LUMS
A Not-for-Profit University

School of Science and Engineering

AI 501 Mathematics for Artificial Intelligence

ASSIGNMENT 3

Due Date: 9:30 am, Saturday, December 1, 2024.
Format: 8 problem, for a total of 100
Instructions:

- You are allowed to collaborate with your peers but copying your colleague's solution is strictly prohibited. This is not a group assignment. Each student must submit his/her own assignment.
- Solve the assignment on blank A4 sheets and staple them before submitting.
- Submit in-class or in the dropbox labeled AI-501 outside the instructor's office.
- **Write your name and roll no. on the first page.**
- Feel free to contact the instructor or the teaching assistants if you have any concerns.

- You represent the most competent individuals in the country, do not let plagiarism come in between your learning. In case any instance of plagiarism is detected, the disciplinary case will be dealt with according to the university's rules and regulations.
- We require you to acknowledge any use or contributions from generative AI tools. Include the following statement to acknowledge the use of AI where applicable.

  I have used [insert Tool Name] to [write, generate, plot or compute; explain specific use of generative AI] [number of times].

Problem 1 (10 marks)

Convex Functions, log-convexity and conjugate functions

Convex functions are one of the most important class of functions, with properties that are of deep importance to Machine Learning. Recall that a function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *convex* if, for all $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$ and for any $\lambda \in [0, 1]$, the following inequality holds:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y})$$

A "shortcut" that one may use for twice-differentiable functions to test for convexity is to differentiate the function twice. Then, given a twice-differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, we say that f is *convex* if $f''(x) \geq 0$ for all $x \in \mathbb{R}^n$.

(a) **[2 marks]** Interpret what the first inequality tells us about Convex functions.

(b) **[5 marks]** For each of the following functions, determine whether they are convex or not. You may use either definition of convexity to prove or disprove this, where applicable. Proper proofs are not required, simple arguments would also suffice.

- $f(x) = ax^2 + bx + c, \forall x \in \mathbb{R}$
- $f(x) = \sin x, \forall x \in \mathbb{R}$
- $f(x) = \cosh x, \forall x \in \mathbb{R}$
- $f(\mathbf{x}) = \min\limits_{i=1,2,\ldots,10} \mathbf{a}_i^T\mathbf{x}, \forall \mathbf{x} \in \mathbb{R}^n$ (Point-wise minimum of 10 linear functions)

(c) **[3 marks]** The conjugate of a function f is defined as:

$$f^*(y) = \sup_x \left(x^T y - f(x)\right)$$

where $f^*(y)$ is the conjugate function, $x \cdot y$ denotes the inner product between x and y, and $\sup_x$ represents the supremum (the least upper-bound) taken over all x.
Find the conjugate function of the functions: [5 marks]

- $f(x) = x^p$ for $x \in \mathbb{R}_{++}$
- $f(x) = \log \sum_{i=1}^n e^{x_i}$, where $x \in \mathbb{R}^n$

Problem 2 (15 marks)

Gradient Descent

Gradient Descent is an iterative optimisation algorithm used to minimize the loss functions in many different machine learning algorithms. This question will explore the difference between Stochastic Gradient Descent (SGD) and Batch Gradient Descent (BGD). We will first start with the definitions of BGD and SGD:

- **Batch Gradient Descent (BGD)**:
  - In Batch Gradient Descent, the entire dataset is used to compute the gradient at each iteration.
  - The update rule for the weights is:

    $$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L(\mathbf{X}, \mathbf{w})$$

    where $L(\mathbf{X}, \mathbf{w})$ is the loss function, η is the learning rate, and $\nabla_{\mathbf{w}} L(\mathbf{X}, \mathbf{w})$ is the gradient of the cost function with respect to the parameters $\mathbf{w}$.
- **Stochastic Gradient Descent (SGD)**:
  - In Stochastic Gradient Descent, only a single training example (or a few in mini-batch) is used to compute the gradient at each iteration.
  - The update rule is:

    $$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L_i(\mathbf{w})$$

    where $L_i(\mathbf{w})$ is the cost function for the i-th training example.

(a) **[5 marks]** Given the following loss function, find an analytical expression for the update rules for both stochastic and batch gradient descent.

$$L(\mathbf{w}, \mathbf{x}) = \frac{1}{N} \sum_{i=0}^{N} \alpha_i (y_i - \mathbf{x}_i^T \mathbf{w})^2 + \lambda \|\mathbf{w}\|_2^2$$

(b) **[6 marks]** In this part, we consider the least square loss function given by:

$$L = \|y - X^T \mathbf{w}\|_2^2$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 9 \\ 5 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} -0.8 \\ 1.0 \\ 0.3 \end{bmatrix}, \quad b = 0.5, \quad y = 2$$

Perform 3 iterations of batch gradient descent on the dataset to find the updated parameter vectors. Assume learning rate $\eta = 0.001$ 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.

(c) **[4 marks]** Copy the code below to a google colab jupyter notebook. Run the code as is without making any changes, except η, which is the learning rate. Choose the following values of η, and explain what you observe about the BGD algorithm in terms of its convergence and the iterations it took for convergence:

- $\eta = 0.07$
- $\eta = 0.2$
- $\eta = 0.5$
- $\eta = 0.6$

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

f = lambda w: 2 * w**2 - w + 1
df = lambda w: 4 * w - 1

wspace = np.linspace(-1.5, 2, 1000)
Objfn = f(wspace)

plt.plot(wspace, Objfn, linewidth=2)
plt.xlabel('w')
plt.ylabel('Loss')
plt.title('Gradient Descent')

w = np.random.choice(wspace)
w = 1.4
maxiter = 100
eta = 0.6
eps = 1e-2
iterno = 1

plt.plot(w, f(w), 'or', markersize=6, markerfacecolor='r')
plt.text(w, f(w) - 0.4, str(iterno), fontsize=10, fontweight='bold')

while (iterno ¡ maxiter) and (abs(df(w)) ¿ eps):
w = w - eta * df(w)
iterno += 1

plt.plot(w, f(w), 'or', markersize=6, markerfacecolor='r')
if iterno ¡ 5:
plt.text(w, f(w) - 0.4, str(iterno), fontsize=10, fontweight='bold')

plt.show()

print(f'Final value of w after iterno iterations: w")
```

Problem 3 (10 marks)

Convex Sets

A set C is called convex if for any two points $x_1, x_2 \in C$, the line segment joining x_1 and x_2 is entirely contained within C. Mathematically, this means that for any$\theta \in [0, 1]$, the point

$$\theta x_1 + (1 - \theta)x_2 \in C$$

In other words, a set is convex if, for any two points in the set, every point on the straight line between them also lies within the set.

Using this definition, show that the following sets are convex:

- $B = \{x \in \mathbb{R}^n | \|x - x_0\| \leq r\}$
- $C = \{x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n | x_1 \leq p_1, x_2 \leq p_2, .., x_n \leq p_n\}$
- $S = \left\{ x \in \mathbb{R}^n \middle| -2 \leq \left(\sum_{k=1}^{n} x_k \cos kt \right) \leq 4 \text{ for } |t| \leq 3 \right\}$

Problem 4 (15 marks)

Linear Programming

Linear Programming deals with the problem of optimizing a linear objective function subject to linear equality and inequality constraints on the decision variables. In matrix form, the standard LP formulation is:

$$\min c^T x + d$$

$$\text{subject to } Gx \preceq h$$

$$Ax = b$$

where $G \in \mathbb{R}^{m \times n}$ and $A \in \mathbb{R}^{p \times n}$.

(a) **[6 marks]** Formulate the following optimization problems as linear programs. Here $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$.

- $\min \|Ax - b\|_\infty$
- $\min \|Ax - b\|_1$ subject to $\|x\|_\infty \leq 1$
- $\min \|Ax - b\|_1 + \|x\|_\infty$

(b) **[9 marks]** We consider a data center with four servers, each handling requests at a certain processing rate. The Quality-of-Service (QoS) for the k-th server is measured by its response rate, given by

$$\text{QoS}_k = \frac{R_k}{C + \sum\limits_{i=1, i \neq k} R_i},$$

where R_i (for $i = 1, 2, 3, 4$) is the rate of requests directed to the i-th server, and C represents a constant overhead in the system.

We aim to allocate request rates to each server to minimize the total rate allocation while meeting the QoS requirements, defined by $\text{QoS}_k \geq \alpha_k$ for $k = 1, 2, 3, 4$. Argue that the problem is convex and formulate the problem as a linear program (LP).

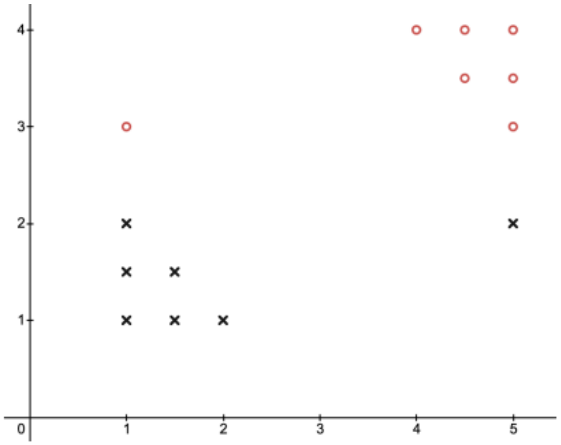

Figure 1

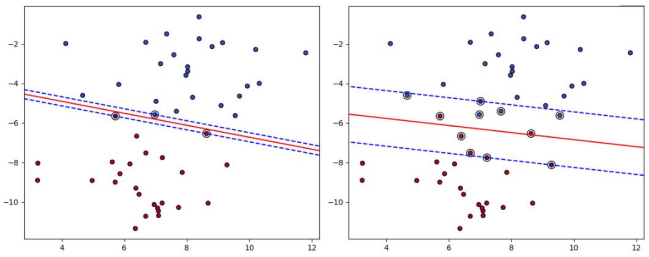

Figure 2

Problem 5 (15 marks)

Support Vector Machine

(a) **[5 marks]** Given the sample points on figure. 1, draw and label two lines: the decision boundary learned by a hard-margin SVM and the decision boundary learned by a soft-margin SVM. We are not specifying the hyperparameter C, but don't make C too extreme. (We are looking for a qualitative difference between hard- and soft-margin SVMs.) Label the two lines clearly. Also draw and label four dashed lines to show the margins of both SVMS.

(b) **[5 marks]** Show that the following functions are valid kernels for SVM:

- $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$, where d is the degree of the polynomial.
- $K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right)$, where σ is a real scalar called standard deviation.
- $K(\mathbf{x}, \mathbf{y}) = \tanh(\alpha\mathbf{x}^T\mathbf{y} + c)$, where α is a real scalar and c is a real constant.

(c) **[5 marks]** Given the SVMs on figure 2, answer the following questions:

- Which one of the two is the Hard SVM and which one the Soft SVM? Explain.
- What are highlighted points in each plot, and what is the role they play?
- Is the data-linearly separable? Which SVM would you prefer to use in a scenario where the data is not linearly separable and why?
- Which of the two SVMs is more sensitive to outliers? Explain your answer.

Problem 6 (10 marks)

SVM as a Quadratic Program The standard Quadratic Program formulation is given as:

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}$$

subject to:

$$G\mathbf{x} \leq \mathbf{h}$$

and, optionally, equality constraints:

$$A\mathbf{x} = \mathbf{b}.$$

Given the hard-SVM formulation, reformulate this as a QP.

We want to minimize $\frac{1}{2}\|\mathbf{w}\|^2$ subject to $y_i(\mathbf{w}^T x_i + b) \geq 1$.

Principal Component Analysis Overview

PCA is a powerful technique used for dimensionality reduction, enabling the projection of high-dimensional data onto a lower-dimensional subspace while preserving as much variance as possible.

- The process begins with mean subtraction, where we compute the mean of the dataset and subtract it from each data point, resulting in a dataset centered around zero.
- The next step is standardization, where each data point is divided by the standard deviation of the entire dataset for that dimension, transforming the data into a unit-free format with a variance of 1 along each axis. This ensures that subsequent analysis is not skewed by differences in scale among the variables.
- This is followed by the construction of the covariance matrix of the standardized data. We find its eigenvalues and corresponding eigenvectors. The eigenvalues indicate the amount of variance captured by their corresponding eigenvectors, and the eigenvectors are scaled according to the magnitude of their eigenvalues, creating a set of orthogonal basis vectors that represent the principal components. The principal subspace corresponding to the largest eigenvector captures the most variance in the data.
- We can project any new data point onto the principal subspace by first standardizing it using the mean and standard deviation of the training data. The projection yields the coordinates in the context of the standardized dataset.
- Finally, to transform these projections back in the original data space, we must undo the standardization by multiplying the standardized projections by the standard deviation and then adding the mean back. This allows us to visualize and interpret the projected data points in relation to the original dataset.

These steps are also illustrated in the figure below:

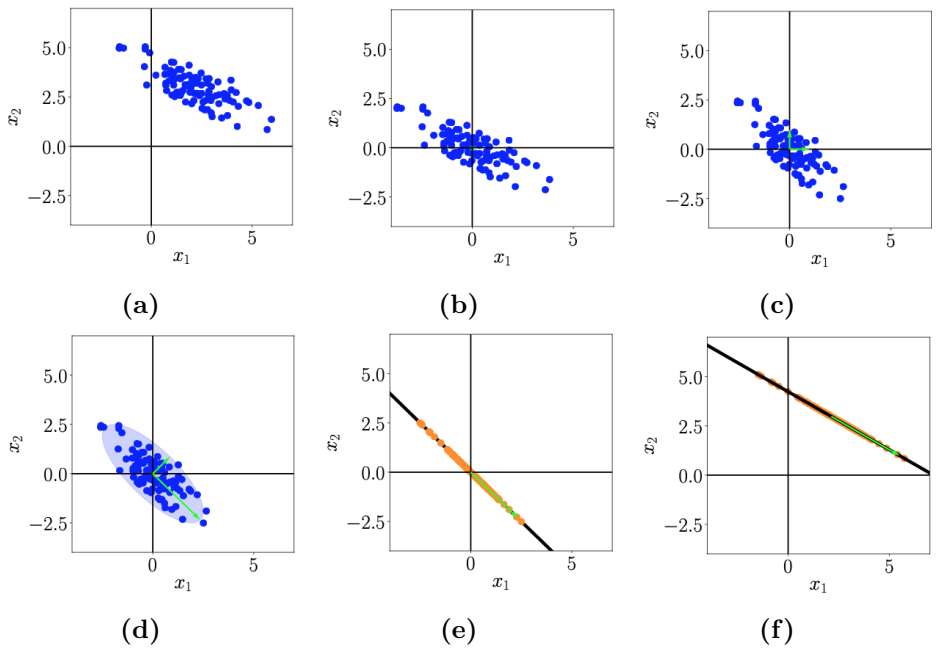

Figure 3: (a) Original dataset. (b) Centering by subtracting mean. (c) Dividing by standard deviation. (d) Compute eigenvalues and eigenvectors. (e) Project data onto principal subspace. (f) Undo standardization, move projected data in original space.
Figure source: Mathematics for Machine Learning by Deisenroth, pg. 337.

Problem 7 (10 marks)

Given the following matrix $\mathbf{X}$,

$$\mathbf{X} = \begin{bmatrix} 2 & 3 \\ 5 & 6 \\ 7 & 8 \\ 6 & 5 \\ 9 & 3 \\ 11 & 10 \\ 12 & 9 \end{bmatrix}$$

for $n = 7$ and $d = 2$.

We will make use of Principal Component Analysis (PCA) to reduce the dimensions of the matrix $\mathbf{X}$ from $d = 2$ to $d = 1$ by carrying out the following steps:

(a) [**2 marks**] Plot the data points on a 2-dimensional plane.

(b) [**4 marks**] Compute the principal components using the procedure taught in class (refer to the slides) and plot them as well.

(c) [**4 marks**] Now, project the original data matrix $\mathbf{X}$ onto its first principal component and plot on a 1-dimensional number line.

Problem 8 (15 marks)

Linear Discriminant Analysis

LDA tries to find a linear combination of features that achieves maximum separation for samples between classes and minimum separation of samples within each class. Here we will assume only two classes, but this can easily be generalized to more classes. We will use LDA to project our data onto a line.

LDA achieves this by:

1. Maximizing the distance between the mean of the two classes.
2. Minimizing the scatter (variation) within each class.

Mathematically, we want to find a projection vector $\mathbf{w}$ which we can use to obtain the one-dimensional approximation (projection) of each data-point $\mathbf{x}_i$ as $z_i = \mathbf{w}^T\mathbf{x}_i$, such that the following objective function is maximized:

$$J(\mathbf{w}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

where the numerator is the difference between the projected class means, and the denominator is the within-class scatter of the projected samples defined as:

$$\tilde{s}_i^2 = \sum_{z \in \text{Class}_i} (z - \tilde{\mu}_i)^2$$

Here $z = \mathbf{w}^T\mathbf{x}$ is the projected sample, and $\tilde{\mu}_i$ is the projected class mean for the i-th class. In simple words, we want a projection such that samples of the same class are projected close to each other and the class means of the projected samples are far from each other.

(a) **[3 marks]** First, we will prove that the objective function formulated above can be expressed in terms of projection vector $\mathbf{w} \in \mathbb{R}^d$ as:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}$$

where

- $\mathbf{S}_B$ is the between-class scatter matrix of the samples in the original space:
$$\mathbf{S}_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$
- $\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$ is the within-class scatter matrix, where $\mathbf{S}_i$ is the covariance matrix of class i, given by:
$$\mathbf{S}_i = \sum_{\mathbf{x} \in \text{Class}_i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T$$
- μ_i denotes the mean of samples for the i-th class.

(b) **[3 marks]** Show that $\mathbf{S}_W$ and $\mathbf{S}_B$ are symmetric and positive semi-definite.

(c) **[7 marks]** In part (a), we have the formulation of the objective function in terms of the projection vector $\mathbf{w}$. We want to determine $\mathbf{w}$ as a solution to the following optimization problem:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} J(\mathbf{w})$$

Assuming that $\mathbf{S}_W$ is non-singular, show that the solution is the eigenvector of $\mathbf{S}_W^{-1}\mathbf{S}_B$ corresponding to the largest eigenvalue.

Now we have a closed-form solution of LDA, we will implement it on a simple dataset for a binary classification problem.

The data set is as follows:

X_1	X_2	Label
1	1	0
2	2	0
3	4	0
8	8	1
7	10	1
8	7	1

1. Visualize the data.
2. Project the data onto a line using LDA and visualize it again.

You may use the following python code for visualization:

https://www.kaggle.com/code/ooyun619/visualization

However, the LDA must be performed by you yourself. No solution to this part will be accepted without handwritten (or Latex) solutions for the LDA. Extensive calculations may be omitted, if the answers to those calculations are reached at correctly.

(d) [**2 marks**] What do you observe about the above visualizations?

— End of Assignment —