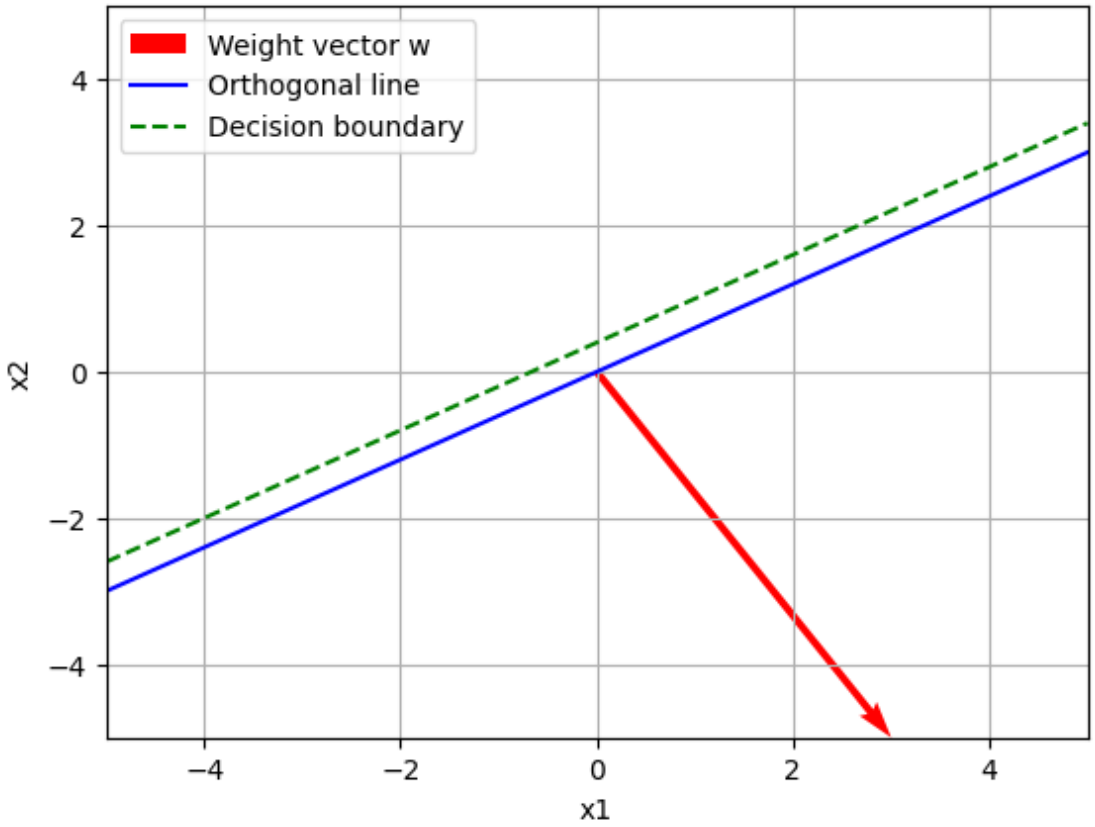
Northeastern University

CS 5100: Problem Set 3 - Machine Learning

Foundations of Artificial Intelligence

Anuj Patel (NU ID: 002874710)

Que.1



Que.2

1. **Determine the eq of the line:**
   The line go through the points $(0.33333, 0)$ and $(0, 2)$.Equation of a line

$$(y - y_1) = m(x - x_1)$$

m is the slope and (x_1, y_1) is a point on the line.

2. **Calculate slope:**

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{2 - 0}{0 - 0.33333} = -6$$

3. **Substitute the slope and one point into eq:**

$$(y - 0) = -6(x - 0.33333)$$

$$y = -6x + 2$$

4. **so updated equation in the form $\mathbf{w}^T\mathbf{x} + b = 0$:**

$$6x + y - 2 = 0$$

final weight and bias are:

$$\mathbf{w} = \begin{bmatrix} 6 \\ 1 \end{bmatrix}, \quad b = -2$$

Que.3

1. Classification of Point at $(1, 1)$

Decision Boundary Eq:

$$\mathbf{w}^T\mathbf{x} + b = 0 \implies 6x + y - 2 = 0$$

Classification condition:

$$\text{Class} = \begin{cases} \text{Positive} & \text{if } 6x + y - 2 > 0 \\ \text{Negative} & \text{if } 6x + y - 2 < 0 \end{cases}$$

for $(1, 1)$**:**

$$f(1, 1) = 6(1) + 1 - 2 = 5 > 0$$

Result: This is classified as **Positive** class sample.

2. Reasoning Behind the Classification

- $f(1, 1) = 5$, greater than zero.
- a positive value indicates the point lies on the positive side of the decision boundary.
- Therefore $(1, 1)$ is labeled as a **Positive** class sample.

3. changing classification of $(1, 1)$ to opposite

To **flip the classification** of $(1, 1)$ from **Positive** to **Negative** without changing the decision boundary, I will **invert the decision function**. Multiplying weights and the bias by **-1**.

Updated Weights and Bias:

$$\mathbf{w}' = -\mathbf{w} = [-6, -1]$$

$$b' = -b = 2$$

New Decision Function:

$$\mathbf{w}'^T\mathbf{x} + b' = -6x - y + 2 = 0 \implies 6x + y - 2 = 0$$

$$f'(1, 1) = -6(1) - 1 + 2 = -5 < 0$$

Ans: With the updated weights and bias this now classified as a **Negative** class sample.

Que.4

Step 1: Regularization Term

$$\|w\|^2 = 3^2 + (-1)^2 = 9 + 1 = 10$$

$$\frac{1}{2}\|w\|^2 = \frac{1}{2} \times 10 = 5$$

Step 2: Calculate the Hinge Loss for Each Data Point

$$\text{Loss}_i = \max\left(0, 1 - y_i(w \cdot x_i + b)\right)$$

1. For $x_1 = (3, 1)$, $y_1 = -1$:

$$w \cdot x_1 + b = 3 \cdot 3 + (-1) \cdot 1 + (-13.5) = 9 - 1 - 13.5 = -5.5$$

$$y_1(w \cdot x_1 + b) = -1 \times (-5.5) = 5.5$$
$$\text{Loss}_1 = \max(0, 1 - 5.5) = \max(0, -4.5) = 0$$

2. For $x_2 = (4, 2)$, $y_2 = -1$:

$$w \cdot x_2 + b = 3 \cdot 4 + (-1) \cdot 2 + (-13.5) = 12 - 2 - 13.5 = -3.5$$

$$y_2(w \cdot x_2 + b) = -1 \times (-3.5) = 3.5$$
$$\text{Loss}_2 = \max(0, 1 - 3.5) = \max(0, -2.5) = 0$$

3. For $x_3 = (5, 2)$, $y_3 = +1$:

$$w \cdot x_3 + b = 3 \cdot 5 + (-1) \cdot 2 + (-13.5) = 15 - 2 - 13.5 = -0.5$$

$$y_3(w \cdot x_3 + b) = 1 \times (-0.5) = -0.5$$
$$\text{Loss}_3 = \max(0, 1 - (-0.5)) = \max(0, 1.5) = 1.5$$

4. For $x_4 = (5, 1)$, $y_4 = +1$:

$$w \cdot x_4 + b = 3 \cdot 5 + (-1) \cdot 1 + (-13.5) = 15 - 1 - 13.5 = 0.5$$

$$y_4(w \cdot x_4 + b) = 1 \cdot 0.5 = 0.5$$
$$\text{Loss}_4 = \max(0, 1 - 0.5) = \max(0, 0.5) = 0.5$$

5. For $x_5 = (6, 2)$, $y_5 = +1$:

$$w \cdot x_5 + b = 3 \cdot 6 + (-1) \cdot 2 + (-13.5) = 18 - 2 - 13.5 = 2.5$$

$$y_5(w \cdot x_5 + b) = 1 \cdot 2.5 = 2.5$$
$$\text{Loss}_5 = \max(0, 1 - 2.5) = \max(0, -1.5) = 0$$

Step 3: Total Hinge Losses

$$\sum_{i=1}^{5} \text{Loss}_i = 0 + 0 + 1.5 + 0.5 + 0 = 2.0$$

Step 4:Total Training Loss

$$L(X, w, b) = 5 + \frac{1}{5} \times 2.0 = 5 + 0.4 = 5.4$$

Final Answer:
The training loss is **5.4**.

Que.5:

1. Gradient of Regularization Term:

The regularization term is:

$$\frac{1}{2}\|w\|^2 = \frac{1}{2}(3^2 + (-1)^2) = 5$$

Gradient w.r.t. w:

$$\nabla_w(\text{Regularization}) = w = [3, -1]$$

Gradient w.r.t. b:

$$\nabla_b(\text{Regularization}) = 0$$

2. Gradient of Hinge Loss Term:

hinge loss for each point is:

$$\text{Hinge Loss} = \max(0, 1 - y_i(w \cdot x_i + b))$$

Will consider only Non-zero hinge loss points:

- For $x_3 = (5, 2), y_3 = +1$:

$$w \cdot x_3 + b = -0.5, \text{ Loss} = 1.5, \text{ Gradient w.r.t. } w = [-5, -2], \text{ w.r.t. } b = -1$$

- For $x_4 = (5, 1), y_4 = +1$:

$$w \cdot x_4 + b = 0.5, \text{ Loss} = 0.5, \text{ Gradient w.r.t. } w = [-5, -1], \text{ w.r.t. } b = -1$$

Total hinge loss gradients:

$$\nabla_w(\text{Hinge Loss}) = \frac{1}{5}([-5, -2] + [-5, -1]) = [-2, -0.6]$$

$$\nabla_b(\text{Hinge Loss}) = \frac{1}{5}(-1 - 1) = -0.4$$

3. Total Gradient:

Combine regularization and hinge loss gradients:

- Gradient w.r.t. w:

$$\nabla_w L = \nabla_w(\text{Regularization}) + \nabla_w(\text{Hinge Loss}) = [3, -1] + [-2, -0.6] = [1, -1.6]$$

- Gradient w.r.t. b:

$$\nabla_b L = \nabla_b(\text{Regularization}) + \nabla_b(\text{Hinge Loss}) = 0 + (-0.4) = -0.4$$

Final Answer:

$$\nabla_w L = [1, -1.6], \quad \nabla_b L = -0.4$$

Que.6

gradient eq:

for a single data point:

$$\delta_i = \begin{cases} 1, & \text{if } y_i(w \cdot x_i + b) < 1 \\ 0, & \text{otherwise} \end{cases}$$

gradien:

$$\frac{\partial L_i}{\partial w} = w - \frac{C}{N}\delta_i y_i x_i$$
$$\frac{\partial L_i}{\partial b} = -\frac{C}{N}\delta_i y_i$$

For Point 1: $x_1 = (3, 1),\ y_1 = -1$

Step 1: find $y_1(w \cdot x_1 + b)$

$$w \cdot x_1 + b = 3 \times 3 + (-1) \times 1 - 13.5 = -5.5$$
$$y_1(w \cdot x_1 + b) = (-1)(-5.5) = 5.5$$

Since $y_1(w \cdot x_1 + b) > 1$, $\delta_1 = 0$.

Step 2: Gradients

$$\frac{\partial L_1}{\partial w} = w - \frac{C}{N}\delta_1 y_1 x_1 = [3, -1] - 0 = [3, -1]$$
$$\frac{\partial L_1}{\partial b} = -\frac{C}{N}\delta_1 y_1 = 0$$

Step 3: Update Weights and Bias

$$w_{\text{new}} = w_{\text{old}} - \alpha\frac{\partial L_1}{\partial w} = [3, -1] - 0.01 \times [3, -1] = [2.97, -0.99]$$
$$b_{\text{new}} = b_{\text{old}} - \alpha\frac{\partial L_1}{\partial b} = -13.5 - 0 = -13.5$$

point 2: $x_2 = (4, 2),\ y_2 = -1$

Step 1

$$w \cdot x_2 + b = 2.97 \times 4 + (-0.99) \times 2 - 13.5 = -3.6$$
$$y_2(w \cdot x_2 + b) = (-1)(-3.6) = 3.6$$

Since $y_2(w \cdot x_2 + b) > 1$, $\delta_2 = 0$.

Step 2

$$\frac{\partial L_2}{\partial w} = w - \frac{C}{N}\delta_2 y_2 x_2 = [2.97, -0.99] - 0 = [2.97, -0.99]$$
$$\frac{\partial L_2}{\partial b} = 0$$

Step 3

$$w_{\text{new}} = [2.97, -0.99] - 0.01 \times [2.97, -0.99] = [2.9403, -0.9801]$$
$$b_{\text{new}} = -13.5$$

point 3: $x_3 = (5, 2),\ y_3 = +1$

Step 1

$$w \cdot x_3 + b = 2.9403 \times 5 + (-0.9801) \times 2 - 13.5 = -0.7587$$
$$y_3(w \cdot x_3 + b) = (1)(-0.7587) = -0.7587$$

Since $y_3(w \cdot x_3 + b) < 1$, $\delta_3 = 1$.

Step 2

$$\frac{\partial L_3}{\partial w} = w - \frac{C}{N} y_3 x_3 = [2.9403, -0.9801] - \frac{1}{5} \times [5, 2]$$
$$= [2.9403, -0.9801] - [1, 0.4] = [1.9403, -1.3801]$$
$$\frac{\partial L_3}{\partial b} = -\frac{C}{N} y_3 = -\frac{1}{5} \times 1 = -0.2$$

Step 3

$$w_{\text{new}} = [2.9403, -0.9801] - 0.01 \times [1.9403, -1.3801]$$
$$= [2.920897, -0.966299]$$
$$b_{\text{new}} = -13.5 - 0.01 \times (-0.2) = -13.5 + 0.002 = -13.498$$

Final Weights and Bias:

1. **After** $x_1 = (3, 1)$**:**
$$w = [2.97, -0.99], \quad b = -13.5$$
2. **After** $x_2 = (4, 2)$**:**
$$w = [2.9403, -0.9801], \quad b = -13.5$$
3. **After** $x_3 = (5, 2)$**:**
$$w = [2.920897, -0.966299], \quad b = -13.498$$

Que7

Step 1: calculate z

Calculate the dot product of w and x_i,:

$$z = w \cdot x_i + b$$
$$= (0.5)(6) + (0.1)(4) + (-0.8)(2) + (0.9)(3) + (0.0)(1) + 0.05$$
$$= 4.55$$

Step 2:Sigmoid Function

apply the sigmoid function to z to get the probability for class 1:

$$\sigma(4.55) = \frac{1}{1 + e^{-4.55}}$$
$$\approx 0.9895$$

Step 3:Probability Vector

probabilities for classes 0 and 1 are:

$$f(x_i) = [1 - \sigma(z),\ \sigma(z)] = [1 - 0.9895,\ 0.9895] = [0.0105,\ 0.9895]$$

Step 4: CrossEntropy loss

Use the cross-entropy loss formula:

$$\begin{aligned} L(x_i, y_i) &= -\left(y_{i,0} \cdot \log f_0(x_i) + y_{i,1} \cdot \log f_1(x_i)\right) \\ &= -\left(0 \times \log(0.0105) + 1 \times \log(0.9895)\right) \\ &= 0.0105 \end{aligned}$$

Answer:

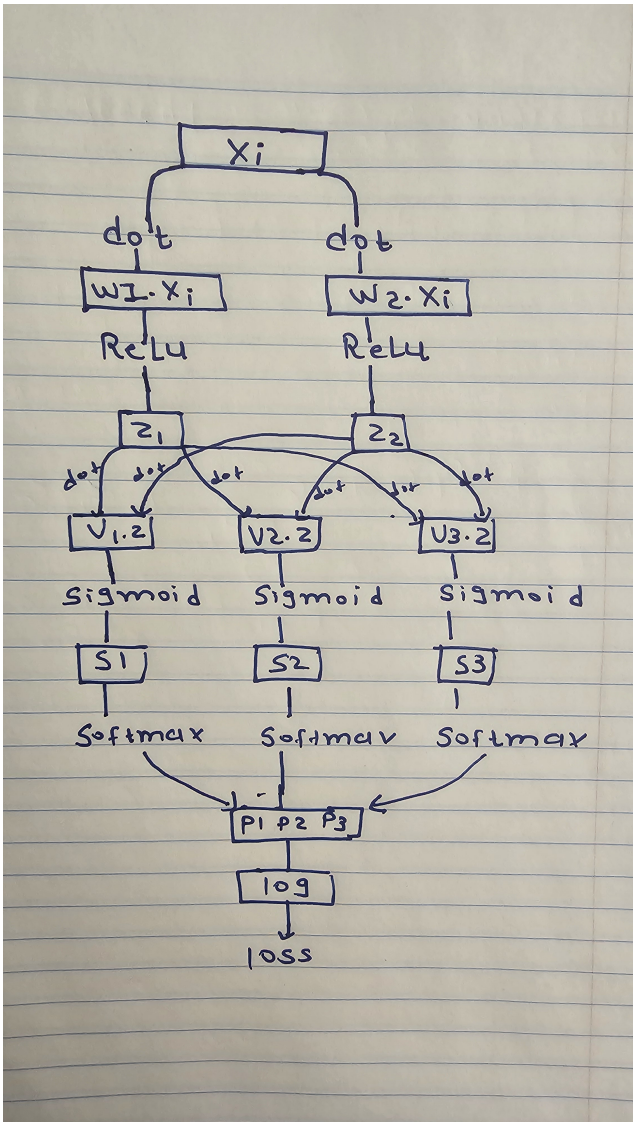
loss is approximately **0.0105**.

Que.8

```python
import numpy as np
from scipy.special import expit

# Hidden layer
xi = np.array([5, 7, 4, 3, 2])
w1 = np.array([0, 2, -4, -5, 4])
w2 = np.array([2, 1, -4, 3, 0])
z1 = np.maximum(0, np.dot(w1, xi) + 2)
z2 = np.maximum(0, np.dot(w2, xi) - 1)

# Output layer
v1 = np.array([-1, 2])
v2 = np.array([2, 4])
v3 = np.array([1, 3])
output1 = expit(np.dot(v1, np.array([z1, z2])) + 1)
output2 = expit(np.dot(v2, np.array([z1, z2])) + 0)
output3 = expit(np.dot(v3, np.array([z1, z2])) + 4)

# Softmax and loss
outputs = np.array([output1, output2, output3])
softmax_probs = np.exp(outputs) / np.sum(np.exp(outputs))
loss = -np.log(softmax_probs[2])  # For true class 2
print(loss)
```

Final Answer is: 1.098612286800534

Que.9


Xi
dot
dot
W1.Xi
W2.Xi
ReLu
ReLu
Z1
Z2
dot
dot
dot
dot
dot
dot
V1.2
V2.2
V3.2
Sigmoid
Sigmoid
Sigmoid
S1
S2
S3
Softmax
Softmax
Softmax
P1 P2 P3
log
loss

Que9.py > ...

```python
import numpy as np
from scipy.special import expit  # Sigmoid function
# Ensure double precision
dtype = np.float64
# Input vector
xi = np.array([5, 7, 4, 3, 2], dtype=dtype)
w1 = np.array([0, 2, -4, -5, 4], dtype=dtype)
w2 = np.array([2, 1, -4, 3, 0], dtype=dtype)

# Hidden layer pre-activations
a1 = np.dot(w1, xi)  # No bias term
a2 = np.dot(w2, xi)  # No bias term

# Hidden layer activations (ReLU)
z1 = np.maximum(0, a1)
z2 = np.maximum(0, a2)
z = np.array([z1, z2], dtype=dtype)

# Output layer weights (ignoring biases)
v1 = np.array([-1, 2], dtype=dtype)
v2 = np.array([2, 4], dtype=dtype)
v3 = np.array([1, 3], dtype=dtype)

# Output layer pre-activations
o1 = np.dot(v1, z)
o2 = np.dot(v2, z)
o3 = np.dot(v3, z)

# Sigmoid activations
s1 = expit(o1)
s2 = expit(o2)
s3 = expit(o3)
s = np.array([s1, s2, s3], dtype=dtype)


# Softmax probabilities
exp_s = np.exp(s)
softmax_denominator = np.sum(exp_s)
softmax_probs = exp_s / softmax_denominator

```

```python
softmax_probs = exp_s / softmax_denominator

# Cross-entropy loss
true_class = 2  # Zero-based index
loss = -np.log(softmax_probs[true_class])

# Gradient initialization
dL_ds = softmax_probs.copy()
dL_ds[true_class] -= 1  # dL/ds_i = p_i - y_i

# Sigmoid derivatives
ds_do = s * (1 - s)  # Element-wise multiplication

# Gradients w.r.t. output pre-activations
dL_do = dL_ds * ds_do

# Gradients w.r.t. output weights (v1, v2, v3)
dL_dv1 = dL_do[0] * z
dL_dv2 = dL_do[1] * z
dL_dv3 = dL_do[2] * z

# Stack gradients for output weights
dL_dv = [dL_dv1, dL_dv2, dL_dv3]
# Gradients w.r.t. hidden activations z
dL_dz = (dL_do[0] * v1) + (dL_do[1] * v2) + (dL_do[2] * v3)
# Gradients through ReLU
dz_da = (a1 > 0).astype(dtype), (a2 > 0).astype(dtype)  # Derivative of ReLU

# Gradients w.r.t. hidden pre-activations
dL_da1 = dL_dz[0] * dz_da[0]
dL_da2 = dL_dz[1] * dz_da[1]

# Gradients w.r.t. input weights (w1, w2)
dL_dw1 = dL_da1 * xi
dL_dw2 = dL_da2 * xi
# Display gradients
print("Gradient w.r.t w1:", dL_dw1)
print("Gradient w.r.t w2:", dL_dw2)
print("Gradient w.r.t v1:", dL_dv1)
print("Gradient w.r.t v2:", dL_dv2)
print("Gradient w.r.t v3:", dL_dv3)
```

Final Ans

```
Gradient w.r.t w1: [-0. -0. -0. -0. -0.]
Gradient w.r.t w2: [6.86957748e-09 9.61740847e-09 5.49566198e-09 4.12174649e-09
 2.74783099e-09]
Gradient w.r.t v1: [0.00000000e+00 6.87051228e-09]
Gradient w.r.t v2: [0. 0.]
Gradient w.r.t v3: [-0.00000000e+00 -6.23205191e-13]
PS C:\Users\Anuj\OneDrive - Northeastern University\Desktop\Programs>
```

Que.10

(a) When the binary classification problem has one class with a double number of instances of the other class in the training and test datasets, accuracy as the sole metric can be misleading, since such a model will always easily achieve a high accuracy by always predicting the majority class without really learning how to distinguish between classes.

(b) To overcome this drawback, one may use a measure of accuracy by calculating the *balanced accuracy*: calculating the accuracy of each class separately and then averaging them. In this way, both classes are weighed equally and, in so doing, provide a more perspicacious performance measurement of the model.

(c) Other evaluation methods for metrics like accuracy, precision, recall, F1-score, and the area under the ROC curve. These metrics focus on different parts of the model predictions, such as how well it identifies positive cases or how precise those identifications are, helping better to evaluate the model in detail. This also helps to tune the hyperparameters.