Backpropagation of Simple Neural Network

Consider a simple neural network with:

- An input layer
- An output layer

1. Neural Network Structure

- **Input Layer:** $\mathbf{x} = (x_1, x_2, \ldots, x_n)$
- **Output Layer:** y (single output neuron)

2. Forward Pass

In the forward pass, we compute the output of the network using the current weights and biases.

Steps:

1. Calculate the Weighted Sum:

- Weights: $\mathbf{W}$ (vector of weights $w_1, w_2, \ldots, w_n$)
- Bias: b
- Activation Function (e.g., sigmoid): σ

The output before applying the activation function:

$$z = \sum_i w_i x_i + b$$

2. Apply Activation Function:

The predicted output:

$$\hat{y} = \sigma(z)$$

3. Backward Pass (Backpropagation)

In the backward pass, we compute the gradient of the loss function with respect to each weight and bias, and update them accordingly.

Steps:

1. Compute the Loss:

Loss function (e.g., Mean Squared Error):

$$L = \frac{1}{2}(\hat{y} - y)^2$$

2. Calculate the Gradient of the Loss with Respect to $\hat{y}$:

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y$$

3. Calculate the Gradient of $\hat{y}$ with Respect to z:

For a sigmoid activation function $\sigma(z) = \frac{1}{1+e^{-z}}$, the derivative is:

$$\sigma'(z) = \hat{y}(1 - \hat{y})$$

Therefore:

$$\frac{\partial \hat{y}}{\partial z} = \hat{y}(1 - \hat{y})$$

4. Calculate the Gradient of z with Respect to Each Weight w_i:

$$\frac{\partial z}{\partial w_i} = x_i$$

5. Calculate the Gradient of z with Respect to the Bias b:

$$\frac{\partial z}{\partial b} = 1$$

6. Combine the Gradients Using the Chain Rule:

- Gradient of the loss with respect to each weight w_i:

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_i} = (\hat{y} - y) \cdot \hat{y}(1 - \hat{y}) \cdot x_i$$

- Gradient of the loss with respect to the bias b:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial b} = (\hat{y} - y) \cdot \hat{y}(1 - \hat{y})$$

4. Update the Weights and Bias

Update the weights and bias using gradient descent:

- Learning rate: η
- Update rule for weights:

$$w_i \leftarrow w_i - \eta \frac{\partial L}{\partial w_i}$$

- Update rule for bias:

$$b \leftarrow b - \eta \frac{\partial L}{\partial b}$$

Example

Let's illustrate with a concrete example:

- Inputs: $\mathbf{x} = (0.5, 0.3)$
- Weights: $\mathbf{W} = (0.2, 0.4)$
- Bias: $b = 0.1$
- True output: $y = 1$
- Learning rate: $\eta = 0.01$

1. Forward Pass:

$$z = 0.2 \cdot 0.5 + 0.4 \cdot 0.3 + 0.1 = 0.35$$
$$\hat{y} = \sigma(0.35) \approx 0.5866$$

2. Backward Pass:

$$\frac{\partial L}{\partial \hat{y}} = 0.5866 - 1 = -0.4134$$
$$\frac{\partial \hat{y}}{\partial z} = 0.5866 \cdot (1 - 0.5866) \approx 0.2425$$
$$\frac{\partial z}{\partial w_1} = 0.5, \quad \frac{\partial z}{\partial w_2} = 0.3, \quad \frac{\partial z}{\partial b} = 1$$
$$\frac{\partial L}{\partial w_1} = -0.4134 \cdot 0.2425 \cdot 0.5 \approx -0.0501$$
$$\frac{\partial L}{\partial w_2} = -0.4134 \cdot 0.2425 \cdot 0.3 \approx -0.0300$$
$$\frac{\partial L}{\partial b} = -0.4134 \cdot 0.2425 \approx -0.1002$$

3. Update Weights and Bias:

$$w_1 \leftarrow 0.2 - 0.01 \cdot (-0.0501) \approx 0.2005$$

$$w_2 \leftarrow 0.4 - 0.01 \cdot (-0.0300) \approx 0.4003$$

$$b \leftarrow 0.1 - 0.01 \cdot (-0.1002) \approx 0.1010$$

After updating, the weights and bias are slightly adjusted to reduce the error in the next iteration. This process is repeated for many iterations until the network learns to produce outputs close to the desired values.