Homework 3 Part 1

This is an individual assignment.

Write your answers using markdown cells or embed handwritten answers with `IPython.display.Image`.

Exercise 1 (8 points)

Consider a binary classification task where each sample $\mathbf{x}_i$ is d-dimensional with its corresponding label $t_i \in \{0, 1\}$. Suppose you have a dataset of N i.i.d. samples, $\{(x_i, t_i)\}_{i=1}^{N}$. Let $\mathbf{m}_i$ be the sample average for class C_i, $\mathbf{m}_i \in \mathbb{R}^d$.

The goal is to train a discriminant function of the form

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

to separate the two classes. Recall that the vector $\mathbf{w}$ is orthogonal to the discriminant function $y($$\mathbf{x}$ and that we are only interested in its direction not magnitude, thus the solution for $\mathbf{w}$ must satisfy the (equality) constraint $\mathbf{w}^T\mathbf{w} = 1$. Answer the following questions:

Suppose that you are interested in finding $\mathbf{w}$ and w_0 such that the projected class means are the most separable. Write down the objective function, the Lagrangian function, and derive for the analytical solutions for $\mathbf{w}$ and w_0.

Objective function:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T(\mathbf{m}_0 - \mathbf{m}_1)}{\|\mathbf{w}\|}$$

Equality constraint:

$$\mathbf{w}^T\mathbf{w} = 1$$

Lagrangian function:

$$\mathcal{L}(\mathbf{w}, \alpha) = \frac{\mathbf{w}^T(\mathbf{m}_0 - \mathbf{m}_1)}{\|\mathbf{w}\|} - \alpha(\mathbf{w}^T\mathbf{w} - 1)$$

Derive for $\mathbf{w}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\mathbf{m}_0 - \mathbf{m}_1}{\|\mathbf{w}\|} - 2\alpha\mathbf{w} = 0$$

$$\mathbf{w} = \frac{\mathbf{m}_0 - \mathbf{m}_1}{2\alpha}$$

Derive for w_0:

$$w_0 = -\frac{1}{2}(\mathbf{w}^T\mathbf{m}_0 + \mathbf{w}^T\mathbf{m}_1)$$

In [ ]:

In [ ]:

In [ ]:

Exercise 2 (8 points)

Consider a multi-class Logistic Regression classifier with a softmax activation function, $\phi(y_k(\mathbf{x})) = \frac{e^{y_k(\mathbf{x})}}{\sum_j e^{y_j(\mathbf{x})}}$. Using the gradient descent learning algorithm with batch learning, derive the update equations for $\mathbf{w}$ and w_0. Show your work.

Objective function for Logistic Regression:

$$J(\mathbf{w}) = -\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik} \log\left(\frac{e^{y_k(\mathbf{x}_i)}}{\sum_{j=1}^{K} e^{y_j(\mathbf{x}_i)}}\right)$$

N is the number of samples, K is the number of classes, t_{ik} is the indicator function for class membership, and $y_k(\mathbf{x}_i) = \mathbf{w}_k^T\mathbf{x}_i + w_{0k}$.

Derive for $\mathbf{w}_k$:

$$\frac{\partial J}{\partial \mathbf{w}_k} = -\sum_{i=1}^{N}\left(t_{ik} - \frac{e^{y_k(\mathbf{x}_i)}}{\sum_{j=1}^{K} e^{y_j(\mathbf{x}_i)}}\right)\mathbf{x}_i$$

Update equation for $\mathbf{w}_k$:

$$\mathbf{w}_k^{t+1} = \mathbf{w}_k^t - \eta\frac{\partial J}{\partial \mathbf{w}_k}$$

η is the learning rate.

Derive for w_{0k}:

$$\frac{\partial J}{\partial w_{0k}} = -\sum_{i=1}^{N}\left(t_{ik} - \frac{e^{y_k(\mathbf{x}_i)}}{\sum_{j=1}^{K} e^{y_j(\mathbf{x}_i)}}\right)$$

Update equation for w_{0k}:

$$w_{0k}^{t+1} = w_{0k}^{t} - \eta \frac{\partial J}{\partial w_{0k}}$$

These update equations are based on the gradient descent algorithm, where you adjust the parameters in the opposite direction of the gradient to minimize the loss function.

In [ ]:

In [ ]:

In [ ]:

Exercise 3 (8 points)

Answer the following questions:

1. (4 points) **What is the implication of the use of a single learning rate η for all samples $\mathbf{x}_i$ in gradient descent?**

1. If the data has varying characteristics, a single learning rate may cause the algorithm to converge slowly.
2. a fixed learning rate may lead to overshooting or divergence, especially if the data has features with different scales.
3. Gradient descent is sensitive to the scale of features. If the features have significantly different scales, using a single learning rate might result in inefficient learning.

In [ ]:

In [ ]:

In [ ]:

2. (4 points) **Consider a univariate training set $\{x_i\}_{i=1}^{N}$. Let $x \in [2, 4]$ belong to C_1 and $x < 2$ or $x > 4$ belong to C_2. How can we separate the two classes using a linear discriminant? Provide an example to solve this specific case.**

Define a threshold t such that if $x \leq t$, then x belongs to C_2, and if $x > t$, then x belongs to C_1.

In this case, where C_1 is for x in the range $[2, 4]$ and C_2 is for x outside this range, we can set the threshold t to be the midpoint of the range $[2, 4]$, which is $t = \frac{2+4}{2} = 3$. Therefore, if $x \leq 3$, it belongs to C_2, and if $x > 3$, it belongs to C_1.

linear discriminant:

$$\text{Classify } x \text{ as } \begin{cases} C_2, & \text{if } x \leq 3 \\ C_1, & \text{if } x > 3 \end{cases}$$

In [ ]:

In [ ]:

In [ ]:

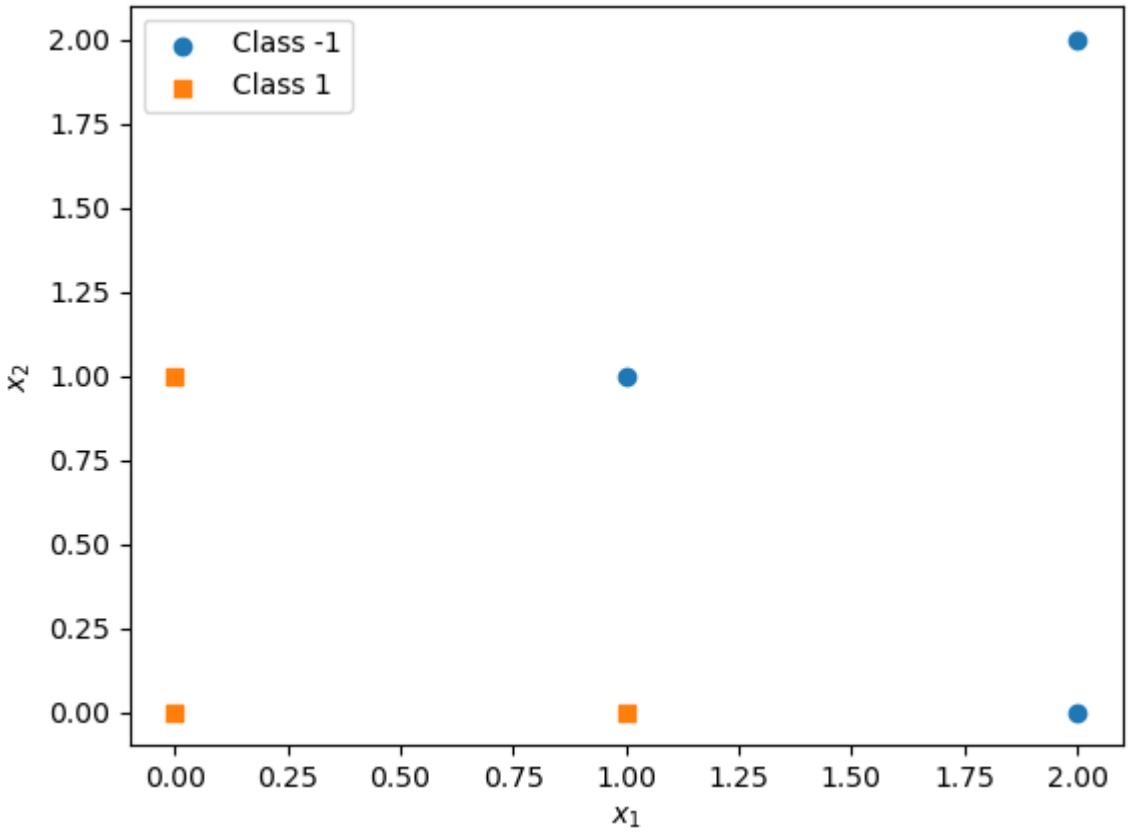
Exercise 4 (6 points)

Consider a Support Vector Machine (SVM) and the following training data, $\{(\mathbf{x}_i, t_i)\}_{i=1}^{6}$, from two classes, $t_i \in \{-1, 1\}$:

t	x1	x2
-1	1	1
-1	2	2
-1	2	0
1	0	0
1	1	0
1	0	1

1. (3 points) **Plot these six training points, and construct by inspection the weight vector for the optimal hyperplane, and the optimal margin.**

In [3]:
```
import matplotlib.pyplot as plt

class_minus1 = [(1, 1), (2, 2), (2, 0)]
class_1 = [(0, 0), (1, 0), (0, 1)]

plt.scatter(*zip(*class_minus1), marker='o', label='Class -1')
plt.scatter(*zip(*class_1), marker='s', label='Class 1')

plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.legend()
plt.show()
```



By inspection, a line passing through (1,1) and (0,0) can separate the classes with a margin. This line corresponds to the equation $x_1 - x_2 = 0$.

The weight vector $\mathbf{w}$ for this hyperplane is $[1, -1]$, and the optimal margin is the perpendicular distance from the hyperplane to the nearest data point. Here, the nearest data point is (1,1), and the margin is the distance from (1,1) to the hyperplane $x_1 - x_2 = 0$.

Optimal weight vector $\mathbf{w}$: $[1, -1]$ Optimal margin: $1/\sqrt{2}$

In [ ]:

In [ ]:

2. (3 points) **What are the support vectors?**

The support vectors are the points that are closest to the decision boundary. In this case, the points (1,1) from class -1 and (1,0) from class 1 are likely to be support vectors, as they are nearest to the hyperplane $x_1 - x_2 = 0$.

In [ ]:

In [ ]:

In [ ]:

3. OPTIONAL. Extra Credit (up to 4.5 points) **Construct the solution in the dual space by finding the Lagrange (undetermined) multipliers, α_i. Compare your result to that in part 1.**

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Exercise 5 (15 points)

Consider the following dataset

x1	x2	t
1	0	1
4	2	1
0	-1	-1
-1	-1	-1

x1	x2	t
-2	1	-1

1. (5 points) **Consider the *Fisher's LDA classifier* and answer the following sub-parts:**
   A. (2.5 points) **Compute the between-class scatter matrix, S_B, and the within-class scatter matrix, S_W.**
   B. (2.5 points) **Find w and w_0. Draw the respective discriminant function.**

$$S_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$$

$\mathbf{m}_1$ and $\mathbf{m}_2$ are the mean vectors of the two classes.

The within-class scatter matrix is given by:

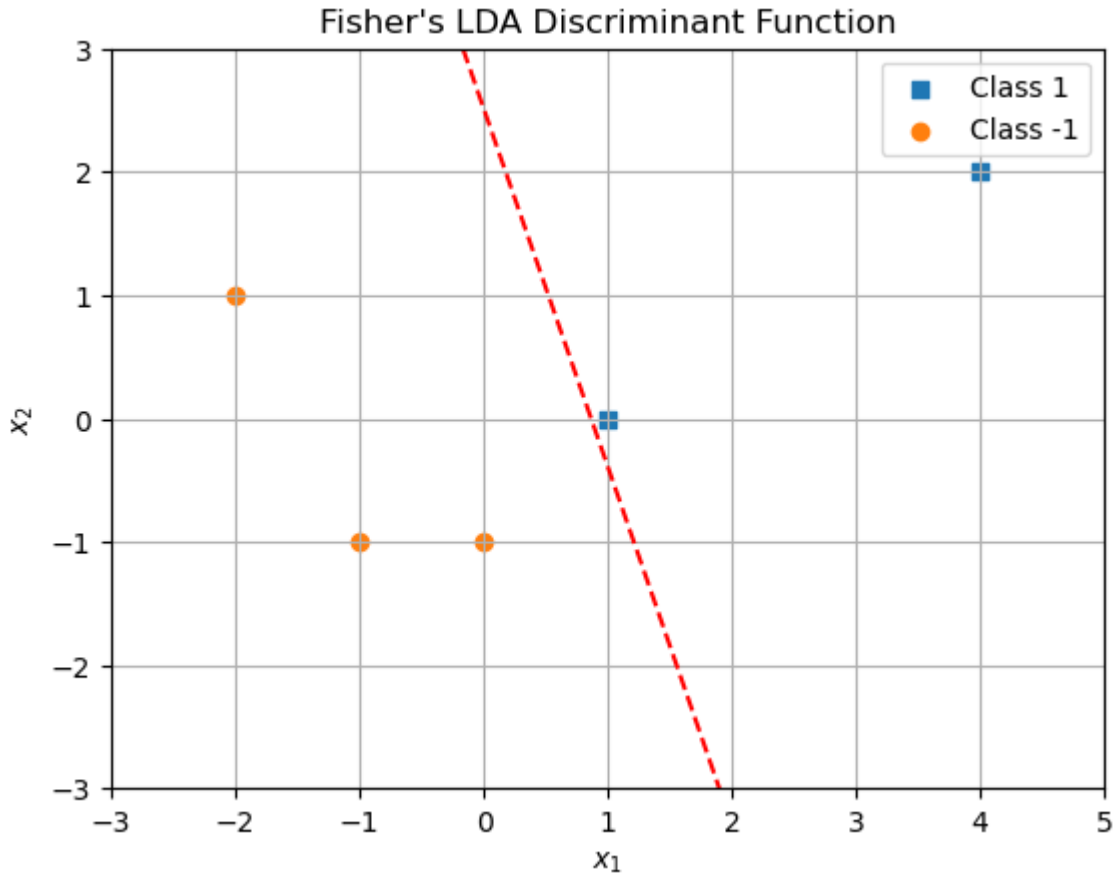
$$S_W = \sum_{i=1}^{N} (X_i - \mathbf{m}_{t_i})(X_i - \mathbf{m}_{t_i})^T$$

$$\mathbf{m}_1 = \frac{1}{2}((1,0) + (4,2)) = (2.5, 1)$$

$$\mathbf{m}_{-1} = \frac{1}{3}((0,-1) + (-1,-1) + (-2,1)) = (-1, -0.33)$$

$$\begin{aligned} S_B &= (\mathbf{m}_1 - \mathbf{m}_{-1})(\mathbf{m}_1 - \mathbf{m}_{-1})^T \\ S_W &= (1-2.5)^2 + (0-1)^2 + (4-2.5)^2 + (2-1)^2 + (0-2.5)^2 + (-1-1)^2 + (-1+1) \\ &= (1.5^2 + 1^2) + (1.5^2 + 1^2) + (2.5^2 + 1^2) + (1^2 + 1^2) + (2.5^2 + 2.33^2) \\ &= 2 \times (1.5^2 + 1^2) + 2 \times (2.5^2 + 1^2) + 2.33^2 + 2.33^2 \\ &= 15.5 \end{aligned}$$

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt

X = np.array([[1, 0], [4, 2], [0, -1], [-1, -1], [-2, 1]])
t = np.array([1, 1, -1, -1, -1])

m1 = np.mean(X[t == 1], axis=0)
m_minus1 = np.mean(X[t == -1], axis=0)

S_B = np.outer((m1 - m_minus1), (m1 - m_minus1))
S_W = np.sum([np.outer((X[i] - (m1 if t[i] == 1 else m_minus1)), (X[i] - (m1 if t[i]

w = np.linalg.inv(S_W).dot((m1 - m_minus1))

w_0 = -0.5 * (m1 + m_minus1).dot(w)

def discriminant_function(x):
    return w.dot(x) + w_0

x_vals = np.linspace(-3, 5, 100)
y_vals = np.linspace(-3, 3, 100)

x_mesh, y_mesh = np.meshgrid(x_vals, y_vals)
xy_vals = np.vstack((x_mesh.ravel(), y_mesh.ravel())).T
z_vals = np.array([discriminant_function(xy) for xy in xy_vals]).reshape(x_mesh.shape)

plt.scatter(X[t == 1][:, 0], X[t == 1][:, 1], marker='s', label='Class 1')
plt.scatter(X[t == -1][:, 0], X[t == -1][:, 1], marker='o', label='Class -1')

contour = plt.contour(x_mesh, y_mesh, z_vals, levels=[0], colors='red', linestyles='da


plt.legend(handles=[contour.collections[0]], labels=['Decision Boundary'])

plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.legend()
plt.grid(True)
plt.title("Fisher's LDA Discriminant Function")

plt.show()
```



In [ ]:

In [ ]:

2. The projection vector, $\mathbf{w}$, is the eigenvector of $S_W^{-1} S_B$ with the largest eigenvalue, and $w_0 = \frac{\mathbf{w}^T(\mathbf{m}_1+\mathbf{m}_2)}{2}$.

In [10]:
```
X = np.array([[1, 0], [4, 2], [0, -1], [-1, -1], [-2, 1]])
t = np.array([1, 1, -1, -1, -1])

m1 = np.mean(X[t == 1], axis=0)
m_minus1 = np.mean(X[t == -1], axis=0)

S_B = np.outer((m1 - m_minus1), (m1 - m_minus1))
S_W = np.sum([np.outer((X[i] - (m1 if t[i] == 1 else m_minus1)), (X[i] - (m1 if t[i]

eigenvalues, eigenvectors = np.linalg.eig(np.linalg.inv(S_W).dot(S_B))

max_eigenvalue_index = np.argmax(eigenvalues)

w = eigenvectors[:, max_eigenvalue_index]

w_0 = 0.5 * w.dot(m1 + m_minus1)

print("Projection Vector w:", w)
print("w_0:", w_0)
```

```
Projection Vector w: [0.94548457 0.32566691]
w_0: 0.8176690605705075
```

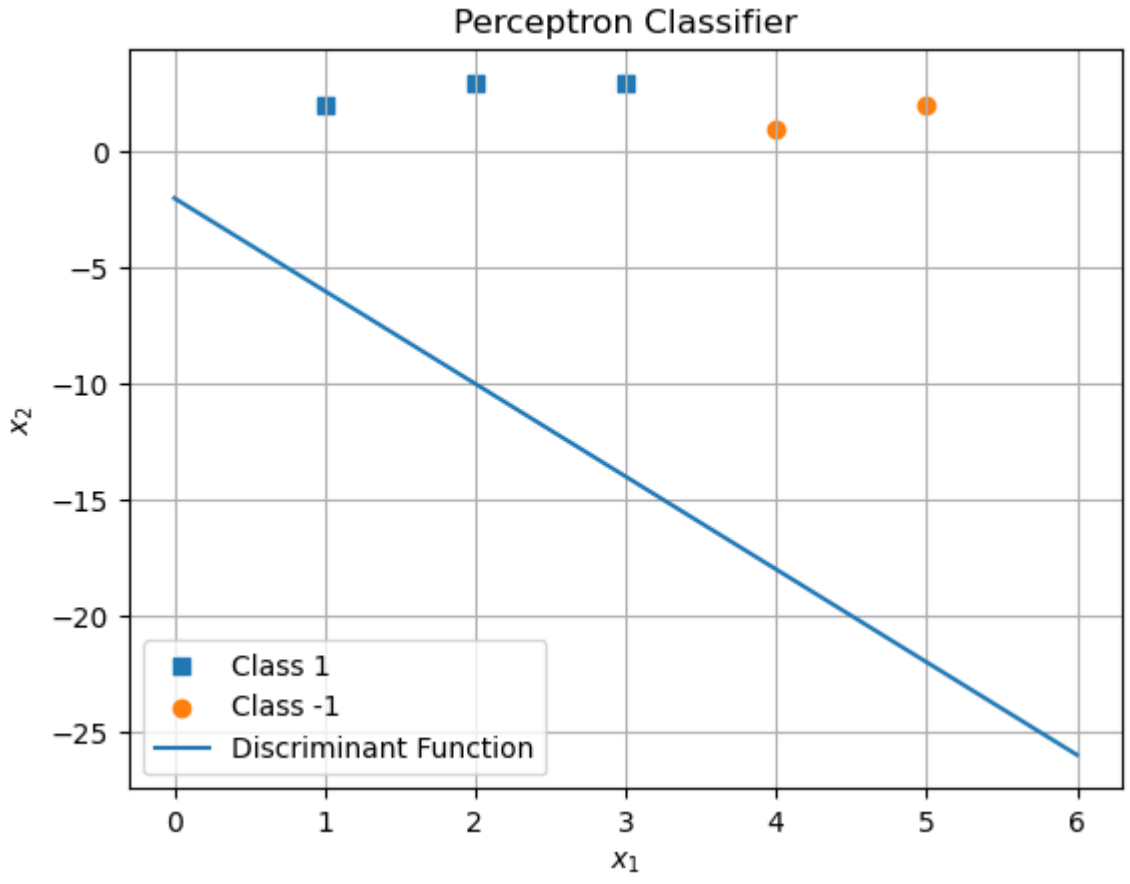
In [ ]:

In [ ]:

In [ ]:

2. (5 points) **Consider the Perceptron classifier with the initial weights $w = [4, 1]$ and intercept $w_0 = 2$. Answer the following sub-parts:**
   A. (2 points) **Draw the samples and the corresponding discriminant function.**
   B. (3 points) **What is the smallest value for the learning rate η such that the updated Perceptron will result in zero misclassified points using only one iteration?**

In [12]:
```
X = np.array([[1, 2], [2, 3], [3, 3], [4, 1], [5, 2]])
y = np.array([1, 1, 1, -1, -1])

w = np.array([4, 1])
w0 = 2

def discriminant_function(x):
    return np.dot(w, x) + w0

plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], marker='s', label='Class 1')
plt.scatter(X[y == -1][:, 0], X[y == -1][:, 1], marker='o', label='Class -1')

x_vals = np.linspace(0, 6, 100)
y_vals = -(w[0] * x_vals + w0) / w[1]  # Rearrange the equation for a line
plt.plot(x_vals, y_vals, label='Discriminant Function')

plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.legend()
plt.grid(True)
plt.title('Perceptron Classifier')

plt.show()
```



B. $\eta \sum_{i \in \text{misclassified}} y_i \mathbf{x}_i = -\mathbf{w}_{\text{old}}$

In [ ]:

In [ ]:

3. (5 points) **Consider the Logistic Regression classifier with the initial weights $w = [4, 1]$ and intercept $w_0 = 2$. Let a label $t = -1$ be mapped to $t = 0$. Answer the following sub-parts:**
   A. (2 points) **Use the initial parameter values to predict the label for each samples above.**
   B. (3 points) **What is the smallest value for the learning rate η such that the updated Logistic Regression will result in zero misclassified points using only one iteration?**

A. $\text{Predicted Label}(\mathbf{x}_i) = \text{round}\left(\frac{1}{1+\exp\left(-(w^T\mathbf{x}_i+w_0)\right)}\right)$

B. $\eta = \frac{\|\mathbf{w}_{\text{old}}-\mathbf{w}_{\text{true}}\|}{\|\nabla E(\mathbf{w}_{\text{old}})\|}$

$\nabla E(\mathbf{w}_{\text{old}})$ is the gradient of the error with respect to the weights, and $\mathbf{w}_{\text{true}}$ is the true weight vector that results in zero misclassified points.

In [ ]:

In [ ]:

On-Time (5 points)

Submit your assignment before the deadline.

Submit Your Solution

Confirm that you've successfully completed the assignment.

Along with the Notebook, include a PDF of the notebook with your solutions.

`add` and `commit` the final version of your work, and `push` your code to your GitHub repository.

Submit the URL of your GitHub Repository as your assignment submission on Canvas.