Problem 1: Perceptron Training for AND Function

You are training a perceptron to learn the logical AND function. The perceptron has two inputs x_1 and x_2, a bias b, and weights w_1 and w_2. The learning rate η is set to 0.1.

The AND function is defined as:

$$\text{AND}(x_1, x_2) = \begin{cases} 0 & \text{if } x_1 = 0 \text{ or } x_2 = 0 \\ 1 & \text{if } x_1 = 1 \text{ and } x_2 = 1 \end{cases}$$

Given:

- Initial weights: $w_1 = 0.1$, $w_2 = 0.1$
- Initial bias: $b = -0.1$
- Training data: (x_1, x_2) with target outputs t

x_1	x_2	t
0	0	0
0	1	0
1	0	0
1	1	1

Perform one epoch of training, updating the weights and bias after each example using the perceptron learning rule.

Solution 1:

Initialize:

- $w_1 = 0.1, w_2 = 0.1$
- $b = -0.1$
- $\eta = 0.1$

Training:

For input $(x_1 = 0, x_2 = 0)$:

- $z = w_1 \times x_1 + w_2 \times x_2 + b = 0.1 \times 0 + 0.1 \times 0 - 0.1 = -0.1$
- $y = f(z) = 0$
- Target $t = 0$. Error $(t - y) = 0 - 0 = 0$.
- No update needed.

For input $(x_1 = 0, x_2 = 1)$:

- $z = w_1 \times x_1 + w_2 \times x_2 + b = 0.1 \times 0 + 0.1 \times 1 - 0.1 = 0$
- $y = f(z) = 1$
- Target $t = 0$. Error $(t - y) = 0 - 1 = -1$.
- Update weights and bias:

$$\Delta w_1 = 0, \quad \Delta w_2 = -0.1 \times 1 \times 1 = -0.1, \quad \Delta b = -0.1$$

New weights and bias:

$$w_1 = 0.1,\ w_2 = 0.1 - 0.1 = 0,\ b = -0.1 - 0.1 = -0.2$$

For input $(x_1 = 1, x_2 = 0)$:

- $z = w_1 \times x_1 + w_2 \times x_2 + b = 0.1 \times 1 + 0 \times 0 - 0.2 = -0.1$
- $y = f(z) = 0$
- Target $t = 0$. Error $(t - y) = 0 - 0 = 0$.
- No update needed.

For input $(x_1 = 1, x_2 = 1)$:

- $z = w_1 \times x_1 + w_2 \times x_2 + b = 0.1 \times 1 + 0 \times 1 - 0.2 = -0.1$
- $y = f(z) = 0$
- Target $t = 1$. Error $(t - y) = 1 - 0 = 1$.
- Update weights and bias:

  $$\Delta w_1 = 0.1 \times 1 \times 1 = 0.1, \quad \Delta w_2 = 0.1 \times 1 \times 1 = 0.1, \quad \Delta b = 0.1$$

  New weights and bias:

  $$w_1 = 0.1 + 0.1 = 0.2,\ w_2 = 0 + 0.1 = 0.1,\ b = -0.2 + 0.1 = -0.1$$

Final Answer:

After one epoch of training:

- Weights: $w_1 = 0.2$, $w_2 = 0.1$
- Bias: $b = -0.1$

Problem 2: Perceptron Learning for Linearly Separable Data

Consider a perceptron with two inputs, trained on the following linearly separable dataset:

x_1	x_2	t
1	2	1
2	1	1
-1	-2	0
-2	-1	0

Given:

- Initial weights: $w_1 = 0.5$, $w_2 = -0.5$
- Initial bias: $b = 0.0$
- Learning rate: $\eta = 0.2$

Perform one epoch of training using the perceptron learning rule.

Solution 2:

Initialize:

- $w_1 = 0.5, w_2 = -0.5$
- $b = 0.0$
- $\eta = 0.2$

Training:

For input $(x_1 = 1, x_2 = 2)$:

- $z = w_1 \times x_1 + w_2 \times x_2 + b = 0.5 \times 1 - 0.5 \times 2 + 0 = -0.5$
- $y = f(z) = 0$
- Target $t = 1$. Error $(t - y) = 1 - 0 = 1$.
- Update weights and bias:

  $$\Delta w_1 = 0.2 \times 1 \times 1 = 0.2, \quad \Delta w_2 = 0.2 \times 1 \times 2 = 0.4, \quad \Delta b = 0.2$$

  New weights and bias:

  $$w_1 = 0.5 + 0.2 = 0.7,\ w_2 = -0.5 + 0.4 = -0.1,\ b = 0 + 0.2 = 0.2$$

For input $(x_1 = 2, x_2 = 1)$:

- $z = w_1 \times x_1 + w_2 \times x_2 + b = 0.7 \times 2 - 0.1 \times 1 + 0.2 = 1.5$
- $y = f(z) = 1$
- Target $t = 1$. Error $(t - y) = 1 - 1 = 0$.
- No update needed.

For input $(x_1 = -1, x_2 = -2)$:

- $z = w_1 \times x_1 + w_2 \times x_2 + b = 0.7 \times (-1) - 0.1 \times (-2) + 0.2 = -0.4$
- $y = f(z) = 0$
- Target $t = 0$. Error $(t - y) = 0 - 0 = 0$.
- No update needed.

For input $(x_1 = -2, x_2 = -1)$:

- $z = w_1 \times x_1 + w_2 \times x_2 + b = 0.7 \times (-2) - 0.1 \times (-1) + 0.2 = -1.1$
- $y = f(z) = 0$
- Target $t = 0$. Error $(t - y) = 0 - 0 = 0$.
- No update needed.

Final Answer:

After one

Problem 1: Forward Propagation in a Simple Neural Network

Consider a neural network with:

- 1 input layer with 2 neurons (x_1, x_2)
- 1 hidden layer with 2 neurons (h_1, h_2)
- 1 output layer with 1 neuron (y)

Activation functions:

- Hidden layer: Sigmoid function
- Output layer: Sigmoid function

Weights and biases:

- Weights from input to hidden layer:
  - $W_{11} = 0.5$, $W_{12} = -0.5$ (for h_1)
  - $W_{21} = 0.3$, $W_{22} = 0.2$ (for h_2)
- Weights from hidden to output layer:
  - $W_{h1} = 0.7$, $W_{h2} = -0.2$
- Biases:
  - Hidden layer: $b_{h1} = 0.1$, $b_{h2} = -0.1$
  - Output layer: $b_y = 0.2$

Input:

- $x_1 = 1$
- $x_2 = 0.5$

Calculate the output y of the neural network.

1. **Compute the activations of the hidden layer:**

   For h_1:

   $$z_{h1} = W_{11} \times x_1 + W_{12} \times x_2 + b_{h1} = 0.5 \times 1 + (-0.5) \times 0.5 + 0.1 = 0.5 - 0.25 + 0.$$

   $$a_{h1} = \sigma(z_{h1}) = \frac{1}{1 + e^{-0.35}} \approx 0.5866$$

   For h_2:

   $$z_{h2} = W_{21} \times x_1 + W_{22} \times x_2 + b_{h2} = 0.3 \times 1 + 0.2 \times 0.5 - 0.1 = 0.3 + 0.1 - 0.1 = 0$$

   $$a_{h2} = \sigma(z_{h2}) = \frac{1}{1 + e^{-0.3}} \approx 0.5744$$

2. **Compute the output of the network:**

   $$z_y = W_{h1} \times a_{h1} + W_{h2} \times a_{h2} + b_y = 0.7 \times 0.5866 + (-0.2) \times 0.5744 + 0.2$$

   $$z_y = 0.4116 - 0.1149 + 0.2 = 0.4967$$

   $$a_y = \sigma(z_y) = \frac{1}{1 + e^{-0.4967}} \approx 0.6217$$

Final Answer:

- Output $y \approx 0.6217$

Solution 2:

1. **Compute the error term for the output layer:**

$$\text{Error} = y - t = 0.6217 - 1 = -0.3783$$

The gradient of the loss function with respect to the output activation a_y is:

$$\frac{\partial L}{\partial a_y} = (a_y - t) \times \sigma'(z_y)$$

where $\sigma'(z_y) = a_y \times (1 - a_y)$.

$$\sigma'(z_y) = 0.6217 \times (1 - 0.6217) \approx 0.2350$$

$$\frac{\partial L}{\partial a_y} = -0.3783 \times 0.2350 \approx -0.089$$

2. **Compute the gradients for the weights from hidden to output layer:**

$$\frac{\partial L}{\partial W_{h1}} = \frac{\partial L}{\partial a_y} \times a_{h1} \approx -0.089 \times 0.5866 \approx -0.052$$

$$\frac{\partial L}{\partial W_{h2}} = \frac{\partial L}{\partial a_y} \times a_{h2} \approx -0.089 \times 0.5744 \approx -0.051$$

$$\frac{\partial L}{\partial b_y} = \frac{\partial L}{\partial a_y} \approx -0.089$$

3. **Compute the gradients for the weights and biases in the hidden layer:**

$$\frac{\partial L}{\partial z_{h1}} = \frac{\partial L}{\partial a_y} \times W_{h1} \times \sigma'(\approx -0.089 \times 0.7 \times 0.2401 \approx -0.015$$

3. **Compute the gradients for the weights and biases in the hidden layer:**

$$\frac{\partial L}{\partial z_{h1}} = \frac{\partial L}{\partial a_y} \times W_{h1} \times \sigma'(z_{h1}) \approx -0.089 \times 0.7 \times 0.2401 \approx -0.015$$

$$\frac{\partial L}{\partial W_{11}} = \frac{\partial L}{\partial z_{h1}} \times x_1 \approx -0.015 \times 1 = -0.015$$

$$\frac{\partial L}{\partial W_{12}} = \frac{\partial L}{\partial z_{h1}} \times x_2 \approx -0.015 \times 0.5 = -0.007$$

$$\frac{\partial L}{\partial b_{h1}} = \frac{\partial L}{\partial z_{h1}} \approx -0.015$$

$$\frac{\partial L}{\partial z_{h2}} = \frac{\partial L}{\partial a_y} \times W_{h2} \times \sigma'(z_{h2}) \approx -0.089 \times (-0.2) \times 0.2350 \approx 0.0042$$

$$\frac{\partial L}{\partial W_{21}} = \frac{\partial L}{\partial z_{h2}} \times x_1 \approx 0.0042 \times 1 = 0.0042$$

$$\frac{\partial L}{\partial W_{22}} = \frac{\partial L}{\partial z_{h2}} \times x_2 \approx 0.0042 \times 0.5 = 0.0021$$

$$\frac{\partial L}{\partial b_{h2}} = \frac{\partial L}{\partial z_{h2}} \approx 0.0042$$

Final Answer:

- Gradient with respect to $W_{h1} \approx -0.052$
- Gradient with respect to $W_{h2} \approx -0.051$
- Gradient with respect to $b_y \approx -0.089$
- Gradient with respect to $W_{11} \approx -0.015$
- Gradient with respect to $W_{12} \approx -0.007$
- Gradient with respect to $b_{h1} \approx -0.015$
- Gradient with respect to $W_{21} \approx 0.0042$
- Gradient with respect to $W_{22} \approx 0.0021$
- Gradient with respect to $b_{h2} \approx 0.0042$

Problem 1: Convolution Operation

Given an input image and a convolutional kernel, perform a convolution operation.

Input image (4x4):

$$\begin{matrix} 1 & 2 & 3 & 0 \\ 4 & 5 & 6 & 1 \\ 7 & 8 & 9 & 2 \\ 0 & 1 & 2 & 3 \end{matrix}$$

Kernel (2x2):

$$\begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$$

Stride: 1

Padding: 0 (valid padding)

Calculate the output of the convolution operation.

Solution 1:

1. **Calculate the size of the output feature map:**

   Output size (width and height) is:

$$\text{Output size} = \left(\frac{\text{Input size} - \text{Kernel size}}{\text{Stride}} \right) + 1$$

$$\text{Output size} = \left(\frac{4 - 2}{1} \right) + 1 = 3$$

2. **Perform the convolution:**

- **Top-left corner:**

$$\text{Region} = \begin{matrix} 1 & 2 \\ 4 & 5 \end{matrix}$$

$$\text{Convolution} = (1 \times 1) + (2 \times 0) + (4 \times 0) + (5 \times -1) = 1 - 5 = -4$$

- **Top-middle:**

$$\text{Region} = \begin{matrix} 2 & 3 \\ 5 & 6 \end{matrix}$$

$$\text{Convolution} = (2 \times 1) + (3 \times 0) + (5 \times 0) + (6 \times -1) = 2 - 6 = -4$$

- **Top-right:**

$$\text{Region} = \begin{matrix} 3 & 0 \\ 6 & 1 \end{matrix}$$

$$\text{Convolution} = (3 \times 1) + (0 \times 0) + (6 \times 0) + (1 \times -1) = 3 - 1 = 2$$

- **Middle-left:**

$$\text{Region} = \begin{matrix} 4 & 5 \\ 7 & 8 \end{matrix}$$

$$\text{Convolution} = (4 \times 1) + (5 \times 0) + (7 \times 0) + (8 \times -1) = 4 - 8 = -4$$

- **Middle-middle:**

$$\text{Region} = \begin{matrix} 5 & 6 \\ 8 & 9 \end{matrix}$$

$$\text{Convolution} = (5 \times 1) + (6 \times 0) + (8 \times 0) + (9 \times -1) = 5 - 9 = -4$$

- **Middle-right:**

$$\text{Region} = \begin{matrix} 6 & 1 \\ 9 & 2 \end{matrix}$$

$$\text{Convolution} = (6 \times 1) + (1 \times 0) + (9 \times 0) + (2 \times -1) = 6 - 2 = 4$$

- **Bottom-left:**

$$\text{Region} = \begin{matrix} 7 & 8 \\ 0 & 1 \end{matrix}$$

$$\text{Convolution} = (7 \times 1) + (8 \times 0) + (0 \times 0) + (1 \times -1) = 7 - 1 = 6$$

- **Bottom-middle:**

$$\text{Region} = \begin{matrix} 8 & 9 \\ 1 & 2 \end{matrix}$$

$$\text{Convolution} = (8 \times 1) + (9 \times 0) + (1 \times 0) + (2 \times -1) = 8 - 2 = 6$$

- **Bottom-right:**

$$\text{Region} = \begin{matrix} 9 & 2 \\ 2 & 3 \end{matrix}$$

$$\text{Convolution} = (9 \times 1) + (2 \times 0) + (2 \times 0) + (3 \times -1) = 9 - 3 = 6$$

Output Feature Map:

$$\begin{matrix} -4 & -4 & 2 \\ -4 & -4 & 4 \\ 6 & 6 & 6 \end{matrix}$$

Problem 2: Max Pooling

Given the output from the previous convolution operation, apply a 2x2 max pooling operation with a stride of 2.

Input Feature Map:

$$\begin{matrix} -4 & -4 & 2 \\ -4 & -4 & 4 \\ 6 & 6 & 6 \end{matrix}$$

Stride: 2

Calculate the output after applying max pooling.

Solution 2:

1. **Apply 2x2 Max Pooling:**
   - **Top-left:**

$$\text{Region} = \begin{matrix} -4 & -4 \\ -4 & -4 \end{matrix}$$

$$\text{Max} = -4$$

   - **Top-right:**

$$\text{Region} = \begin{matrix} -4 & 2 \\ -4 & 4 \end{matrix}$$

$$\text{Max} = 4$$

- **Bottom-left:**

$$\text{Region} = 6 \quad 6$$

$$\text{Max} = 6$$

- **Bottom-right:**

$$\text{Region} = 6 \quad 6$$

$$\text{Max} = 6$$

Output Feature Map:

$$\begin{matrix} -4 & 4 \\ 6 & 6 \end{matrix}$$

Problem 3: CNN Forward Pass with Multiple Channels

Given an input with multiple channels, perform a forward pass through a convolutional layer.

Input:

- Input volume: 3 channels (2x2 each)

Channel 1:

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$$

Channel 2:

$$\begin{matrix} 0 & 1 \\ 2 & 3 \end{matrix}$$

Channel 3:

$$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$$

Kernel:

- 2x2 with 3 channels

Kernel:

$$\begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$$

(Used for all 3 channels)

Bias: 1

Stride: 1

Padding: 0 (valid padding)

Calculate the output after applying the convolution operation.

Solution 3:

1. **Perform the convolution for each channel and add them up:**

   For each channel:

   - **Channel 1:**

     $$\text{Convolution} = (1 \times 1) + (2 \times 0) + (3 \times 0) + (4 \times -1) = 1 - 4 = -3$$

   - **Channel 2:**

     $$\text{Convolution} = (0 \times 1) + (1 \times 0) + (2 \times 0) + (3 \times -1) = 0 - 3 = -3$$

   - **Channel 3:**

     $$\text{Convolution} = (1 \times 1) + (0 \times 0) + (0 \times 0) + (1 \times -1) = 1 - 1 = 0$$

2. **Sum the results and add bias:**

$$\text{Sum} = -3 + (-3) + 0 = -6$$

$$\text{Output} = -6 + 1 = -5$$

Output Feature Map:

$$-5$$

Problem 1: Autoencoder Forward Pass

Consider a simple autoencoder with:

- **Input layer**: 3 neurons
- **Hidden layer (encoder)**: 2 neurons
- **Output layer (decoder)**: 3 neurons

Weights and biases for the encoder:

Weights (W_1) from input to hidden layer:

$$\begin{matrix} 0.2 & 0.8 \\ 0.5 & 0.1 \\ -0.3 & 0.6 \end{matrix}$$

Biases (b_1) for the hidden layer:

$$\begin{matrix} 0.1 \\ -0.2 \end{matrix}$$

Weights (W_2) from hidden layer to output layer:

$$\begin{matrix} 0.4 & -0.6 & 0.5 \\ 0.7 & -0.2 & -0.1 \end{matrix}$$

Biases (b_2) for the output layer:

$$\begin{matrix} 0.2 \\ -0.1 \\ 0.3 \end{matrix}$$

Input vector:

$$\begin{matrix} 1 \\ 0.5 \\ -1 \end{matrix}$$

Calculate the output of the autoencoder.

Solution 1:

1. **Forward Pass through Encoder:**

   Calculate hidden layer activations:

   - **Hidden neuron 1:**

     $$z_{h1} = (0.2 \times 1) + (0.5 \times 0.5) + (-0.3 \times -1) + 0.1 = 0.2 + 0.25 + 0.3 + 0.1 = 0$$

     $$a_{h1} = \sigma(z_{h1}) = \frac{1}{1 + e^{-0.85}} \approx 0.7004$$

   - **Hidden neuron 2:**

     $$z_{h2} = (0.8 \times 1) + (0.1 \times 0.5) + (0.6 \times -1) - 0.2 = 0.8 + 0.05 - 0.6 - 0.2 = 0.0$$

     $$a_{h2} = \sigma(z_{h2}) = \frac{1}{1 + e^{-0.05}} \approx 0.5125$$

2. **Forward Pass through Decoder:**

   Calculate output layer activations:

   - **Output neuron 1:**

     $$z_{o1} = (0.4 \times 0.7004) + (-0.6 \times 0.5125) + 0.2 = 0.2802 - 0.3085 + 0.2 = 0.171$$

     $$a_{o1} = \sigma(z_{o1}) = \frac{1}{1 + e^{-0.1717}} \approx 0.5428$$

   - **Output neuron 2:**

     $$z_{o2} = (0.7 \times 0.7004) + (-0.2 \times 0.5125) - 0.1 = 0.4893 - 0.1025 - 0.1 = 0.286$$

$$a_{o2} = \sigma(z_{o2}) = \frac{1}{1 + e^{-0.2868}} \approx 0.5717$$

- **Output neuron 3:**

$$z_{o3} = (0.5 \times 0.7004) + (-0.1 \times 0.5125) + 0.3 = 0.3502 - 0.0526 + 0.3 = 0.597$$

$$a_{o3} = \sigma(z_{o3}) = \frac{1}{1 + e^{-0.5976}} \approx 0.6451$$

Output of the Autoencoder:

$$\begin{matrix} 0.5428 \\ 0.5717 \\ 0.6451 \end{matrix}$$

Problem 2: Autoencoder Loss Calculation

Using the same autoencoder setup from Problem 1, calculate the Mean Squared Error (MSE) loss given the target output vector.

Target output vector:

$$\begin{matrix} 0.6 \\ 0.5 \\ 0.7 \end{matrix}$$

Calculated output from the previous problem:

$$\begin{matrix} 0.5428 \\ 0.5717 \\ 0.6451 \end{matrix}$$

Calculate the Mean Squared Error (MSE) loss.

Solution 2:

1. **Calculate the Mean Squared Error (MSE) loss:**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where n is the number of output neurons, y_i is the target output, and $\hat{y}_i$ is the calculated output.

$$\text{MSE} = \frac{1}{3}\left[(0.6 - 0.5428)^2 + (0.5 - 0.5717)^2 + (0.7 - 0.6451)^2\right]$$

$$\text{MSE} = \frac{1}{3}\left[0.0032 + 0.0051 + 0.0030\right]$$

$$\text{MSE} = \frac{1}{3} \times 0.0113 \approx 0.0038$$

Mean Squared Error (MSE) Loss:

$$0.0038$$

Problem 3: Autoencoder Gradient Calculation

Given the following setup:

Hidden layer activations (from Problem 1):

$$a_{h1} = 0.7004$$
$$a_{h2} = 0.5125$$

Output layer activations (from Problem 1):

$$a_{o1} = 0.5428$$
$$a_{o2} = 0.5717$$
$$a_{o3} = 0.6451$$

Target output vector:

$$\begin{matrix} 0.6 \\ 0.5 \\ 0.7 \end{matrix}$$

Calculate the gradient of the loss with respect to the output layer weights W_2 and biases b_2. Use a learning rate $\eta = 0.01$.

Solution 3:

1. **Calculate the error term for the output layer:**

   For each output neuron:

   - **Output neuron 1:**

$$\text{Error}_1 = a_{o1} - t_{o1} = 0.5428 - 0.6 = -0.0572$$

$$\text{Gradient w.r.t. } z_{o1} = \text{Error}_1 \times \sigma'(z_{o1})$$

$$\sigma'(z_{o1}) = a_{o1} \times (1 - a_{o1}) \approx 0.5428 \times (1 - 0.5428) \approx 0.2481$$

$$\text{Gradient w.r.t. } z_{o1} = -0.0572 \times 0.2481 \approx -0.0142$$

- **Output neuron 2:**

$$\text{Error}_2 = a_{o2} - t_{o2} = 0.5717 - 0.5 = 0.0717$$

$$\text{Gradient w.r.t. } z_{o2} = \text{Error}_2 \times \sigma'(z_{o2})$$

$$\sigma'(z_{o2}) = a_{o2} \times (1 - a_{o2}) \approx 0.5717 \times (1 - 0.5717) \approx 0.2452$$

$$\text{Gradient w.r.t. } z_{o2} = 0.0717 \times 0.2452 \approx 0.0176$$

- **Output neuron 3:**

$$\text{Error}_3 = a_{o3} - t_{o3} = 0.6451 - 0.7 = -0.0549$$

$$\text{Gradient w.r.t. } z_{o3} = \text{Error}_3 \times \sigma'(z_{o3})$$

$$\sigma'(z_{o3}) = a_{o3} \times (1 - a_{o3}) \approx 0.6451 \times (1 - 0.6451) \approx 0.2271$$

$$\text{Gradient w.r.t. } z_{o3} = -0.0549 \times 0.2271 \approx -0.0125$$

2. **Calculate gradient of the loss with respect to the weights W_2 and biases b_2:**

   The gradients with respect to weights are:

$$\frac{\partial L}{\partial W_{2ij}} = a_{hi} \times \text{Gradient w.r.t. } z_{o_j}$$

   where a_{hi} is the activation of the i-th hidden neuron.

   For W_{21} (contributes to output neuron 1):

$$\frac{\partial L}{\partial W_{21}} = a_{h1} \times -0.0142 = 0.7004 \times -0.0142 \approx -0.0099$$

   For W_{22} (contributes to output neuron 2):

$$\frac{\partial L}{\partial W_{22}} = a_{h1} \times 0.0176 = 0.7004 \times 0.0176 \approx 0.0123$$

   For W_{23} (contributes to output neuron 3):

$$\frac{\partial L}{\partial W_{23}} = a_{h1} \times -0.0125 = 0.7004 \times -0.0125 \approx -0.0088$$

   Bias gradients are:

$$\frac{\partial L}{\partial b_{2j}} = \text{Gradient w.r.t. } z_{o_j}$$

   For b_{21}:

$$\frac{\partial L}{\partial b_{21}} = -0.0142$$

   For b_{22}:

$$\frac{\partial L}{\partial b_{22}} = 0.0176$$

For b_{23}:

$$\frac{\partial L}{\partial b_{23}} = -0.0125$$

Updated Weights and Biases:

- Adjust the weights and biases using the learning rate $\eta = 0.01$.

Example:
For W_{21}:

$$W_{21} = W_{21} - \eta \times \frac{\partial L}{\partial W_{21}} = 0.4 - 0.01 \times (-0.0099) = 0.4 + 0.000099 = 0.4001$$

Problem 1: Basic RNN Forward Pass

Consider a simple RNN with:

- **Input vector** x: 2-dimensional
- **Hidden state vector** h: 2-dimensional
- **Output vector** y: 1-dimensional

Weights and biases:

Input-to-Hidden weights (W_x):

$$\begin{matrix} 0.5 & -0.2 \\ 0.3 & 0.8 \end{matrix}$$

Hidden-to-Hidden weights (W_h):

$$\begin{matrix} 0.6 & -0.4 \\ 0.1 & 0.9 \end{matrix}$$

Hidden-to-Output weights (W_y):

$$\begin{matrix} 0.7 \\ -0.3 \end{matrix}$$

Biases:

Hidden layer bias (b_h):

$$\begin{matrix} 0.1 \\ -0.2 \end{matrix}$$

Output bias (b_y):

$$0.2$$

Input vector x:

$$\begin{matrix} 1 \\ -1 \end{matrix}$$

Initial hidden state h_0:

$$\begin{matrix} 0 \\ 0 \end{matrix}$$

Calculate the hidden state h_1 and the output y_1 for this RNN.

Solution 1:

1. **Calculate the hidden state h_1:**

   The RNN hidden state is given by:

   $$h_1 = \tanh(W_x \cdot x + W_h \cdot h_0 + b_h)$$

   - **Input-to-Hidden Contribution:**

   $$W_x \cdot x = \begin{matrix} 0.5 & -0.2 \\ 0.3 & 0.8 \end{matrix} \cdot \begin{matrix} 1 \\ -1 \end{matrix} = \begin{matrix} (0.5 \cdot 1) + (-0.2 \cdot -1) \\ (0.3 \cdot 1) + (0.8 \cdot -1) \end{matrix} = \begin{matrix} 0.5 + 0.2 \\ 0.3 - 0.8 \end{matrix} = \begin{matrix} 0.7 \\ -0.5 \end{matrix}$$

   - **Hidden-to-Hidden Contribution:**

   $$W_h \cdot h_0 = \begin{matrix} 0.6 & -0.4 \\ 0.1 & 0.9 \end{matrix} \cdot \begin{matrix} 0 \\ 0 \end{matrix} = \begin{matrix} 0 \\ 0 \end{matrix}$$

   - **Add biases:**

   $$h_1 = \tanh\begin{pmatrix} 0.7 + 0.1 \\ -0.5 - 0.2 \end{pmatrix} = \tanh\begin{pmatrix} 0.8 \\ -0.7 \end{pmatrix}$$

$$h_1 = \begin{matrix} \tanh(0.8) \\ \tanh(-0.7) \end{matrix} \approx \begin{matrix} 0.664 \\ -0.604 \end{matrix}$$

2. **Calculate the output y_1:**

   The output is given by:

$$y_1 = W_y \cdot h_1 + b_y$$

$$W_y \cdot h_1 = \begin{matrix} 0.7 \\ -0.3 \end{matrix} \cdot \begin{matrix} 0.664 \\ -0.604 \end{matrix} = (0.7 \cdot 0.664) + (-0.3 \cdot -0.604) = 0.4668 + 0.1812 = 0.6$$

$$y_1 = 0.648 + 0.2 = 0.848$$

Hidden State h_1:

$$\begin{matrix} 0.664 \\ -0.604 \end{matrix}$$

Output y_1:

$$0.848$$

Problem 2: LSTM Cell Forward Pass

Consider an LSTM cell with the following parameters:

- **Input vector** x: 3-dimensional
- **Hidden state vector** h: 2-dimensional
- **Cell state vector** c: 2-dimensional

Weights and biases:

Weights for input-to-hidden (for all gates) (W):

$$\begin{matrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.8 \\ 0.9 & 1.0 & 1.1 & 1.2 \end{matrix}$$

Weights for hidden-to-hidden (for all gates) (U):

$$\begin{matrix} 0.2 & 0.1 \\ 0.4 & 0.3 \\ 0.6 & 0.5 \end{matrix}$$

Biases for gates (b):

$$\begin{matrix} 0.1 & 0.1 & 0.1 & 0.1 \end{matrix}$$

Input vector x:

$$\begin{matrix} 0.5 \\ -0.2 \\ 0.1 \end{matrix}$$

Previous hidden state h_{t-1}:

$$\begin{matrix} 0.1 \\ -0.1 \end{matrix}$$

Previous cell state c_{t-1}:

$$\begin{matrix} 0.2 \\ -0.2 \end{matrix}$$

Calculate the new hidden state h_t and cell state c_t.

Solution 2:

1. **Calculate the gates:**

   Concatenate input and previous hidden state:

$$[x, h_{t-1}] = \begin{matrix} 0.5 \\ -0.2 \\ 0.1 \\ 0.1 \\ -0.1 \end{matrix}$$

   Calculate the gate activations:

$$\text{Concatenated} = \begin{matrix} 0.5 & -0.2 & 0.1 & 0.1 \\ -0.2 & 0.1 & -0.1 & -0.1 \end{matrix}$$

$$Z = W \cdot \text{Concatenated} + b$$

   - **Input Gate i:**

- **Input Gate i:**

$$i = \sigma \begin{pmatrix} (0.1 \times 0.5) + (0.2 \times -0.2) + (0.3 \times 0.1) + (0.4 \times 0.1) + 0.1 \\ (0.5 \times 0.5) + (0.6 \times -0.2) + (0.7 \times 0.1) + (0.8 \times 0.1) + 0.1 \end{pmatrix}$$

$$i = \sigma \begin{pmatrix} 0.05 - 0.04 + 0.03 + 0.04 + 0.1 \\ 0.25 - 0.12 + 0.07 + 0.08 + 0.1 \end{pmatrix}$$

$$i = \sigma \begin{pmatrix} 0.14 \\ 0.38 \end{pmatrix}$$

$$i = \begin{matrix} 0.535 \\ 0.593 \end{matrix}$$

- **Forget Gate f:**

$$f = \sigma \begin{pmatrix} (0.1 \times 0.5) + (0.2 \times -0.2) + (0.3 \times 0.1) + (0.4 \times 0.1) + 0.1 \\ (0.5 \times 0.5) + (0.6 \times -0.2) + (0.7 \times 0.1) + (0.8 \times 0.1) + 0.1 \end{pmatrix}$$

$$f = \sigma \begin{pmatrix} 0.14 \\ 0.38 \end{pmatrix}$$

$$f = \begin{matrix} 0.535 \\ 0.593 \end{matrix}$$

- **Output Gate o:**

$$o = \sigma \begin{pmatrix} (0.1 \times 0.5) + (0.2 \times -0.2) + (0.3 \times 0.1) + (0.4 \times 0.1) + 0.1 \\ (0.5 \times 0.5) + (0.6 \times -0.2) + (0.7 \times 0.1) + (0.8 \times 0.1) + 0.1 \end{pmatrix}$$

$$\downarrow = \sigma \begin{pmatrix} 0.14 \\ 0.38 \end{pmatrix}$$

$$o = \begin{matrix} 0.535 \\ 0.593 \end{matrix}$$

- **Cell State $\tilde{c}$:**

$$\tilde{c} = \tanh\begin{pmatrix} (0.1 \times 0.5) + (0.2 \times -0.2) + (0.3 \times 0.1) + (0.4 \times 0.1) + 0.1 \\ (0.5 \times 0.5) + (0.6 \times -0.2) + (0.7 \times 0.1) + (0.8 \times 0.1) + 0.1 \end{pmatrix}$$

$$\tilde{c} = \tanh\begin{pmatrix} 0.14 \\ 0.38 \end{pmatrix}$$

$$\tilde{c} = \begin{matrix} 0.139 \\ 0.364 \end{matrix}$$

2. **Calculate the new cell state c_t:**

$$c_t = f \cdot c_{t-1} + i \cdot \tilde{c}$$

$$c_t = \begin{matrix} 0.535 \cdot 0.2 + 0.593 \cdot 0.364 \\ 0.535 \cdot -0.2 + 0.593 \cdot 0.364 \end{matrix}$$

$$c_t = \begin{matrix} 0.107 + 0.216 \\ -0.107 + 0.216 \end{matrix}$$

$$c_t = \begin{matrix} 0.323 \\ 0.109 \end{matrix}$$

3. **Calculate the new hidden state h_t:**

$$h_t = o \cdot \tanh(c_t)$$

$$h_t = \begin{matrix} 0.535 \cdot \tanh(0.323) \\ 0.593 \cdot \tanh(0.109) \end{matrix}$$

$$h_t = \begin{matrix} 0.535 \cdot 0.316 \\ 0.593 \cdot 0.109 \end{matrix}$$

$$h_t = \begin{matrix} 0.169 \\ 0.065 \end{matrix}$$

New Hidden State h_t:

$$\begin{matrix} 0.169 \\ 0.065 \end{matrix}$$

New Cell State c_t:

$$\begin{matrix} 0.323 \\ 0.109 \end{matrix}$$
