Department of Electrical Engineering
School of Science and Engineering

EE514/CS535 Machine Learning

HOMEWORK 1 – SOLUTIONS

Due Date: 23:55, Tuesday, March 08, 2022 (Submit online on LMS)
Format: 6 problems, for a total of 100 marks

Instructions:

- Each student must submit his/her own hand-written assignment, scanned in a single PDF document.
- You are allowed to collaborate with your peers but copying other's solution is strictly prohibited. Anybody found guilty would be subjected to disciplinary action in accordance with the university rules and regulations.
- Note: Vectors are written in lowercase and bold in the homework. For your written submission, kindly use an underline instead. In addition, use capital letters for matrices and lowercase for scalars.

Problem 1 (30 marks)

(**Note:** Compile and submit screenshots of your plots for this question.)

Curse of Dimensionality - In this question we will look closely at the problem associated with higher dimensions using k-NN. During your lectures, you learnt that as the dimensions increase, the neighbours in a constant vicinity decrease. Let us now visualise this phenomenon through simulations.

We are interested in finding the average distance between the two points randomly (uniformly) distributed inside a unit ball. d-dimensional unit ball of radius R is mathematically defined as

$$\mathbb{B}(d) = \{\mathbf{x} \in \mathbb{R}^d, \|\mathbf{x}\|_2 \leq R\},$$

which is referred to as unit ball for $R = 1$.

(a) [**2 marks**] What does the the ball $\mathbb{B}(d)$ represent for $d = 2$ and $d = 3$?

Solution: $\mathbb{B}(d)$ represents a disk of unit radius in 2D for $d = 2$ and a ball of unit radius in 3D for $d = 3$.

(b) [**6 marks**] You are required to build a function that will generate random (uniformly distributed) n number of data points inside the unit ball in d dimensional space. Generate a dataset where *n=50* and *d=100*. Submit your code or attach a screenshot of your code.

Solution:

```
import matplotlib.pyplot as plt
import numpy as np
import math

def points_ball(n,d):
    gauss_points = np.random.normal(size = (n,d))
    length = np.linalg.norm(gauss,axis = 1)
    fact = 1.0/d
    uniform_points = np.random.uniform(0, 1, size=(n, 1)) ** fact
    points = np.multiply(gauss_points, uniform_points/length[:,None])
    return points
```

(c) [**10 marks**] You are required to build a function that will randomly take two points from a $d-$dimensional unit ball and calculate the Euclidean distance for $i = 10000$ iterations. You are given the following starter code:

```
import matplotlib.pyplot as plt
%matplotlib inline
import math
import random

def freq_plot(d_dim, iterations = 10000):
dist = []
# code required here

plt.figure()
plt.hist(dist, range = [0, math.sqrt(d_dim)], bins=100,
density = True)
title = 'n = ' +  str(d_dim)
plt.gca().set(title=title, xlabel='Distances', ylabel='Frequency')
return
```

Use the function developed in part (b) for randomly selecting two points inside the unit ball.
Attach a screenshot of your final working function code.

(d) **[6 marks]** Run the function in part (c) for d_dim $= 2, 3, 5, 10, 100$, and attach screenshots of your plots.

(e) **[6 marks]** Interpret your results by explaining the difference in the plots for each dimension.

Problem 2 (20 marks)

Principal Component Analysis - In class we studied PCA as a means to reduce dimensionality, by minimizing the squared error of predictions in lower dimension, also known as projections onto a range space, as compared to the actual values. In this question, we will prove that this problem is equivalent to maximizing the squared length of these projections, i.e., maximizing variance of projected data.

Let us define the original problem from the class lecture. Considering n feature vectors of the form $\boldsymbol{x} \in \mathbb{R}^d$. Using only k basis vectors, we want to project $\boldsymbol{x}$ in a new space of lower dimensionality, from $\mathbb{R}^d$ to $\mathbb{R}^k$:

$$\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{x},$$

where $\boldsymbol{W}$ is the orthogonal mapping matrix of size $d \times k$. To obtain a approximation of $\boldsymbol{x}$ we use the following reconstruction:

$$\hat{\boldsymbol{x}} = \boldsymbol{W}\boldsymbol{z},$$

which is equivalent to:

$$\hat{\boldsymbol{x}} = \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x}$$
$$\hat{\boldsymbol{x}} = \boldsymbol{P}\boldsymbol{x},$$

where $\boldsymbol{P} = \boldsymbol{W}\boldsymbol{W}^T$ is the projection operator, that is idempotent: has the following properties: $\boldsymbol{P}^2 = \boldsymbol{P} = \boldsymbol{P}^T$.
Our objective is to minimize the sum of squared error during reconstruction, we can write it as follows:

$$\text{minimize} \quad \|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|_2^2$$

(a) **[10 marks]** Show that this expression is minimized when:

$$\boldsymbol{P} = \boldsymbol{W}\boldsymbol{W}^T$$

Solution: Rewriting

$$\|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|_2^2$$

as

$$\boldsymbol{x}^T\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{P}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{P}^T\boldsymbol{P}\boldsymbol{x},$$

and substituting $\boldsymbol{P} = \boldsymbol{W}\boldsymbol{W}^T$ yields

$$\boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{x}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x} = \|\boldsymbol{x}\|_2^2 - \|\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x}\|_2^2,$$

Since $\boldsymbol{W}$, by definition, is comprised of eigenfunctions of the covariance matrix and maximizes the norm of the projected vector $\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x}$, $\|\boldsymbol{x} - \boldsymbol{P}\boldsymbol{x}\|_2^2$ is minimized for the choice of $\boldsymbol{P} = \boldsymbol{W}\boldsymbol{W}^T$.

(b) **[10 marks]** Perform PCA on the following input matrix X to reduce the number of dimensions to $k = 1$, where $n = 6$ and $d = 2$, such that $\boldsymbol{x} \in \mathbb{R}^d$ and $\hat{\boldsymbol{x}} \in \mathbb{R}^k$.

$$\boldsymbol{X} = \begin{bmatrix} 2 & 4 & 5 & 5 & 3 & 2 \\ 2 & 3 & 4 & 5 & 4 & 3 \end{bmatrix}$$

Solution: We first compute sample mean, $\bar{\boldsymbol{x}} = \begin{bmatrix} 3.5 \\ 3.5 \end{bmatrix}$ and subtract it from $\boldsymbol{X}$ to obtain $\boldsymbol{S} = \boldsymbol{X} - \bar{\boldsymbol{x}}$, which can be used to obtain the covariance matrix as

$$\boldsymbol{\Sigma} = \tfrac{1}{n}\boldsymbol{S}\boldsymbol{S}^T = \tfrac{1}{6}\begin{bmatrix} 9.5 & 5.5 \\ 5.5 & 5.5 \end{bmatrix}.$$

The eigenvalue decomposition of $\mathbf{\Sigma}$ as $\mathbf{\Sigma} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$ yields

$$\mathbf{V} = \begin{bmatrix} 0.8191 & -0.5737 \\ 0.5737 & 0.8191 \end{bmatrix}, \qquad \mathbf{D} = \begin{bmatrix} 2.2254 & 0 \\ 0 & 0.2746 \end{bmatrix}$$

Using v_1, we construct $\boldsymbol{W} = [0.81910.5737]$ to obtain

$$\boldsymbol{z} = \boldsymbol{W}X.$$

Problem 3 (10 marks)

Analytical Solution of Least Square - Linear regression can be expressed in Matrix notation:

$$\boldsymbol{y} = \boldsymbol{X}\,\boldsymbol{\theta}$$

Here $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ (where $n > m$) is the input data and each column is a data feature, $\theta \in \mathbb{R}^m$ is a vector of coefficients, and $\boldsymbol{y} \in \mathbb{R}^n$ is a vector of output variables for each row in $\boldsymbol{X}$.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^M \\ 1 & x_2 & x_2^2 & \dots & x_2^M \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^M \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_M \end{bmatrix}$$

This is an over-determined system which results in multiple possible values for the coefficients. A typical way to find a solution is where the values of b in the model minimize the squared error (linear least square solution). Recall that the least square expression is $\|X\boldsymbol{\theta} - \boldsymbol{y}\|_2^2$. In matrix notation, this problem is formulated using the so-called normal equation that can be rearranged to give:

$$\theta = (\boldsymbol{X}^{\mathrm{T}}\,\boldsymbol{X})^{-1}\,\boldsymbol{X}^{\mathrm{T}}\,\mathbf{y}$$

(a) Given the following $\boldsymbol{X}$ and $\boldsymbol{y}$, find the least square solution for $\boldsymbol{\theta}$.
(**Note:** Submit the code and final result for this part)

$$\boldsymbol{X} = \begin{bmatrix} 0.05 \\ 0.18 \\ 0.31 \\ 0.42 \\ 0.5 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} 0.12 \\ 0.22 \\ 0.35 \\ 0.38 \\ 0.49 \end{bmatrix}$$

(b) Draw a scatter plot of $\boldsymbol{y}$ vs $\boldsymbol{X}$. On the same plot, draw a line plot for the model using the value of $\boldsymbol{\theta}$ found in the previous part.
(**Note:** Compile and submit screenshots of your plots for this question.)

Solution:

(a) Code to find least square theta:

```
# direct solution to linear least squares
from numpy import array
from numpy.linalg import inv
from matplotlib import pyplot
# define dataset
data = array([
  [0.05, 0.12],
  [0.18, 0.22],
  [0.31, 0.35],
  [0.42, 0.38],
  [0.5, 0.49]])
# split into inputs and outputs
X, y = data[:,0], data[:,1]
X = X.reshape((len(X), 1))
```

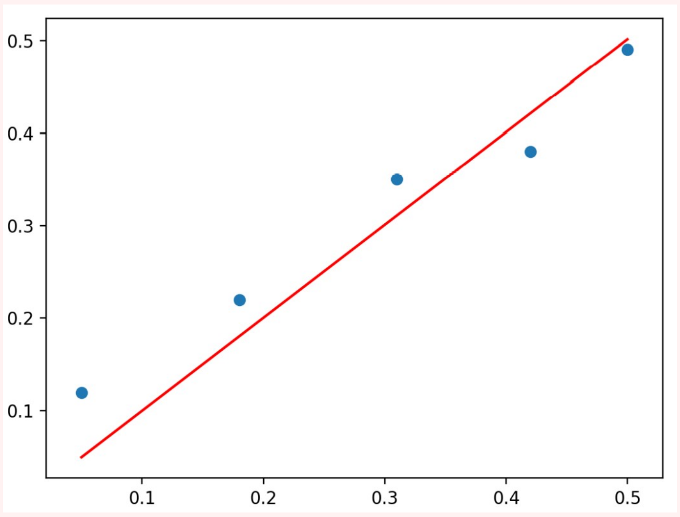
```
# linear least squares
theta = inv(X.T.dot(X)).dot(X.T).dot(y)
print(theta)
```

Running this code performs the calculation and prints the vector $\theta = [\mathbf{1.00233226}]$.

(b) Code to predict and plot:

```
# predict using coefficients
yhat = X.dot(b)
# plot data and predictions
pyplot.scatter(X, y)
pyplot.plot(X, yhat, color='red')
pyplot.show()
```

Plot:



Problem 4 (10 marks)

Pakistan International Airlines has developed 2 different classifiers (A and B) for the prediction whether a flight originating from Lahore will arrive at its final destination on time or not. True or Positive here is ‘On time’ and it refers to the case when the flight is no more than 5 minutes late than the scheduled time. The classifiers were tested on a data-set of 500 flights, and the results are as follows:

	Actual	
	On time	Late
Classifier A, predicted on time	131	155
Classifier A, predicted late	19	195
Classifier B, predicted on time	82	72
Classifier B, predicted late	68	278

(a) [**5 marks**] Which is the preferable classifier in terms of F_1 score?

(b) [**5 marks**] Which is the preferable classifier in terms of accuracy?

Solution:

We first construct confusion matrix for both the classifiers

Classifier A

TP=131	**FP=155**
FN=19	**TN=195**

Classifier B

TP=82	**FP=72**
FN=68	**TN=278**

Noting

$$F_1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2(\text{Precision})(\text{Recall})}{(\text{Precision}) + (\text{Recall})} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}},$$

we obtain $F_1(A) = 0.6009$ and $F_1(B) = 0.5395$ and conclude that the classifier A is better in terms of F_1 score.

Problem 5 (15 marks)

Multivariate Gradient Descent - Gradient descent is an iterative optimization algorithm used to find a local minimum of a function. You are presented with the following feature vector $\boldsymbol{X}$, the parameter vector $\boldsymbol{\theta}$, the bias term $\boldsymbol{b}$ and the gold label $\boldsymbol{y}$.

$$\boldsymbol{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 9 \\ 5 \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} -0.8 \\ 1.0 \\ 0.3 \end{bmatrix}, \quad \boldsymbol{b} = 0.5, \quad \boldsymbol{y} = 2$$

(a) [**13 marks**] Perform 3 iterations of batch gradient descent on the dataset to find updated parameter vectors. Assume learning rate $\alpha = 0.001$ and round your answers to 4 decimal places. Use the least square loss function, and show all your steps. Verify your answer by showing that the loss function is decreasing.

(b) [**2 marks**] Why is the learning rate usually a small value? What is the caveat of a really small learning rate?

Solution:

(a) Incorporate the bias term in the parameter vector as θ_0 and add a corresponding entry x_0 = 1 in the feature vector:

$$\boldsymbol{X} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 9 \\ 5 \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -0.8 \\ 1.0 \\ 0.3 \end{bmatrix}, \quad \boldsymbol{y} = 2$$

We use the following loss function for our calculations:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\theta}^\intercal \boldsymbol{X} - \boldsymbol{y})^2$$

Then we simultaneously update our parameters using the following equation:

$$\theta_i = \theta_i - \alpha \frac{\partial \mathcal{L}}{\partial \theta_i}$$

The parameters vector in the 3 iterations is then as follows:

$$\boldsymbol{\theta_1} = \begin{bmatrix} 0.4942 \\ -0.8232 \\ 0.0522 \\ 0.0290 \end{bmatrix}, \quad \boldsymbol{\theta_2} = \begin{bmatrix} 0.4891 \\ -0.8435 \\ 0.9020 \\ 0.2456 \end{bmatrix}, \quad \boldsymbol{\theta_3} = \begin{bmatrix} 0.4847 \\ -0.8619 \\ 0.8619 \\ 0.2233 \end{bmatrix}$$

and the cost $\mathcal{L}(\boldsymbol{\theta})$ decreases from 16.82 to 12.93 to 9.94.

(b) The learning rate is kept small to avoid overshooting and oscillating around the optimum minimum value. Keeping it very small will increase the number of epochs to reach the optimum solution and significantly increase the convergence time.

Problem 6 (15 marks)

Weighted Ridge Regression - Suppose we have a dataset $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$. Unlike ridge regression where each example is equally important, sometimes certain examples are more important than others and we would want to assign a positive weight ω_i to each training example to indicate the level of importance of each training example. The corresponding loss function is defined as

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{N} \omega_i (\boldsymbol{\beta}^T \boldsymbol{x}_i - y_i)^2$$

(a) **[5 marks]** Suppose $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^T$ and $\boldsymbol{Y} = [y_1, \ldots, y_n]^T$. Find a diagonal matrix $\boldsymbol{W}$ such that we can rewrite the loss function as

$$L(\boldsymbol{\beta}) = (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})^T \boldsymbol{W} (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})$$

(b) **[10 marks]** Using the rewritten loss function, derive a closed form solution for $\boldsymbol{\beta}$ by setting the gradient of the loss function equal to zero.

Solution:

(a) The matrix **W** is such that $diag(\mathbf{W}) = [\omega_1\ \omega_2\ \ldots\ \omega_n]^T$

(b) First simplify $L(\beta)$

$$\begin{aligned}
L(\beta) &= (Y - X\beta)^T W (Y - X\beta) \\
&= (Y^T - \beta^T X^T)(WY - WX\beta) \\
&= Y^T WY - Y^T WX\beta - \beta^T X^T WY + \beta^T X^T WX\beta \\
&= Y^T WY - Y^T WX\beta - (\beta^T X^T WY)^T + \beta^T X^T WX\beta \rightarrow \textcircled{1} \\
&= Y^T WY - Y^T WX\beta - Y^T W^T X\beta + \beta^T X^T WX\beta \\
&= Y^T WY - Y^T WX\beta - Y^T WX\beta + \beta^T X^T WX\beta \rightarrow \textcircled{2} \\
&= Y^T WY - 2Y^T WX\beta + \beta^T X^T WX\beta
\end{aligned}$$

$\textcircled{1} \rightarrow Y^T WX\beta = \beta^T X^T WY$ since their result is a scalar.

$\textcircled{2} \rightarrow W^T = W$ as **W** is a diagonal matrix.

Taking the gradient,

$$\begin{aligned}
\frac{dL(\beta)}{d\beta} &= -2Y^T WX + 2X^T WX\beta \\
&= -2(Y^T WX)^T + 2X^T WX\beta \\
&= -2X^T W^T Y + 2X^T WX\beta \\
&= -2X^T WY + 2X^T WX\beta \\
&= 2X^T W(X\beta - Y)
\end{aligned}$$

Setting $\frac{dL(\beta)}{d\beta} = 0$, we get

$$\begin{aligned}
\frac{dL(\beta)}{d\beta} &= 0 \\
X^T WX\beta &= X^T WY \\
\beta &= (X^T WX)^{-1} X^T WY
\end{aligned}$$

— End of Homework —