A single neuron receives two inputs.

The setup is as follows:

- Input vector: $x = [3, -1]$
- Weights: $w = [0.4, -0.6]$
- Bias: $b = 0.2$
- Activation function: **Identity** (i.e., no non-linearity)

You are training this neuron with:

- True target label: $y = 2$
- Loss function: **Mean Squared Error (MSE)**:

$$L = (y - \hat{y})^2$$

- Learning rate: $\alpha = 0.05$

1. **Forward Pass:**
   Calculate the predicted output $\hat{y}$ of the neuron.
2. **Loss Calculation:**
   Calculate the loss using MSE between the predicted output $\hat{y}$ and the true label y.
3. **Backpropagation:**
   - Compute the gradients of the loss with respect to each weight.
   - (Hint: use chain rule: $\frac{dL}{dw} = \frac{dL}{d\hat{y}} \times \frac{d\hat{y}}{dw}$)
4. **Weight Updates:**
   Update the weights using **gradient descent** and the learning rate.

- "$\frac{dL}{d\hat{y}} = 2(\hat{y} - y)$"
- "$\frac{d\hat{y}}{dw_i} = x_i$ (**because** $\hat{y} = w_1x_1 + w_2x_2 + b$)"

1. Forward Pass

Compute predicted output:

$$\hat{y} = (w_1 \times x_1) + (w_2 \times x_2) + b$$

Substituting:

$$\hat{y} = (0.4 \times 3) + (-0.6 \times -1) + 0.2$$

$$\hat{y} = 1.2 + 0.6 + 0.2$$

$$\hat{y} = 2.0$$

✅ **Predicted output:** $\hat{y} = 2.0$

2. Loss Calculation

Using MSE:

$$L = (y - \hat{y})^2$$

Substituting:

$$L = (2 - 2)^2$$

$$L = (0)^2 = 0$$

✅ **Loss:** $L = 0$

3. Backpropagation (Gradient Calculation)

First, find:

$$\frac{dL}{d\hat{y}} = 2(\hat{y} - y)$$

Substituting:

$$\frac{dL}{d\hat{y}} = 2(2 - 2) = 2(0) = 0$$

Now, gradients with respect to weights:

- For w_1:

$$\frac{dL}{dw_1} = \frac{dL}{d\hat{y}} \times \frac{d\hat{y}}{dw_1} = 0 \times x_1 = 0 \times 3 = 0$$

- For w_2:

$$\frac{dL}{dw_2} = \frac{dL}{d\hat{y}} \times \frac{d\hat{y}}{dw_2} = 0 \times x_2 = 0 \times (-1) = 0$$

✅ **Gradients**:

$$\nabla w = [0, 0]$$

4. Weight Updates

Update rule:

$$w_{\text{new}} = w_{\text{old}} - \alpha \times \nabla w$$

Since gradients are 0, the weights do not change:

- w_1 update:

$$w_1 = 0.4 - 0.05 \times 0 = 0.4$$

- w_2 update:

$$w_2 = -0.6 - 0.05 \times 0 = -0.6$$

✅ **New weights**:

$$w = [0.4, -0.6]$$

Example:

Let's say you have the following:

- Input: $x = [x_1, x_2] = [1, 2]$
- Hidden layer: ReLU activation
- Output layer: Sigmoid activation
- Loss: MSE

1. Compute the **forward pass**:
   - Hidden layer: $h = f(w \times x + b)$
   - Output layer: $y_{\text{pred}} = f(v \times h + b_{\text{output}})$
2. Compute the **backpropagation** gradients:
   - Compute $\frac{\partial L}{\partial v_k}$
   - Compute $\frac{\partial L}{\partial w_k}$
3. Update the weights with gradient descent:
   - $v_k = v_k - \alpha \cdot \frac{\partial L}{\partial v_k}$
   - $w_k = w_k - \alpha \cdot \frac{\partial L}{\partial w_k}$

Neural Network Setup

- **Input Layer**: $x = [1, 2]$
- **Hidden Layer**:
  - Weights: $w = \begin{bmatrix} 0.5 & 0.6 \\ 0.2 & 0.3 \end{bmatrix}$ (2 weights for each neuron, for 2 inputs)
  - Bias: $b^{(1)} = [0.1, 0.1]$
  - Activation function: ReLU
- **Output Layer**:
  - Weights: $v = \begin{bmatrix} 0.4 & 0.7 \end{bmatrix}$
  - Bias: $b^{(2)} = 0.2$
  - Activation function: Sigmoid
- **True output (target label)**: $y = 1.0$
- **Learning rate**: $\alpha = 0.01$

1. Forward Pass

1.1 Hidden Layer

First, we compute the pre-activation of the hidden layer:

$$z_h = w \cdot x + b^{(1)} = \begin{bmatrix} 0.5 & 0.6 \\ 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

Calculating the dot product:

- $z_{h1} = (0.5 \cdot 1) + (0.6 \cdot 2) + 0.1 = 0.5 + 1.2 + 0.1 = 1.8$
- $z_{h2} = (0.2 \cdot 1) + (0.3 \cdot 2) + 0.1 = 0.2 + 0.6 + 0.1 = 0.9$

So, the pre-activations for the hidden layer are:

$$z_h = [1.8, 0.9]$$

Now, apply the **ReLU** activation function:

$$a_h = \text{ReLU}(z_h) = [1.8, 0.9]$$

(As both are positive, the activation is simply the same as the pre-activation.)

So, the output of the hidden layer is:

$$a_h = [1.8, 0.9]$$

1.2 Output Layer

Next, we compute the pre-activation of the output layer:

$$z_o = v \cdot a_h + b^{(2)} = \begin{bmatrix} 0.4 & 0.7 \end{bmatrix} \begin{bmatrix} 1.8 \\ 0.9 \end{bmatrix} + 0.2$$

Calculating the dot product:

$$z_o = (0.4 \cdot 1.8) + (0.7 \cdot 0.9) + 0.2 = 0.72 + 0.63 + 0.2 = 1.55$$

Now, apply the **sigmoid** activation function:

$$y_{\text{pred}} = \sigma(z_o) = \frac{1}{1 + e^{-1.55}} \approx 0.824$$

So, the predicted output is:

$$y_{\text{pred}} = 0.824$$

2. Loss Calculation

The loss function is **Mean Squared Error (MSE)**:

$$L = \frac{1}{2}(y_{\text{true}} - y_{\text{pred}})^2$$

$$L = \frac{1}{2}(1.0 - 0.824)^2 = \frac{1}{2}(0.176)^2 = 0.0155$$

So, the loss is:

$$L = 0.0155$$

3. Backpropagation

3.1 Gradient at Output Layer

Now, we compute the gradient of the loss with respect to the output layer weights v.

First, we calculate the derivative of the loss with respect to y_{pred}:

$$\frac{\partial L}{\partial y_{\text{pred}}} = -(y_{\text{true}} - y_{\text{pred}}) = -(1.0 - 0.824) = -0.176$$

Next, compute the derivative of the sigmoid activation function:

$$f'(z_o) = \sigma(z_o)(1 - \sigma(z_o)) = 0.824 \times (1 - 0.824) = 0.824 \times 0.176 = 0.145$$

Now, compute the gradient of the loss with respect to the output layer weights v_k:

$$\frac{\partial L}{\partial v_k} = \frac{\partial L}{\partial y_{\text{pred}}} \cdot f'(z_o) \cdot a_{h_k}$$

For each weight v_1 and v_2:

- $\frac{\partial L}{\partial v_1} = (-0.176) \times 0.145 \times 1.8 = -0.0461$
- $\frac{\partial L}{\partial v_2} = (-0.176) \times 0.145 \times 0.9 = -0.0229$

3.2 Gradient at Hidden Layer

Now, backpropagate the gradient to the hidden layer:

$$\frac{\partial L}{\partial a_{h_k}} = \sum_k \frac{\partial L}{\partial v_k} \cdot v_k$$

For each activation a_{h1} and a_{h2}:

- $\frac{\partial L}{\partial a_{h1}} = (-0.0461 \times 0.4) + (-0.0229 \times 0.7) = -0.01844 - 0.01603 = -0.03447$
- $\frac{\partial L}{\partial a_{h2}} = (-0.0461 \times 0.7) + (-0.0229 \times 0.3) = -0.03227 - 0.00687 = -0.03914$

Next, compute the gradient with respect to the hidden layer pre-activations z_h using the ReLU derivative. Since both z_{h1} and z_{h2} are positive, the derivative of ReLU is 1 for both:

$$\frac{\partial L}{\partial z_{h1}} = \frac{\partial L}{\partial a_{h1}} \cdot f'(z_{h1}) = -0.03447 \times 1 = -0.03447$$

$$\frac{\partial L}{\partial z_{h2}} = \frac{\partial L}{\partial a_{h2}} \cdot f'(z_{h2}) = -0.03914 \times 1 = -0.03914$$

Finally, calculate the gradients with respect to the hidden layer weights w_k:

$$\frac{\partial L}{\partial w_{k1}} = \frac{\partial L}{\partial z_{h1}} \cdot x_1 = -0.03447 \cdot 1 = -0.03447$$

$$\frac{\partial L}{\partial w_{k2}} = \frac{\partial L}{\partial z_{h1}} \cdot x_2 = -0.03447 \cdot 2 = -0.06894$$

$$\frac{\partial L}{\partial w_{k3}} = \frac{\partial L}{\partial z_{h2}} \cdot x_1 = -0.03914 \cdot 1 = -0.03914$$

$$\frac{\partial L}{\partial w_{k4}} = \frac{\partial L}{\partial z_{h2}} \cdot x_2 = -0.03914 \cdot 2 = -0.07828$$

4. Weight Updates

Now, we update the weights using **gradient descent**:

Update Output Layer Weights:

- $v_1 = v_1 - \alpha \cdot \frac{\partial L}{\partial v_1} = 0.4 - 0.01 \cdot (-0.0461) = 0.4 + 0.000461 = 0.400461$
- $v_2 = v_2 - \alpha \cdot \frac{\partial L}{\partial v_2} = 0.7 - 0.01 \cdot (-0.0229) = 0.7 + 0.000229 = 0.700229$