LUMS
A Not-for-Profit University

School of Science and Engineering

AI 501 Mathematics for Artificial Intelligence

ASSIGNMENT 3 – SOLUTIONS

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 $x \in \mathbb{R}^n$

Solution:

(a) The inequality

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

tells us that for any two points $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$ and any $\lambda \in [0, 1]$, the function value at a convex combination of $\mathbf{x}$ and $\mathbf{y}$ is less than or equal to the convex combination of the function values at $\mathbf{x}$ and $\mathbf{y}$. This implies that the graph of a convex function lies below or on the line segment connecting any two points on the graph. This is the geometric interpretation of convexity.

(b)

- $f(x) = ax^2 + bx + c, \forall x \in \mathbb{R}$:
  The second derivative is $f''(x) = 2a$. If $a \geq 0$, then $f(x)$ is convex, as $f''(x) \geq 0$. Otherwise, it is not convex.
- $f(x) = \sin x, \forall x \in \mathbb{R}$:
  The second derivative is $f''(x) = -\sin x$, which oscillates between -1 and 1. Since $f''(x)$ is not non-negative everywhere, $f(x)$ is not convex.
- $f(x) = \cosh x, \forall x \in \mathbb{R}$:
  The second derivative is $f''(x) = \cosh x$, which is always non-negative. Hence, $f(x)$ is convex.
- $f(\mathbf{x}) = \min\limits_{i=1,2,\ldots,10} \mathbf{a}_i^T \mathbf{x}, \forall \mathbf{x} \in \mathbb{R}^n$:
  The point-wise minimum is not in general a convex function. For affine functions, this is in fact a concave function. To see this, refer to 1.
  We can generalize this to more than four functions, but the principle is the same. Hence, $f(x)$ is not convex.

(c)

- $f(x) = x^p$ for $x \in \mathbb{R}_{++}$:
  The conjugate function is defined as:

$$f^*(y) = \sup_{x>0} \left(xy - x^p\right).$$

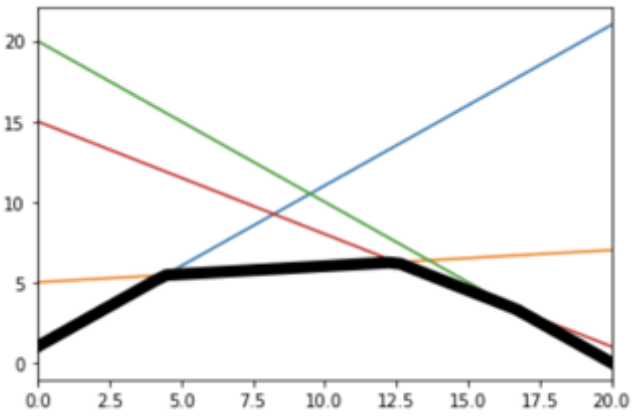

Figure 1: The point-wise minimum of 4 affine functions

Since we need to find the value of x for which the function in the brackets is maximum, we take the derivative of the function, find the value of x for which it is maximum in terms of y, and then substitute back.

Taking the derivative with respect to x and setting it to zero gives:

$$y - px^{p-1} = 0 \implies x = \left(\frac{y}{p}\right)^{\frac{1}{p-1}}.$$

Substituting back:

$$f^*(y) = \left(\frac{y}{p}\right)^{\frac{1}{p-1}} y - \left(\frac{y}{p}\right)^{\frac{p}{p-1}} = \frac{p-1}{p}\left(\frac{y}{p}\right)^{\frac{p}{p-1}},$$

provided $y > 0$.

- $f(x) = \log \sum_{i=1}^n e^{x_i}$, where $x \in \mathbb{R}^n$:
  The conjugate function is:

$$f^*(y) = \sup_x \left(x^T y - \log \sum_{i=1}^n e^{x_i}\right).$$

Let $g(x) = x^T y - \log \sum_{i=1}^n e^{x_i}$. Taking the gradient with respect to x and setting it to zero:

$$y - \frac{e^x}{\sum_{i=1}^n e^{x_i}} = 0.$$

This implies $y_i = \frac{e^{x_i}}{\sum_{i=1}^n e^{x_i}}$, and thus y is a probability vector ($\sum_{i=1}^n y_i = 1$). Taking the natural log on both sides leads us to:

$$\log y = x - \log \sum_{i=1}^n \exp^{x_i}$$

$$x = \log y + \log \sum_{i=1}^n \exp^{x_i}$$

Which implies

$$x_i = \log y_i + \log \sum_{i=1}^n \exp^{x_i}$$

Substituting back, we find:

$$f^*(y) = \sum_{i=1}^n x_i y_i - \log \sum_{i=1}^n \exp^{x_i} = \sum_{i=1}^n (\log y_i + \log \sum_{i=1}^n \exp^{x_i}) y_i - \log \sum_{i=1}^n \exp^{x_i}$$

which simplifies to

$$f^*(y) = \sum_{i=1}^n y_i \log y_i$$

which is the negative entropy function when $y \in \Delta^n$ (the probability simplex).

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")
```

Solution:

(a) [**5 marks**] Analytical expression for the update rules for both Stochastic and Batch Gradient Descent:
Given the loss function:

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

where α_i is a constant, we first recast it in matrix notation:

$$L(\mathbf{w}, \mathbf{x}) = \frac{1}{N} (y - X^T \mathbf{w})^T \text{diag}(\alpha_i)(y - X^T \mathbf{w}) + \lambda \mathbf{w}^\mathbf{T} \mathbf{w}$$

- Batch Gradient Descent (BGD): The gradient of the loss function L with respect to $\mathbf{w}$ is:

$$\nabla_\mathbf{w} L(\mathbf{w}, \mathbf{x}) = -\frac{2}{N} X \text{diag}(\alpha_i)(y - X^T \mathbf{w}) + 2\lambda \mathbf{w}.$$

The update rule for Batch Gradient Descent is:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \nabla_{\mathbf{w}} L(\mathbf{w}_k, \mathbf{x}),$$

where η is the learning rate.

Substituting the gradient:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \frac{2\eta}{N} X \text{diag}(\alpha_i)(y - X^T \mathbf{w}_k) - 2\eta\lambda \mathbf{w}_k.$$

Stochastic Gradient Descent (SGD): For Stochastic Gradient Descent, the loss function is computed for a single data point $(\mathbf{x}_i, y_i)$. The gradient of the loss with respect to $\mathbf{w}$ for this single data point is:

$$\nabla_{\mathbf{w}} L_i(\mathbf{w}, \mathbf{x}_i) = -2\alpha_i \mathbf{x}_i (y_i - \mathbf{x}_i^T \mathbf{w}) + 2\lambda \mathbf{w}.$$

The update rule for Stochastic Gradient Descent is:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \nabla_{\mathbf{w}} L_i(\mathbf{w}_k, \mathbf{x}_i).$$

Substituting the gradient:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + 2\eta\alpha_i \mathbf{x}_i (y_i - \mathbf{x}_i^T \mathbf{w}_k) - 2\eta\lambda \mathbf{w}_k.$$

(b) **[6 marks]** Performing 3 iterations of batch gradient descent:

Least Squares Loss Function:

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

Given:

$$X = \begin{bmatrix} 4 \\ 9 \\ 5 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} -0.8 \\ 1.0 \\ 0.3 \end{bmatrix}, \quad b = 0.5, \quad y = 2.$$

Gradient of L with respect to $\mathbf{w}$:

$$\nabla_{\mathbf{w}} L = -2X(y - X^T \mathbf{w} - b).$$

Gradient of L with respect to b:

$$\nabla_b L = -2(y - X^T \mathbf{w} - b).$$

Iteration 1:

- Compute predictions: $y_{\text{pred}} = X^T \mathbf{w} + b$

$$y_{\text{pred}} = 4(-0.8) + 9(1.0) + 5(0.3) + 0.5 = -3.2 + 9 + 1.5 + 0.5 = 7.8.$$

- Compute gradients:

$$\nabla_{\mathbf{w}} L = -2X(y - y_{\text{pred}}) = -2 \begin{bmatrix} 4 \\ 9 \\ 5 \end{bmatrix} (2 - 7.8) = -2 \begin{bmatrix} 4 \\ 9 \\ 5 \end{bmatrix} (-5.8) = \begin{bmatrix} -46.4 \\ -104.4 \\ -58.0 \end{bmatrix}.$$

$$\nabla_b L = -2(y - y_{\text{pred}}) = -2(2 - 7.8) = 11.6.$$

- Update parameters:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L, \quad b \leftarrow b - \eta \nabla_b L.$$

$$\mathbf{w} = \begin{bmatrix} -0.8 \\ 1.0 \\ 0.3 \end{bmatrix} - 0.001 \begin{bmatrix} -46.4 \\ -104.4 \\ -58.0 \end{bmatrix} = \begin{bmatrix} -0.7536 \\ 1.1044 \\ 0.3580 \end{bmatrix}.$$

$$b = 0.5 - 0.001(11.6) = 0.4884.$$

Iteration 2:

- Compute new prediction and gradients:

$$y_{\text{pred}} = 4(-0.7536) + 9(1.1044) + 5(0.3580) + 0.4884 = 7.3528.$$

$$\nabla_{\mathbf{w}} L = -2X(2 - 7.3528) = \begin{bmatrix} -42.8224 \\ -96.8508 \\ -53.1804 \end{bmatrix}, \quad \nabla_b L = 10.7056.$$

$$\mathbf{w} = \begin{bmatrix} -0.7536 \\ 1.1044 \\ 0.3580 \end{bmatrix} - 0.001 \begin{bmatrix} -42.8224 \\ -96.8508 \\ -53.1804 \end{bmatrix} = \begin{bmatrix} -0.7108 \\ 1.2012 \\ 0.4112 \end{bmatrix}.$$

$$b = 0.4884 - 0.001(10.7056) = 0.4777.$$

Iteration 3:

- Compute new prediction and gradients:

$$y_{\text{pred}} = 4(-0.7108) + 9(1.2012) + 5(0.4112) + 0.4777 = 6.9201.$$

$$\nabla_{\mathbf{w}} L = -2X(2 - 6.9201) = \begin{bmatrix} -39.3608 \\ -89.6118 \\ -49.7844 \end{bmatrix}, \quad \nabla_b L = 9.8402.$$

$$\mathbf{w} = \begin{bmatrix} -0.7108 \\ 1.2012 \\ 0.4112 \end{bmatrix} - 0.001 \begin{bmatrix} -39.3608 \\ -89.6118 \\ -49.7844 \end{bmatrix} = \begin{bmatrix} -0.6714 \\ 1.2908 \\ 0.4610 \end{bmatrix}.$$

$$b = 0.4777 - 0.001(9.8402) = 0.4679.$$

Verification: Compute the loss values at each iteration:

- Initial: $L = (2 - 7.8)^2 = 33.64$
- Iteration 1: $L = (2 - 7.3528)^2 = 28.03$
- Iteration 2: $L = (2 - 6.9201)^2 = 24.21$

The loss decreases, verifying correctness.

(c) There is a trade-off between the learning rate η and the number of iterations it takes for the algorithm to converge. As we decrease the learning rate, the number of iterations it takes to converge increases. However, if the learning rate is increased too much, it results in the algorithm not converging. BGD just oscillates between two points and cannot find the global minima. This is because the learning rate is a measure of the step-size, and increasing the step-size too much leads to the algorithm missing the optimal point.

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:

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

Solution: 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

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

Using this definition, we will verify the convexity of the given sets.

1. **Set B:** $B = \{x \in \mathbb{R}^n | \|x - x_0\| \leq r\}$
   Let $x_1, x_2 \in B$. This implies:
   $$\|x_1 - x_0\| \leq r \quad \text{and} \quad \|x_2 - x_0\| \leq r.$$
   Consider any $\theta \in [0,1]$ and the convex combination $x = \theta x_1 + (1-\theta)x_2$. Using the triangle inequality:
   $$\|x - x_0\| = \|\theta(x_1 - x_0) + (1-\theta)(x_2 - x_0)\| \leq \theta\|x_1 - x_0\| + (1-\theta)\|x_2 - x_0\|.$$
   Since $\|x_1 - x_0\| \leq r$ and $\|x_2 - x_0\| \leq r$, we have:
   $$\|x - x_0\| \leq \theta r + (1-\theta)r = r.$$
   Thus, $x \in B$, proving B is convex.

2. **Set C:** $C = \{x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n | x_i \leq p_i \text{ for all } i = 1, 2, \ldots, n\}$
   Let $x_1, x_2 \in C$. This implies $x_{1i} \leq p_i$ and $x_{2i} \leq p_i$ for all i. Consider any $\theta \in [0,1]$ and the convex combination $x = \theta x_1 + (1-\theta)x_2$. For each coordinate i:
   $$x_i = \theta x_{1i} + (1-\theta)x_{2i} \leq \theta p_i + (1-\theta)p_i = p_i.$$
   Thus, $x \in C$, proving C is convex.

3. **Set S:** $S = \left\{ x \in \mathbb{R}^n \middle| -2 \leq \sum_{k=1}^{n} x_k \cos(kt) \leq 4 \text{ for } |t| \leq 3 \right\}$
   Let $x_1, x_2 \in S$. This implies:
   $$-2 \leq \sum_{k=1}^{n} x_{1k} \cos(kt) \leq 4 \quad \text{and} \quad -2 \leq \sum_{k=1}^{n} x_{2k} \cos(kt) \leq 4 \quad \text{for all } |t| \leq 3.$$
   Consider any $\theta \in [0,1]$ and the convex combination $x = \theta x_1 + (1-\theta)x_2$. Then:
   $$\sum_{k=1}^{n} x_k \cos(kt) = \theta \sum_{k=1}^{n} x_{1k} \cos(kt) + (1-\theta) \sum_{k=1}^{n} x_{2k} \cos(kt).$$
   Using the bounds for x_1 and x_2:
   $$-2\theta \leq \theta \sum_{k=1}^{n} x_{1k} \cos(kt) \leq 4\theta \quad \text{and} \quad -2(1-\theta) \leq (1-\theta) \sum_{k=1}^{n} x_{2k} \cos(kt) \leq 4(1-\theta)$$
   Adding these two inequalities up we get
   $$-2 \leq \theta \sum_{k=1}^{n} x_{1k} \cos(kt) + (1-\theta) \sum_{k=1}^{n} x_{2k} \cos(kt) \leq 4.$$
   Hence, $x \in S$, proving S is convex.

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

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

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).

Solution:

(a)

- Equivalent to the LP

$$\text{minimize } t \quad \text{subject to } \begin{cases} Ax - b \leq t\mathbf{1} \\ Ax - b \geq -t\mathbf{1} \end{cases}$$

in the variables x, t. To see the equivalence, assume x is fixed in this problem, and we optimize only over t. The constraints say that

$$-t \leq a_k^T x - b_k \leq t$$

for each k, i.e., $t \geq |a_k^T x - b_k|$, i.e.,

$$t \geq \max_k |a_k^T x - b_k| = \|Ax - b\|_\infty.$$

Clearly, if x is fixed, the optimal value of the LP is $p^*(x) = \|Ax - b\|_\infty$. Therefore optimizing over t and x simultaneously is equivalent to the original problem.

- Equivalent to the LP

$$\text{minimize } \mathbf{1}^T y \quad \text{subject to } \begin{cases} -y \preceq Ax - b \preceq y \\ -1 \leq x \leq 1 \end{cases}$$

with variables $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$.

- Equivalent to

$$\text{minimize } \mathbf{1}^T y + t \quad \text{subject to } \begin{cases} -y \preceq Ax - b \preceq y \\ -t\mathbf{1} \preceq x \preceq t\mathbf{1} \end{cases}$$

with variables x, y, and t.

QoS for the k-th server is given by:

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

and the requirement is $\text{QoS}_k \geq \alpha_k$, where $\alpha_k > 0$ for $k = 1, 2, 3, 4$.

Rewriting the inequality:

$$\frac{R_k}{C + \sum\limits_{i=1, i\neq k} R_i} \geq \alpha_k.$$

Multiplying through by the denominator (which is positive for all valid $R_i \geq 0$):

$$R_k \geq \alpha_k \left(C + \sum_{i=1, i\neq k} R_i \right).$$

Expanding the summation:

$$R_k \geq \alpha_k C + \alpha_k \sum_{i=1, i\neq k} R_i.$$

Rearranging terms to isolate the variables R_i on one side:

$$R_k - \alpha_k \sum_{i=1, i\neq k} R_i \geq \alpha_k C.$$

Notice that this is a **linear inequality** in R_k and R_i. The objective, as we will show below, is also linear. Since both the constraints and the objective function are linear, the problem can be formulated as a linear program (LP), which is a convex optimization problem.

We aim to **minimize the total rate allocation**:

$$\text{minimize} \sum_{k=1}^{4} R_k.$$

Subject to the QoS constraints for each server $k = 1, 2, 3, 4$:

$$R_k - \alpha_k \sum_{i=1, i\neq k} R_i \geq \alpha_k C.$$

Expanding the summation for each constraint explicitly, the constraints become:

$$R_k - \alpha_k (R_1 + R_2 + R_3 + R_4 - R_k) \geq \alpha_k C, \quad \text{for } k = 1, 2, 3, 4.$$

Simplify each constraint:

$$R_k(1 + \alpha_k) - \alpha_k \sum_{i=1}^{4} R_i \geq \alpha_k C, \quad \text{for } k = 1, 2, 3, 4.$$

Finally, the constraints and the objective can be summarized as:

Objective:

$$\text{minimize} \sum_{k=1}^{4} R_k$$

Subject to (for each k):

$$(1 + \alpha_k) R_k - \alpha_k \sum_{i=1}^{4} R_i \geq \alpha_k C,$$
$$R_k \geq 0, \quad \text{for all } k.$$

This is a **convex optimization problem** since the objective and constraints are linear, and LPs are a subset of convex problems.

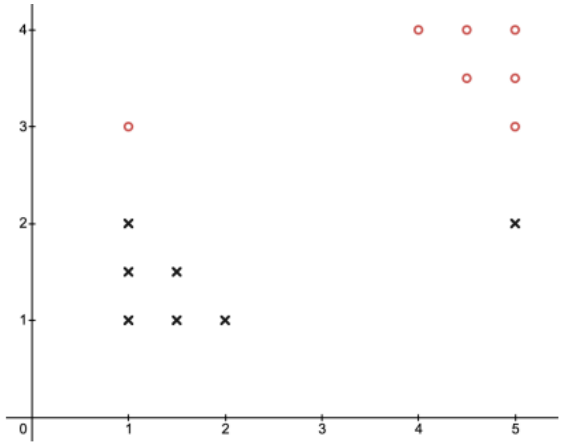

Figure 2

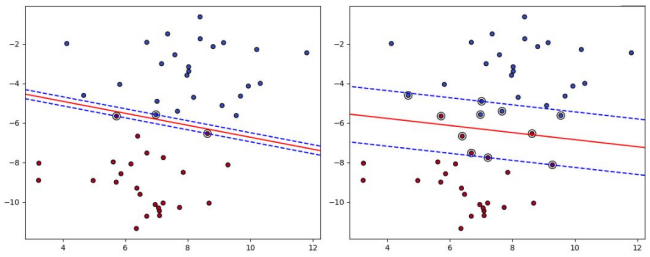

Figure 3

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.

Solution:

(a) The figure 4 shows one of the possible decision boundaries

(b) We are given three candidate functions, and we need to show that each of them is a valid kernel for Support Vector Machines (SVM).

To show that the following kernel functions are valid, we use the fact that a function $K(\mathbf{x}, \mathbf{y})$ is a valid kernel if it can be written as an inner product in some feature space. We demonstrate this for each kernel by showing how each function can be related to an inner product.

- **Polynomial Kernel:**
$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$
The polynomial kernel can be interpreted as an inner product in a feature space constructed by mapping the input vectors $\mathbf{x}$ and $\mathbf{y}$ to a higher-dimensional space. We can expand the expression $(\mathbf{x}^T\mathbf{y} + 1)^d$ using the binomial expansion. This expansion corresponds to the inner product of the feature mappings of $\mathbf{x}$ and $\mathbf{y}$ in the higher-dimensional space, as shown below:
$$(\mathbf{x}^T\mathbf{y} + 1)^d = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$
where $\phi(\mathbf{x})$ is a feature map that involves terms of the form $x_1^{k_1} x_2^{k_2} \dots x_n^{k_n}$ (where the exponents sum to d), which corresponds to the monomials in the expansion. Since this function can be written as an inner product, it is a valid kernel.

- **Gaussian (RBF) Kernel:**
$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right)$$
The Gaussian kernel is known to be a valid kernel, and it can be interpreted as an inner product in a high-dimensional feature space. The function $\|\mathbf{x} - \mathbf{y}\|^2$ can be expanded as:
$$\|\mathbf{x} - \mathbf{y}\|^2 = \mathbf{x}^T\mathbf{x} + \mathbf{y}^T\mathbf{y} - 2\mathbf{x}^T\mathbf{y}$$
The exponential function of the squared distance between $\mathbf{x}$ and $\mathbf{y}$ corresponds to an inner product in an infinite-dimensional feature space, where each point $\mathbf{x}$ is mapped to a set of basis functions that depend on the distance between the points. This function satisfies the conditions of Mercer's theorem, making it a valid kernel.

- **Hyperbolic Tangent Kernel:**
$$K(\mathbf{x}, \mathbf{y}) = \tanh(\alpha\mathbf{x}^T\mathbf{y} + c)$$
The hyperbolic tangent kernel can be interpreted as the inner product of feature mappings derived from a single-layer neural network with a sigmoid activation function. Specifically, this kernel corresponds to the inner product in a feature space where the features are derived from the weights and bias of the neural network. Since the function tanh is continuous and the corresponding feature map can be constructed from the neural network, this kernel is valid as well.
$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$
where $\phi(\mathbf{x})$ represents the features derived from the neural network. Therefore, the hyperbolic tangent kernel is a valid kernel.

(c)
- **Hard vs Soft SVM:** The Hard SVM corresponds to the plot where there is no margin violation; all data points lie on the correct side of the margin. This corresponds to the plot on the left. In contrast, the Soft SVM allows some data points to be on the wrong side of the margin, introducing slack variables to handle noise or non-linearly separable data. Therefore, the plot on the right is the soft SVM.
- **Highlighted Points:** The highlighted points in the plots are the support vectors. These are the critical data points that lie on the margin or violate it. The support vectors are the data points closest to the decision boundary and play a crucial role in defining the optimal hyperplane. They are used to calculate the optimal margin in SVM.
- **Linear Separability:** Yes the data is linearly separable. In most cases, a hard SVM formulation would be best for Linearly separable data. However, in this case, the separability of data isn't very clearcut. A soft SVM formulation may therefore be more useful as the datapoints most near the margin may be outliers.
- **Sensitivity to Outliers:** Hard SVM is more sensitive to outliers compared to soft SVM. we can see this because the margins of the hard SVM are really tight, aiming for perfect separation between the classes. As such, it overfits and is more sensitive to noise. Soft SVM on the otherhand uses slack variables to ensure that it does not let outliers influence it too much.

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 $y_i(\mathbf{w}^T x_i + b) \geq 1$.

Solution:

The standard formulation of a Quadratic Program (QP) is:

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

subject to:

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

and optionally, equality constraints:

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

In the case of the hard-SVM, we want to minimize the following objective:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2,$$

subject to the constraints:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \text{for all } i.$$

Reformulating as a Quadratic Program: 1. **Objective Function**: The objective function $\frac{1}{2}\|\mathbf{w}\|^2$ can be rewritten as:

$$\frac{1}{2}\mathbf{w}^T\mathbf{w}.$$

This is a quadratic function of the weight vector $\mathbf{w}$.

2. **Constraints**: The constraints $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$ for each training sample i can be written as:

$$-y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq -1.$$

In matrix form, these constraints can be written as:

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

where G is a matrix of the data points and labels, and $\mathbf{h}$ is a vector containing the value -1 for each constraint.

Thus, the hard-SVM problem can be reformulated as the following Quadratic Program:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i.$$

This is a standard Quadratic Program where we aim to minimize the quadratic objective subject to linear constraints.

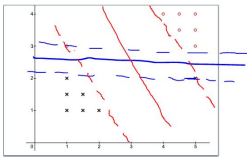

Figure 4

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:

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?

Solution:

(a)

$$S_i = \sum_{\mathbf{x} \in Class_i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T$$

S_i is the covariance matrix of the original data.

$$\tilde{S}_i = \sum_{\mathbf{x} \in Class_i} (\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mu_i)(\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mu_i)^T$$

Each bracket in the above term is a scalar, so multiplying by its transpose is the same as taking its square, which makes it equal to the denominator in the original loss function.

$$\tilde{S}_i = \sum_{\mathbf{x} \in Class_i} \mathbf{w}^T(\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T\mathbf{w}$$

We can take W out of summation because summation is over x, what is left inside summation is S_i.

$$\tilde{S}_i = \mathbf{w}^T S_i \mathbf{w}$$

$$\tilde{S}_W = \tilde{S}_1 + \tilde{S}_2 = \mathbf{w}^T S_1 \mathbf{w} + \mathbf{w}^T S_2 \mathbf{w} = \mathbf{w}^T(S_1 + S_2)\mathbf{w} = \mathbf{w}^T S_W \mathbf{w}$$

Now for the numerator of the original objective function.

$$\tilde{\mu} = \mathbf{w}^T\mu$$

$$||\mathbf{w}^T\mu_\mathbf{1} - \mathbf{w}^T\mu_\mathbf{2}||_2^2 = (\mathbf{w}^T\mu_\mathbf{1} - \mathbf{w}^T\mu_\mathbf{2})(\mathbf{w}^T\mu_\mathbf{1} - \mathbf{w}^T\mu_\mathbf{2})^T$$

We can write them like this because these are scalars.

$$\mathbf{w}^T(\mu_\mathbf{1} - \mu_\mathbf{2})(\mu_\mathbf{1} - \mu_\mathbf{2})^T\mathbf{w}$$

Distributive property.

$$\mathbf{w}^T S_B \mathbf{w}$$

This gives us the required representation.

(b) S_B is symmetric.

$$\mathbf{S_B} = (\mu_\mathbf{1} - \mu_\mathbf{2})(\mu_\mathbf{1} - \mu_\mathbf{2})^T$$

$$\mathbf{S_B}^T = [(\mu_\mathbf{1} - \mu_\mathbf{2})(\mu_\mathbf{1} - \mu_\mathbf{2})^T]^T$$

$$\mathbf{S_B} = \mathbf{S_B}^T$$

S_B is symmetric.

$$\mathbf{v^T S_B v} \geq 0$$

for any non-zero $\mathbf{v}$.

$$\mathbf{v^T}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^\mathbf{T}\mathbf{v} = \mathbf{v^T}(\mu_1 - \mu_2)^\mathbf{2}\mathbf{v} = \mathbf{v^T v}(\mu_1 - \mu_2)^\mathbf{2} \geq 0$$

Both $\mathbf{v^T v}$ and $(\mu_1 - \mu_2)^\mathbf{2}$ are ≥ 0.
S_B is PSD.

Exactly similar working can be used to prove that $\mathbf{S_W}$ is symmetric and PSD. Note that sum of two symmetric PSD matrices is also symmetric PSD, so it is sufficient to prove that S_i is symmetric PSD.

(c) To solve this equation and get the optimal projection matrix we just take the gradient and equate it to zero.

$$\frac{d}{dw}J(w) = \frac{d}{dw}(\frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}) = 0$$
$$(\mathbf{w}^T S_W \mathbf{w})\frac{d}{dw}(\mathbf{w}^T S_B \mathbf{w}) - (\mathbf{w}^T S_B \mathbf{w})\frac{d}{dw}(\mathbf{w}^T S_W \mathbf{w}) = 0$$
$$(\mathbf{w}^T S_W \mathbf{w})(2S_B \mathbf{w}) - (\mathbf{w}^T S_B \mathbf{w})(2S_W \mathbf{w}) = 0$$

Do some manipulation to get,

$$S_B \mathbf{w} - J(\mathbf{w})S_W \mathbf{w} = 0$$
$$S_W^{-1} S_B \mathbf{w} = J(\mathbf{w})\mathbf{w}$$

This is the eigenvector equation!
We can see that $J(\mathbf{w})$ are the eigenvalues. So the solution would be the eigenvectors of $S_W^{-1}S_B$ corresponding to the largest eigenvalues. **LDA Computation for the Given Dataset**

1) First, let's calculate the means for each class:

Class 0: $\mu_0 = \begin{pmatrix} \frac{1+2+3}{3} \\ \frac{1+2+4}{3} \end{pmatrix} = \begin{pmatrix} 2 \\ 2.33 \end{pmatrix}$

Class 1: $\mu_1 = \begin{pmatrix} \frac{8+7+8}{3} \\ \frac{8+10+7}{3} \end{pmatrix} = \begin{pmatrix} 7.67 \\ 8.33 \end{pmatrix}$

2) Calculate within-class scatter matrix $\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$:

For Class 0: $\mathbf{S}_0 = \sum_{i=1}^{3}(\mathbf{x}_i - \mu_0)(\mathbf{x}_i - \mu_0)^T$

$= \begin{pmatrix} -1 \\ -1.33 \end{pmatrix}\begin{pmatrix} -1 & -1.33 \end{pmatrix} + \begin{pmatrix} 0 \\ -0.33 \end{pmatrix}\begin{pmatrix} 0 & -0.33 \end{pmatrix} + \begin{pmatrix} 1 \\ 1.67 \end{pmatrix}\begin{pmatrix} 1 & 1.67 \end{pmatrix}$

For Class 1: $\mathbf{S}_1 = \sum_{i=1}^{3}(\mathbf{x}_i - \mu_1)(\mathbf{x}_i - \mu_1)^T$

$= \begin{pmatrix} 0.33 \\ -0.33 \end{pmatrix}\begin{pmatrix} 0.33 & -0.33 \end{pmatrix} + \begin{pmatrix} -0.67 \\ 1.67 \end{pmatrix}\begin{pmatrix} -0.67 & 1.67 \end{pmatrix} + \begin{pmatrix} 0.33 \\ -1.33 \end{pmatrix}\begin{pmatrix} 0.33 & -1.33 \end{pmatrix}$

After calculations: $\mathbf{S}_W = \begin{pmatrix} 2.67 & 1.33 \\ 1.33 & 9.33 \end{pmatrix}$

3) Calculate between-class scatter matrix $\mathbf{S}_B$: $\mathbf{S}_B = (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T$

$= \begin{pmatrix} 5.67 \\ 6 \end{pmatrix}\begin{pmatrix} 5.67 & 6 \end{pmatrix}$

$= \begin{pmatrix} 32.111 & 34 \\ 34 & 36 \end{pmatrix}$

4) Calculate $\mathbf{S}_W^{-1}\mathbf{S}_B$:

$\mathbf{S}_W^{-1} = \begin{pmatrix} 0.404 & -0.0577 \\ -0.0577 & 0.1154 \end{pmatrix}$

$\mathbf{S}_W^{-1}\mathbf{S}_B = \begin{pmatrix} 11 & 11.65 \\ 2.07 & 2.19 \end{pmatrix}$

5) Find the eigenvector corresponding to the largest eigenvalue:

The characteristic equation gives us eigenvalues: $\lambda_1 \approx 13.2$ (largest) $\lambda_2 \approx 0$

The eigenvector corresponding to λ_1 is: $\mathbf{w}^* \approx \begin{pmatrix} 0.983 \\ 0.185 \end{pmatrix}$

This eigenvector gives us the optimal projection direction for maximizing class separation while minimizing within-class scatter. The nearly equal components indicate that both features contribute almost equally to the discrimination between classes.

(d)

— End of Assignment —

Question	Points	Score
1	10	
2	15	
3	10	
4	15	
5	15	
6	10	
7	10	
8	15	
Total:	100	