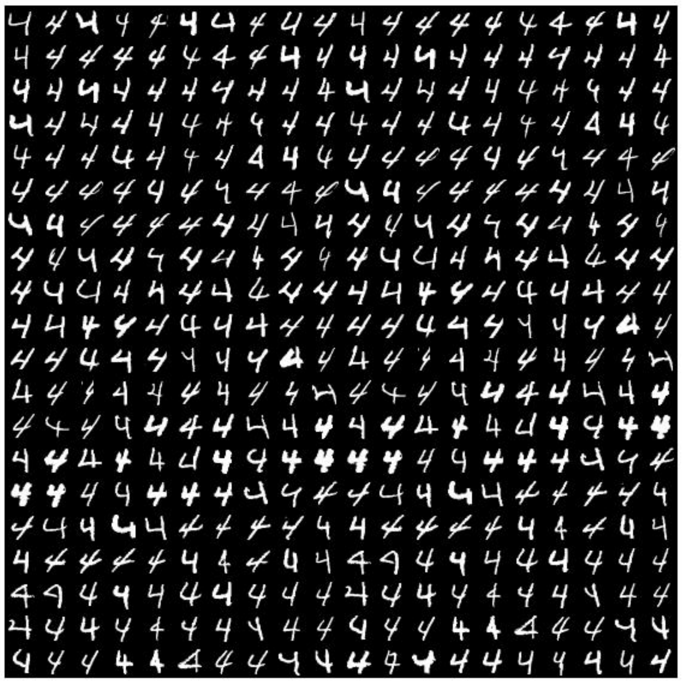
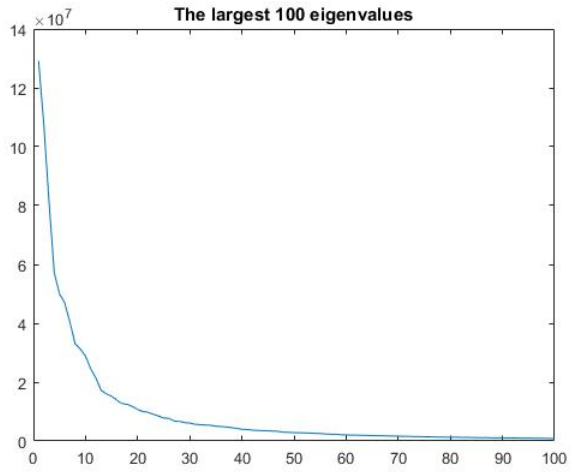


Figure 1: 400 images of 4

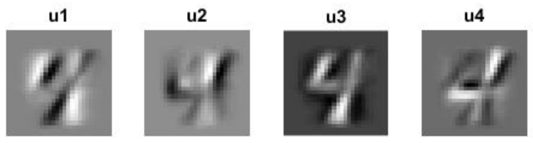
(a) **Eigenvalues** Calculate the eigenvalues and eigenvectors of the data covariance matrix. You may use `eig` in MATLAB and `scipy.linalg.eigh` in Python. Do not use `numpy.linalg.eig` as it will give complex eigenvalues. Plot the 100 largest eigenvalues.

Solution:



(b) **Eigenvectors visualization** Visualize the first 4 eigenvectors by first reshaping the eigenvector into size 28×28 and then showing it as an image using *imshow*. For a better visualization result, scale the range of each eigenvector into $[0-255]$. What do you observe?

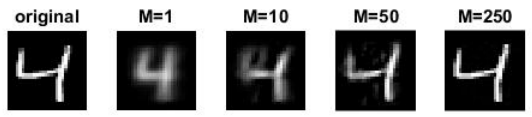
Solution:



The 4 largest eigenvectors have the shape 4 in it. Some difference due to complex eigenvalues are okay.

(c) **Compression using PCA** Compress the first image, i.e., the one on the top left corner, into $M = 1, 10, 50$ and 250 dimensions. Plot the compressed images along with the original image in the same figure. Note that the original image corresponds to $M = 784$. Comment on the quality of the compressed images. What do you get when $M = 0$?

Solution:



The image quality gets better with large M. We notice that the image quality is already pretty good with $M = 15$ and 50. When $M = 0$, the compressed image is just the mean image of the 400 images. Please grade based on the relative quality of the images. For $M = 250$, the result may be different for python user if you use `numpy.linalg.eig`.

4. In this question, we are going to derive results that give us intuition about bagging. Suppose we have N balls in a jar that are numbered $1, \cdots, N$.

   (a) We pick the ball randomly one at a time without replacement. What is the probability that ball 1 is not picked in N realization of this experiment?
   Solution: This probability is 0. Out of the N balls we picked, one ball must be ball 1 because we pick without replacement.

   (b) We pick the ball randomly one at a time with replacement. What is the probability that ball 1 is not picked in N realization of this experiment?
   Solution: $P = \left(1 - \frac{1}{N}\right)^N$.

   (c) For $N = 1000$, verify that the expression you get in (b) is close to $1/e = 0.3679$. Show that probability you get in (b) approaches $\frac{1}{e}$ when $N \to \infty$. Hint: Take the natural log of the limit and then apply the L'Hospital's Rule.
   Solution: We are trying to find the following limit:

$$a = \lim_{N\to\infty} \left(1 - \frac{1}{n}\right)^N .$$

   Taking natural log of this limit, we get

$$\begin{aligned}
\ln a &= \lim_{N\to\infty} N \ln(1 - \frac{1}{N}) \\
&= \lim_{N\to\infty} \frac{\ln(1 - \frac{1}{N})}{\frac{1}{N}} \\
&= \lim_{N\to\infty} \frac{\frac{1}{1-\frac{1}{N}} \cdot \left(\frac{1}{N}\right)^2}{-\left(\frac{1}{N}\right)^2} \\
&= \lim_{N\to\infty} -\frac{1}{1 - \frac{1}{N}} \\
&= -1.
\end{aligned}$$

   We then get $a = \frac{1}{e}$.

5. In class, we learned that the log likelihood function for the Gaussian mixture model is of this form:

$$J = \ln P(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\}.$$

Here, π_k is the prior probability of each Gaussian component; μ_k and Σ_k are the mean and covariance matrix for the k-th Gaussian component.

Suppose we want to maximize J with respect to π_k. Here we must take account of the constraint $\sum_{k=1}^{K} \pi_k = 1$. Use a Lagrange multiplier to enforce this constraint. Show that the π_k that maximize J is of the form:

$$\pi_k = \frac{N_k}{N},$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}),$$

and

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}.$$

You may assume that all $\gamma(z_{nk})$ are known for this step.
Solution: Using a Lagrange multiplier, we maximize the following quantity

$$\ln P(X|\pi, \mu, \Sigma) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right).$$

Taking derivative with respect to π_k and set it equals 0 gives:

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} + \lambda. \tag{1}$$

Multiply both side by π_k and sum over all k. We get

$$0 = \sum_{k=1}^{N} \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} + \lambda \sum_{k=1}^{K} \pi_k.$$

Make use of the constraint $\sum_{k=1}^{K} \pi_k = 1$. We get

$$\lambda = -N.$$

Multiply (1) both side with π_k and substitute λ and the definition of N_k. We find

$$\pi_k = \frac{N_k}{N}.$$

6. In this exercise, you will implement the algorithm on page 167 of *A course in Machine Learning* and use it to perform classification on the data in *AdaBoost_data.csv.*

   (a) **Visualization** The data file contains a matrix in which the 11 rows represent 11 data points. For each row, the first two columns contain the values of x_1 and x_2 and the third column contains the label y for each data point.

   Generate a scatter plot of the dataset where data points from different classes are plotted using different color. Is this dataset linearly separable? Can we use a single layer decision tree to classify all points correctly?

   Solution: Plot is in the figure for $k = 1$ below. The data is not linearly separable. It can not be classified all correctly using a single layer decision tree.

   (b) **Implementation** Consider the decision stump (1 layered decision tree) of the following form as the base classifier for the AdaBoost algorithm.

$$\hat{y} = \text{sign}(s(x_i - t)), s \in \{+1, -1\}, i \in \{1, 2\}, t \in \mathbb{Z}.$$

   The above classifier simply classifies x_i to the right of the threshold t as either $+1$ or -1 based on the sign of either $x_i - t$ or $t - x_i$. For simplicity, in this problem, we restrict t to be integer. The data is designed to avoid the evaluation of sign(0).

   Implement the AdaBoost algorithm on page 167 of *A course in Machine Learning* using the above base classifier for $K = 3$. Use natural log for the log operator in the algorithm and make sure to normalize the weights so that all weights sum to 1. To train the k-th classifier, for $i \in \{1, 2\}$ and $s \in \{+1, -1\}$, search through all integers in the range of x_i exhaustively and find $t^{(k)}$ that minimizes the weighted misclassification error $\hat{\epsilon}^{(k)}$. To avoid exhaustive search for both s and i, we provide the optimal s and i for each iteration as follows: for $k = 1, i = 1, s = -1$; for $k = 2, i = 1, s = -1$; for $k = 3, i = 2, s = 1$. Choose the smaller t in the case that multiple t's give the same weighted misclassification error.

   As a sanity check, you should get $t^{(1)} = 3$ which gives you the first decision stump as:

$$\hat{y} = \text{sign}(3 - x_1).$$

   Run the algorithm and report $d^{(0)}$,$d^{(1)}$ and $d^{(2)}$ in a table. What is your $t^{(k)}$ and $\alpha^{(k)}$ for $k = 1, 2$ and 3? What is the final combined classifier? What is the training accuracy using this combined classifier?

   Plot the data as a scatter plot for $k = 1, 2$ and 3 with the size of each point proportional to $d^{(k-1)}$. Draw the decision boundary of each decision stump.

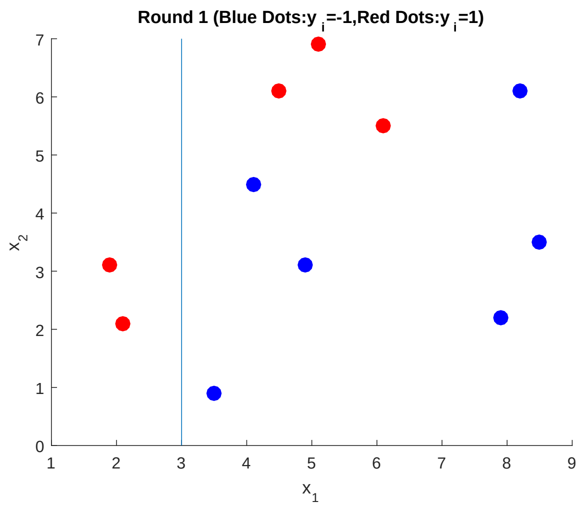
   Solution: The weights are reported in the following table. We get $t^{(1)} = 3$, $t^{(2)} = 7$, $t^{(3)} = 5$, $a^{(1)} = 0.4904$, $a^{(2)} = 0.7332$ and $a^{(3)} = 1.0184$. As a result, the final classifier is:

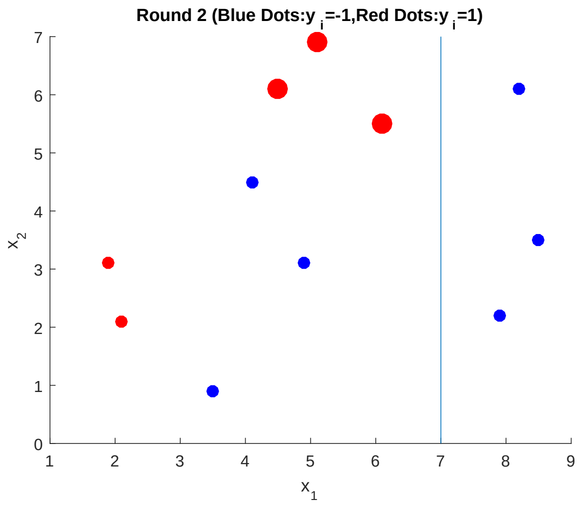
$$\hat{y} = \text{sign}[0.4904 \times \text{sign}(3 - x_1) + 0.7332 \times \text{sign}(7 - x_1) + 1.0184 \times \text{sign}(x_2 - 5)].$$

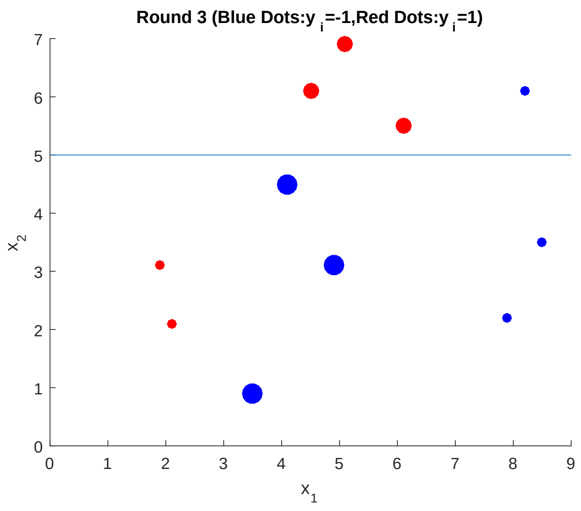
   This classifier classifies all training points correctly.

   The plots are shown in the subsequent figures.

data #	$d^{(0)}$	$d^{(1)}$	$d^{(2)}$
1	0.0909	0.0625	0.0385
2	0.0909	0.0625	0.0385
3	0.0909	0.1667	0.1026
4	0.0909	0.0625	0.1667
5	0.0909	0.0625	0.1667
6	0.0909	0.1667	0.1026
7	0.0909	0.0625	0.1667
8	0.0909	0.1667	0.1026
9	0.0909	0.0625	0.0385
10	0.0909	0.0625	0.0385
11	0.0909	0.0625	0.0385




Round 2 (Blue Dots:y_i=-1,Red Dots:y_i=1)
x_1
x_2
0 1 2 3 4 5 6 7
1 2 3 4 5 6 7 8 9


Round 3 (Blue Dots:y_i=-1,Red Dots:y_i=1)
x_1
x_2
0
1
2
3
4
5
6
7
8
9