Backpropagation Overview

1 Backpropagation Overview

Backpropagation is an algorithm used to train neural networks by updating the weights and biases of each layer based on the gradient of the loss function with respect to each parameter. It works by propagating the error (the loss) backwards from the output layer to the input layer, layer by layer.

2 1. Forward Pass

In the forward pass, for each layer i:

- The pre-activation Z_i is computed as:

  $$Z_i = W_i \cdot A_{i-1} + b_i$$

  where:

  - W_i is the weight matrix of layer i.
  - A_{i-1} is the output (activation) of the previous layer $i-1$ (or the input to the network for the first layer).
  - b_i is the bias of layer i.

- The activation A_i is computed by applying the activation function f to Z_i:

  $$A_i = f(Z_i)$$

  where $f(Z_i)$ is the activation function (e.g., ReLU, Sigmoid, etc.).

3 2. Backward Pass (Backpropagation)

In the backward pass, for each layer i, we need to compute the gradients and propagate the error back to the previous layers. There are two main components we need to calculate:

3.1 2.1 Gradient with Respect to the Output of Layer i (Activations)

- dA_i is the gradient of the total loss E_{tot} with respect to the activations A_i. This gradient is passed backward from the loss function (if it's the output layer) or from the subsequent layer (if it's a hidden layer).

3.2 2.2 Gradient with Respect to the Pre-Activation (Z_i)

- The gradient of the loss with respect to Z_i (the pre-activation of layer i) is:

  $$dZ_i = dA_i \cdot f'(Z_i)$$

  where:

  - $f'(Z_i)$ is the derivative of the activation function applied to Z_i.

3.3 2.3 Gradient with Respect to Weights and Biases

- The gradients of the loss with respect to the weights W_i and biases b_i are:

  $$dW_i = dZ_i \cdot A_{i-1}^T$$

  $$db_i = dZ_i$$

  where A_{i-1}^T is the transpose of the activations from the previous layer.

3.4 2.4 Gradient with Respect to the Input (X_i)

- The gradient of the loss with respect to the input of layer i (which is X_i, the input to the layer i) is:

  $$dX_i = W_i^T \cdot dZ_i$$

  This is the gradient passed back to the previous layer as dA_{i-1}, which will be used by the previous layer to compute its own gradients.

4 Key Points

- **Forward Pass**:
  - Compute activations for each layer.
  - Output of layer i: $A_i = f(Z_i)$.
- **Backward Pass**:
  - Compute dZ_i, the gradient with respect to the pre-activation.
  - Compute dW_i and db_i to update the weights and biases.
  - Compute dX_i (or dA_{i-1}), which is the gradient of the loss with respect to the input to the current layer, and pass it back to the previous layer.

5 The Flow of Gradients

- dA_i: Loss gradient with respect to the current layer's activation (output).
- dZ_i: Loss gradient with respect to the current layer's pre-activation (input to the activation function).
- dX_i: Loss gradient with respect to the input of layer i, which is passed back to the previous layer.

6 Summary

For layer i:

- **Step 1**: Calculate the loss gradient with respect to the activation dA_i (this comes from the next layer or the loss function).
- **Step 2**: Compute the gradient with respect to the pre-activation $dZ_i = dA_i \cdot f'(Z_i)$.
- **Step 3**: Compute the gradients with respect to weights and biases:

$$dW_i = dZ_i \cdot A_{i-1}^T$$

$$db_i = dZ_i$$

- **Step 4**: Compute the gradient with respect to the input, which is passed back to the previous layer:

$$dX_i = W_i^T \cdot dZ_i$$

This process is repeated layer by layer during training to adjust the weights and biases to minimize the total loss. The gradients allow us to know how to change the weights and biases to reduce the error, and the backpropagation algorithm efficiently computes these gradients across all layers.