Backpropagation Mechanism

The Learning Algorithm of Neural Networks

Sina Moghimi, Autumn 2024

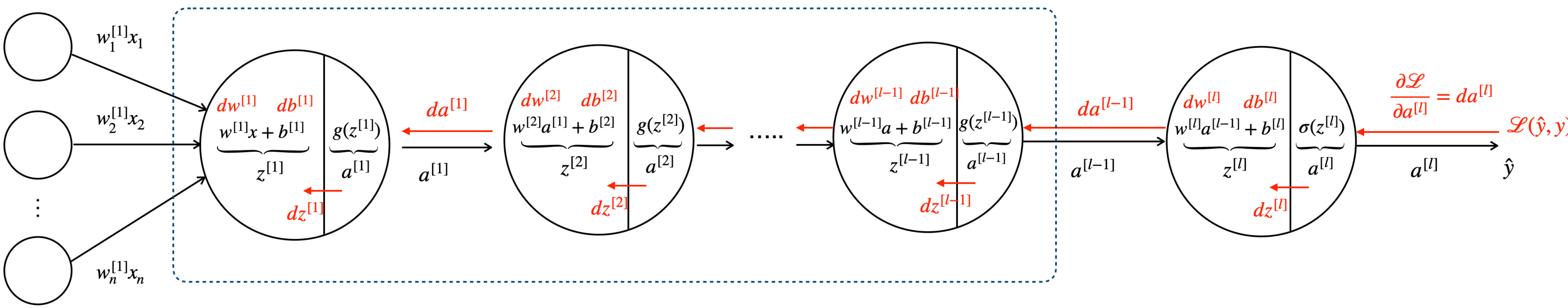
Outline

- Introduction
- Error Calculation
- Gradient Computation Basics
- Chain Rule of Calculus
- Weight Update Mechanism
- Computational Process
- Challenges
- Optimization Techniques

МФТИ

Introduction

- Short for **backward propagation of errors**
    - Calculate the error
    - Compute the gradient of the error with respect to each parameter and propagate the error backward through the layers.



Activation = Relu, $g(z)$

Activation = Sigmoid, $\sigma(z)$

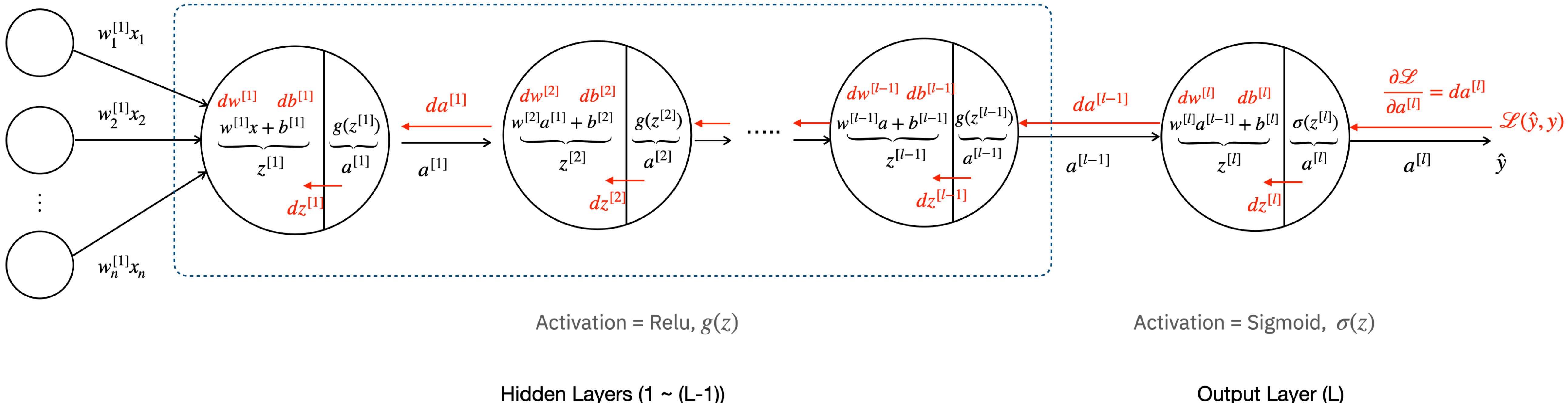
Hidden Layers (1 ~ (L-1))

Output Layer (L)

https://community.deeplearning.ai/t/week-4-exercise-9-backpropagation-l-model/182118

Introduction

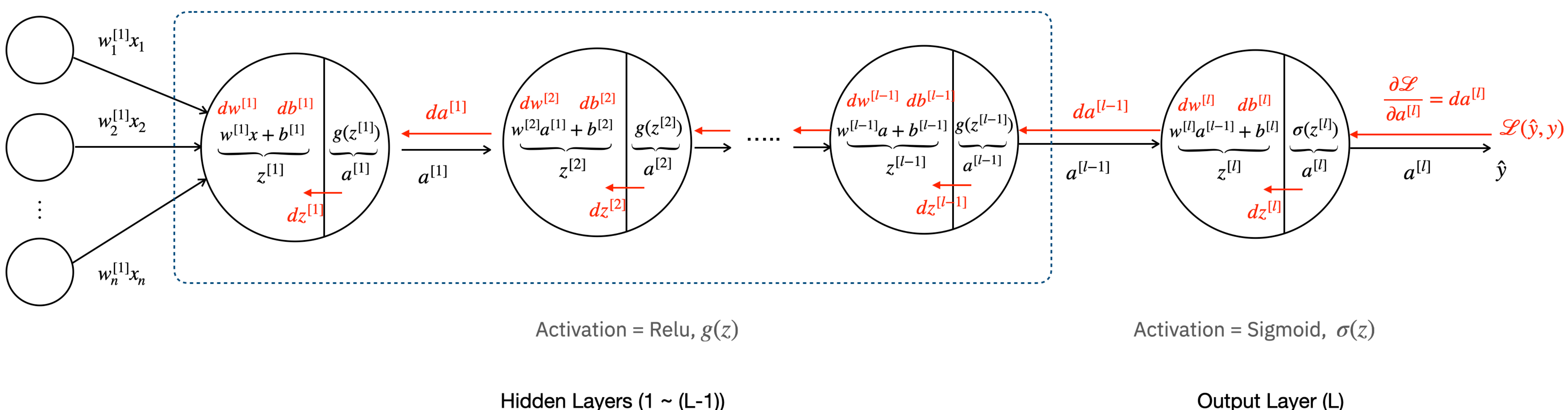
- **Purpose**
    - Modify the weights and biases to make the output closely match the ground truth (real output).
    - It enables neural networks to learn patterns in the data by updating parameters in the direction of decreasing error, iteratively improving predictions.



https://community.deeplearning.ai/t/week-4-exercise-9-backpropagation-l-model/182118

Key Components of Learning

- Loss function
- Gradients
- Chain Rule
- Optimization Algorithm
- Learning Rate
- Weight Updates



https://community.deeplearning.ai/t/week-4-exercise-9-backpropagation-l-model/182118

Loss Function

- Quantifies the difference between prediction and the ground truth.
- **Common Loss Functions**
  - **Mean Squared Error**:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

  - **Cross-Entropy Loss:**

$$H = -\frac{1}{n}\sum_{i=1}^{n} Y_i \,.\, log(p(\hat{Y}_i))$$

  - **Mean Absolute Error:**

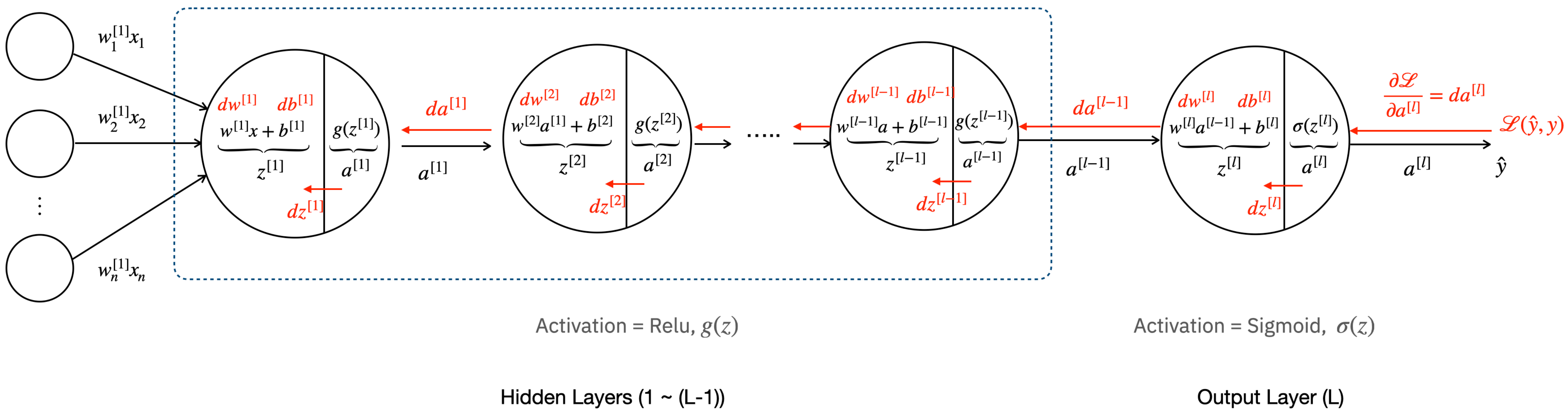
$$MAE = \frac{1}{n}\sum_{i=1}^{n} | Y_i - \hat{Y}_i |$$

- $\underline{Y}$ is the **ground truth**, and $\underline{\hat{Y}}$ is the **prediction**.

Gradients

- Represent the sensitivity of the loss function to changes in each parameter

- It is done using chain rule during backpropagation

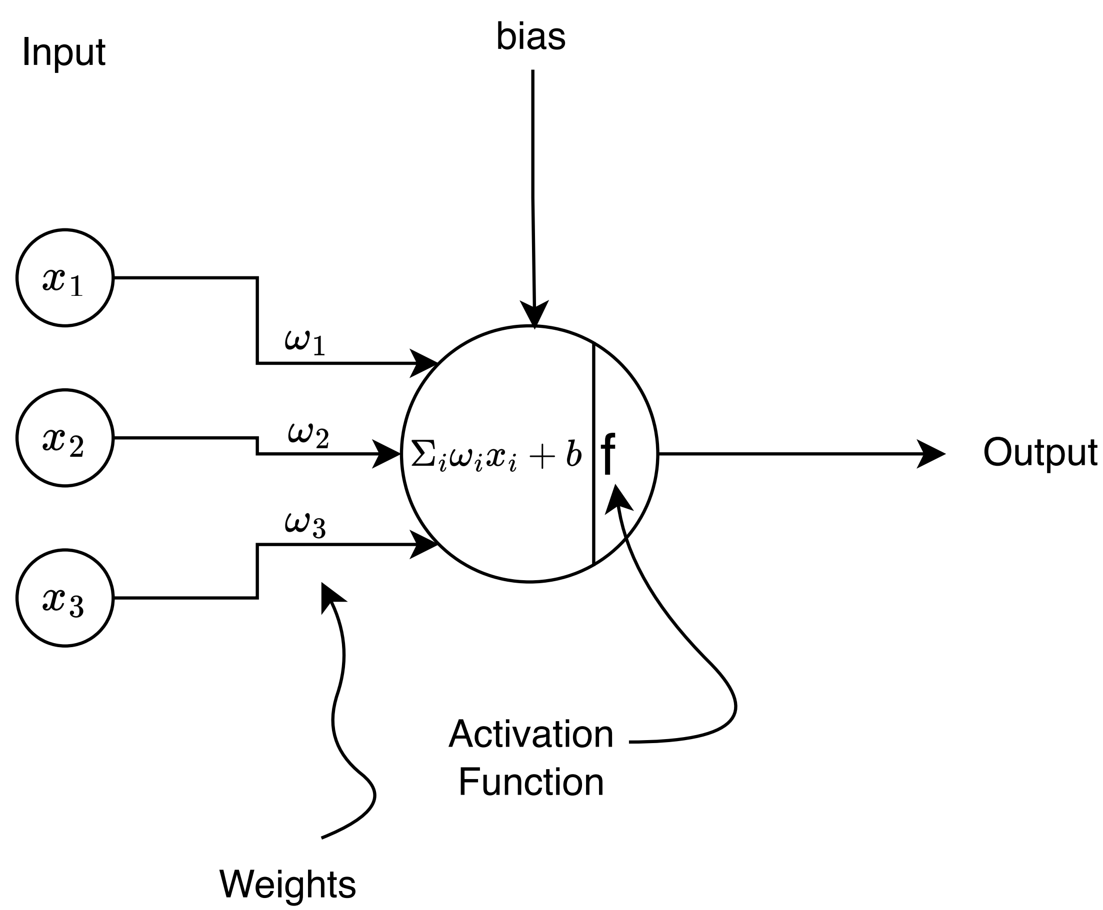


МФТИ

Chain Rule

- Error = Output - GT
- Loss = MSE(Error)

- $$\frac{\delta\ Loss}{\delta x_1} = \frac{\delta\ Loss}{\delta f} \cdot \frac{\delta f}{\delta z} \cdot \frac{\delta z}{\delta \omega_1} \cdot \frac{\delta \omega_1}{\delta x_1}$$

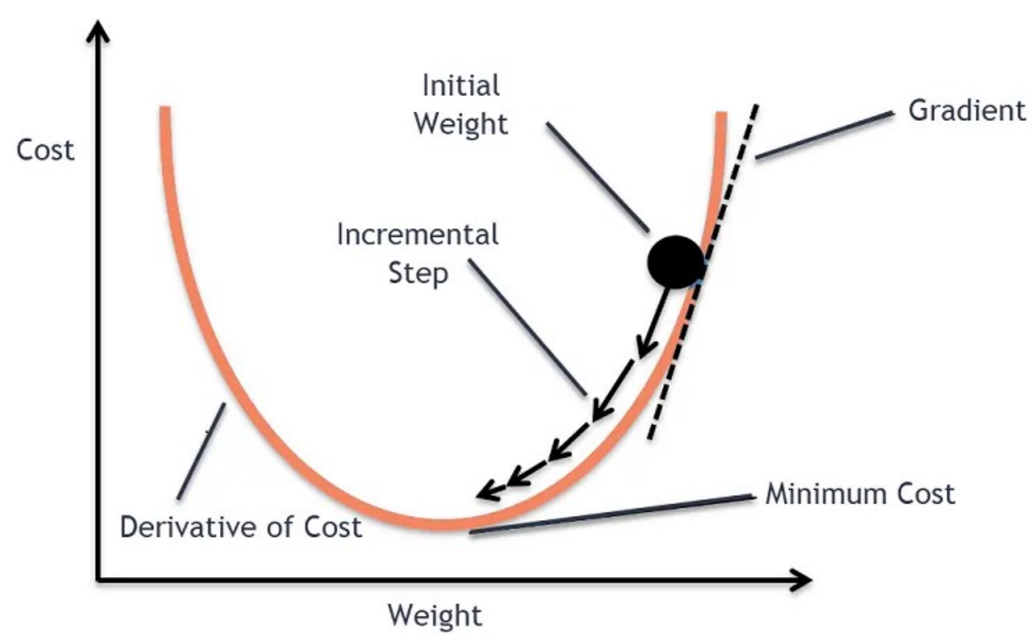


Weight Update and Learning Rate

- **Learning Rate (η)**
  - Determines the step size when updating the parameters
  - Balances the convergence speed and stability

- **Weight Update**
  - $\omega = \omega - \eta \, . \dfrac{\delta Loss}{\delta \omega}$

- **Purpose**:
  - Find the local optimum of the loss

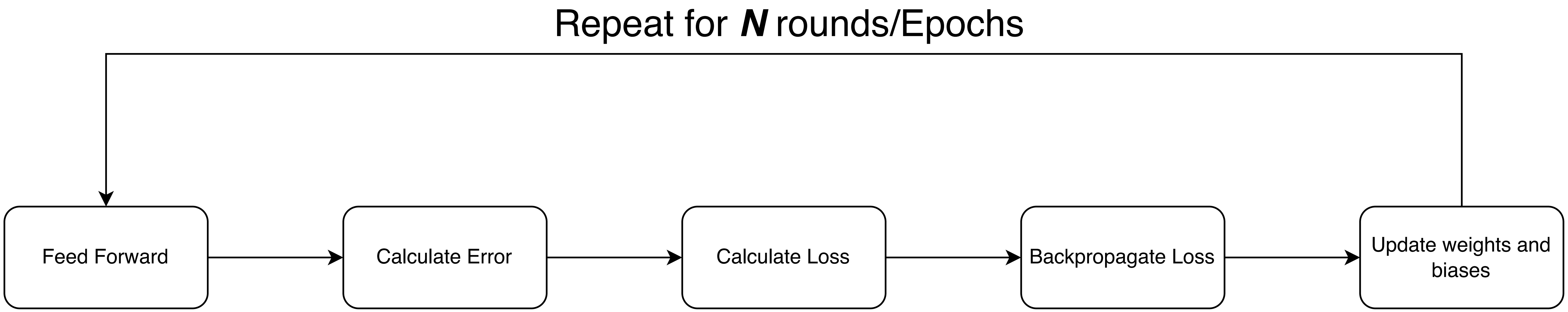


Optimizers

- Optimization algorithms are used to handle the backpropagation for us
- Each optimization algorithms has it own way of updating weights and biases.
- **Some Famous Optimizers**:
  - Stochastic Gradient Descent (**SGD**)
  - Adaptive Moment estimation (**ADAM**)
  - Adaptive Gradient (**AdaGrad**)

МФТИ

PipeLine


Repeat for ***N*** rounds/Epochs
Feed Forward
Calculate Error
Calculate Loss
Backpropagate Loss
Update weights and biases