Exercise 1: Perceptron Learning with Multiple Updates

You have a perceptron with initial weights $w = [0, 0]$ and bias $b = 0$. The dataset is:

x_1	x_2	y
2	3	1
-1	-2	-1
1	-1	-1
-2	2	1

Use the perceptron update rule with learning rate $\eta = 1$ for two epochs. Update w and b accordingly.

Solution

The perceptron update rule is:

$$w = w + \eta y x, \quad b = b + \eta y$$

Epoch 1

1. **Classifying (2,3), y = 1**

   $$y_{\text{pred}} = \text{sign}(0 \cdot 2 + 0 \cdot 3 + 0) = 0 \quad \text{(Misclassified)}$$

   Update:

   $$w = [0, 0] + (1)(1)[2, 3] = [2, 3], \quad b = 0 + 1 = 1$$

2. **Classifying (-1,-2), y = -1**

   $$y_{\text{pred}} = \text{sign}(2(-1) + 3(-2) + 1) = \text{sign}(-2 - 6 + 1) = \text{sign}(-7) = -$$

   Correct classification (no update).

3. **Classifying (1,-1), y = -1**

   $$y_{\text{pred}} = \text{sign}(2(1) + 3(-1) + 1) = \text{sign}(2 - 3 + 1) = \text{sign}(0) = 0$$

   Misclassified, update:

   $$w = [2, 3] + (-1)[1, -1] = [1, 4], \quad b = 1 - 1 = 0$$

4. **Classifying (-2,2), y = 1**

   $$y_{\text{pred}} = \text{sign}(1(-2) + 4(2) + 0) = \text{sign}(-2 + 8) = \text{sign}(6) = 1$$

   Correct classification (no update).

Epoch 2

Repeat the process, but in this case, all points are correctly classified, so no further updates.

Final weights: $w = [1, 4]$, $b = 0$.

Exercise 2: Perceptron Learning Rule

Consider a perceptron with the weight vector $w = [1, -1]$ and bias $b = 0$. Given a dataset with the following points:

$$(2, 3) \rightarrow y = 1, \quad (1, -1) \rightarrow y = -1$$

Update the weights using the perceptron learning rule for each misclassified point with a learning rate $\eta = 1$.

Solution

1. **Checking classification of (2,3)**

   $$y_{\text{pred}} = \text{sign}(w_1 \cdot x_1 + w_2 \cdot x_2 + b) = \text{sign}(1(2) + (-1)(3) + 0) = \text{sign}(-1) = -1$$

   Since $y_{\text{pred}} \neq y_{\text{true}}$, update weights:

   $$w = w + \eta y x = [1, -1] + (1)[2, 3] = [3, 2]$$

2. **Checking classification of (1,-1)**

   $$y_{\text{pred}} = \text{sign}(3(1) + 2(-1) + 0) = \text{sign}(3 - 2) = \text{sign}(1) = 1$$

   Since $y_{\text{pred}} \neq y_{\text{true}}$, update weights:

   $$w = w + \eta y x = [3, 2] + (-1)[1, -1] = [3 - 1, 2 + 1] = [2, 3]$$

Final updated weights: $w = [2, 3]$.

Exercise 2: Logistic Regression – One Step of Gradient Descent

Given a logistic regression model with:

$$h(x) = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + b)}}$$

For a single data point $(x_1, x_2) = (2, 3)$, with label $y = 1$, initial weights $w = [0.5, -0.5]$, bias $b = 0$, and learning rate $\eta = 0.1$, compute one update step using gradient descent.

Solution

Step 1: Compute the Linear Combination (Logits)

The linear combination before applying the sigmoid function is:

$$z = w_1 x_1 + w_2 x_2 + b$$

Substituting the given values:

$$z = (0.5 \cdot 2) + (-0.5 \cdot 3) + 0$$

$$= 1 - 1.5 = -0.5$$

Step 2: Compute the Sigmoid Function

The logistic (sigmoid) function is:

$$h(x) = \frac{1}{1 + e^{-z}}$$

Substituting $z = -0.5$:

$$h(x) = \frac{1}{1 + e^{0.5}}$$

Approximating $e^{0.5} \approx 1.6487$:

$$h(x) = \frac{1}{1 + 1.6487} = \frac{1}{2.6487} \approx 0.3775$$

This is the predicted probability for $y = 1$.

The **cost function $J(w, b)$ for logistic regression** is the **log loss (cross-entropy loss)**

$$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)}) + (1-y^{(i)})\log(1-h(x^{(i)}))\right]$$

For a **single training example** (as in our exercise), the cost function simplifies to:

$$J(w,b) = -\left[y\log h(x) + (1-y)\log(1-h(x))\right]$$

where:

- y is the true label (0 or 1),
- $h(x)$ is the predicted probability from the sigmoid function.

Gradient of $J(w, b)$

To minimize $J(w, b)$, we compute its gradient with respect to the parameters:

Gradient w.r.t. w_j (weights):

$$\frac{\partial J}{\partial w_j} = (h(x) - y)x_j$$

Gradient w.r.t. b (bias):

$$\frac{\partial J}{\partial b} = (h(x) - y)$$

These gradients are used in **gradient descent updates**:

$$w_j = w_j - \eta\frac{\partial J}{\partial w_j}$$

$$b = b - \eta\frac{\partial J}{\partial b}$$

Step 3: Compute the Gradient

The gradient of the loss function with respect to the weights is:

$$\frac{\partial J}{\partial w_j} = (h(x) - y)x_j$$

Similarly, the gradient with respect to bias b is:

$$\frac{\partial J}{\partial b} = (h(x) - y)$$

Substituting $h(x) = 0.3775$ and $y = 1$:

$$h(x) - y = 0.3775 - 1 = -0.6225$$

Compute gradients for w_1, w_2, and b:

$$\nabla w_1 = -0.6225 \times 2 = -1.245$$

$$\nabla w_2 = -0.6225 \times 3 = -1.8675$$

$$\nabla b = -0.6225$$

Step 4: Update the Weights

Using the gradient descent update rule:

$$w_j = w_j - \eta \frac{\partial J}{\partial w_j}$$

$$b = b - \eta \frac{\partial J}{\partial b}$$

Updating each parameter:

$$w_1 = 0.5 - 0.1(-1.245) = 0.5 + 0.1245 = 0.6245$$

$$w_2 = -0.5 - 0.1(-1.8675) = -0.5 + 0.18675 = -0.3133$$

$$b = 0 - 0.1(-0.6225) = 0.06225$$

Exercise 1: Implement the Perceptron Algorithm

Implement the **Perceptron Learning Algorithm** for a binary classification problem.

Steps:

1. Initialize weights and bias to zero.
2. Use the following dataset:

x_1	x_2	Label y
1	1	1
-1	-1	-1
2	-2	-1
-2	2	1

3. Apply the Perceptron update rule with a learning rate $\eta = 1$.
4. Run for **5 iterations** and print the updated weights after each iteration.
5. Plot the decision boundary after training.

Exercise 3: Implementing Gradient Descent for Linear Regression

Given the loss function for linear regression:

$$J(w) = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i)^2$$

Derive the update rule for w using gradient descent and perform one iteration with:

- $x = [1, 2, 3]$
- $y = [2, 2.5, 3.5]$
- Initial weight $w = 0$
- Learning rate $\eta = 0.1$

Solution

1. Compute the gradient:

$$\nabla J(w) = -\frac{2}{m}\sum_{i=1}^{m} x_i(y_i - wx_i)$$

With $m = 3$:

$$\nabla J(0) = -\frac{2}{3}[(1)(2-0) + (2)(2.5-0) + (3)(3.5-0)]$$

$$= -\frac{2}{3}[2 + 5 + 10.5] = -\frac{2}{3} \times 17.5 = -11.67$$

2. Update the weight:

$$w_1 = w_0 - \eta \nabla J(w_0) = 0 - 0.1(-11.67) = 1.167$$

Final updated weight: $w = 1.167$.

Exercise 3: Multi-Class Classification Using Softmax

Consider a dataset with three classes $y \in \{0, 1, 2\}$ and a softmax classifier with weight matrix:

$$W = \begin{bmatrix} 0.5 & -0.2 & 0.3 \\ 0.1 & 0.7 & -0.5 \end{bmatrix}$$

For input $x = [1, 2]$, compute the softmax probabilities and update the weights using gradient descent for $y = 1$ with learning rate $\eta = 0.1$.

Step 1: Compute Logits z

The logits are computed as:

$$z = W^T x$$

$$z = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5(1) + 0.1(2) \\ -0.2(1) + 0.7(2) \\ 0.3(1) + (-0.5)(2) \end{bmatrix}$$

$$= \begin{bmatrix} 0.5 + 0.2 \\ -0.2 + 1.4 \\ 0.3 - 1.0 \end{bmatrix} = \begin{bmatrix} 0.7 \\ 1.2 \\ -0.7 \end{bmatrix}$$

So the logits (pre-softmax scores) are:

$$z = [0.7, 1.2, -0.7]$$

Step 2: Compute Softmax Probabilities

The **softmax function** converts logits into class probabilities:

$$P(y = i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Compute exponentials

$$e^{0.7} \approx 2.014$$

$$e^{1.2} \approx 3.320$$

$$e^{-0.7} \approx 0.497$$

Compute denominator

$$\sum e^z = 2.014 + 3.320 + 0.497 = 5.831$$

Compute softmax probabilities

$$P(0) = \frac{2.014}{5.831} \approx 0.345$$

$$P(1) = \frac{3.320}{5.831} \approx 0.570$$

$$P(2) = \frac{0.497}{5.831} \approx 0.085$$

So, our predicted probabilities are:

$$[0.345, 0.570, 0.085]$$

Step 3: Compute Gradient of Loss

The **cross-entropy loss** for softmax is:

$$J(W) = -\sum_i y_i \log P(y_i)$$

Since the true class is $y = 1$, the loss simplifies to:

$$J(W) = -\log P(1) = -\log(0.570) \approx 0.561$$

Compute Gradients

The gradient of the softmax loss w.r.t. each weight w_{ij} is:

$$\frac{\partial J}{\partial w_{ij}} = (P(y = j) - y_j)x_i$$

Since $y = 1$, the one-hot encoding is:

$$y = [0, 1, 0]$$

For each class:

$$\frac{\partial J}{\partial W_0} = (0.345 - 0)[1, 2] = [0.345, 0.69]$$

$$\frac{\partial J}{\partial W_1} = (0.570 - 1)[1, 2] = [-0.43, -0.86]$$

$$\frac{\partial J}{\partial W_2} = (0.085 - 0)[1, 2] = [0.085, 0.17]$$

Step 4: Update Weights

Using the **gradient descent update rule**:

$$W = W - \eta \nabla W$$

Updating each row:

$$W_0 = [0.5, 0.1] - 0.1[0.345, 0.69] = [0.5 - 0.0345, 0.1 - 0.069] = [0.4655, 0.031]$$

$$W_1 = [-0.2, 0.7] - 0.1[-0.43, -0.86] = [-0.2 + 0.043, 0.7 + 0.086] = [-0.157, 0.786]$$

$$W_2 = [0.3, -0.5] - 0.1[0.085, 0.17] = [0.3 - 0.0085, -0.5 - 0.017] = [0.2915, -0.517]$$

Final Updated Weight Matrix

$$W = \begin{bmatrix} 0.4655 & -0.157 & 0.2915 \\ 0.031 & 0.786 & -0.517 \end{bmatrix}$$

Exercise 1: Implementing Perceptron Learning with Gradient Descent

Implement a single-layer perceptron using gradient descent to update the weights. Use the following dataset:

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

Tasks

1. Initialize weights and bias randomly.
2. Use the sigmoid activation function:
$$f(z) = \frac{1}{1 + e^{-z}}$$
3. Compute the loss using binary cross-entropy:
$$L = -\frac{1}{N} \sum [y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$
4. Perform weight updates using gradient descent:
$$w = w - \eta \frac{\partial L}{\partial w}$$
5. Train the perceptron and evaluate accuracy on the dataset.