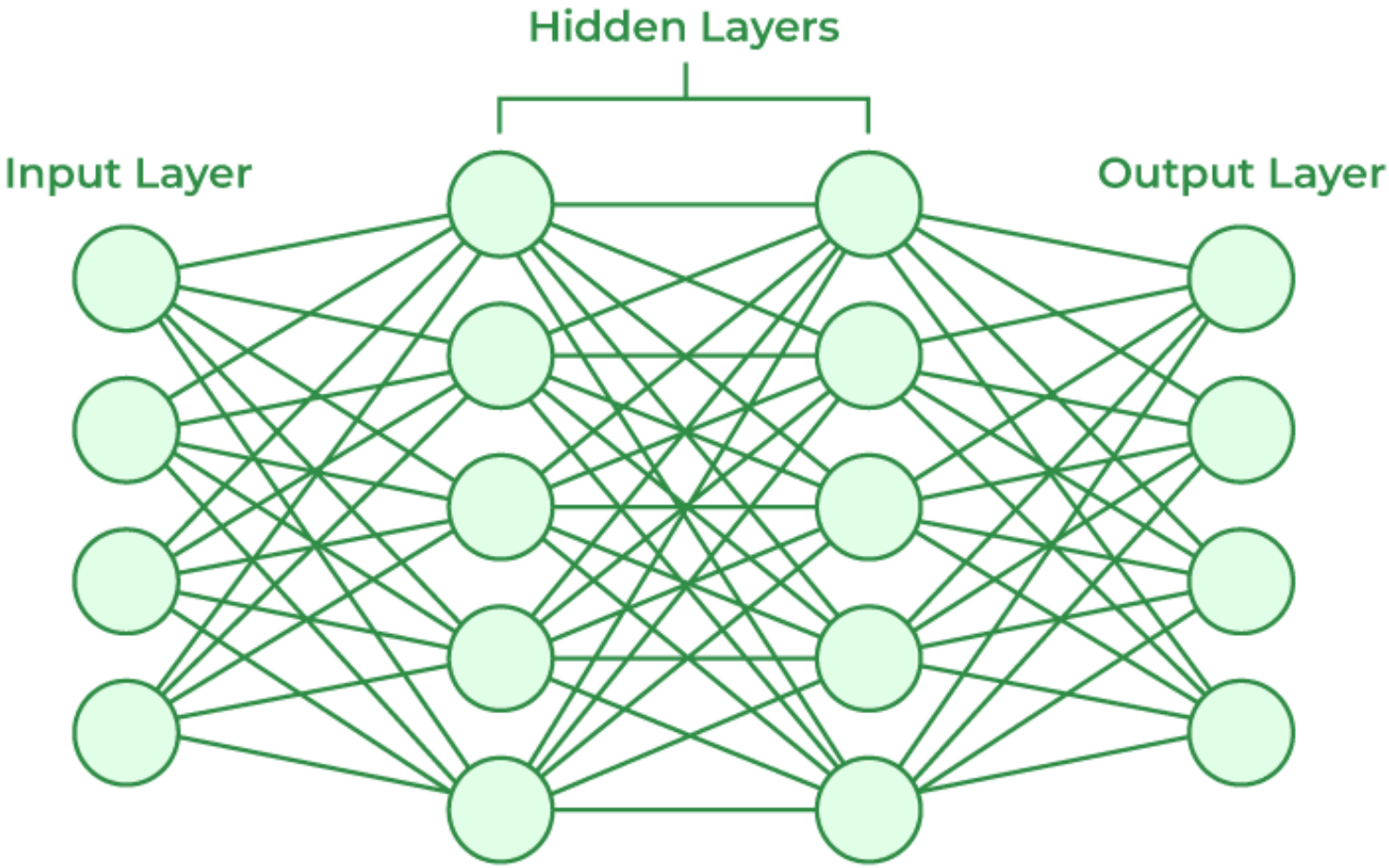
CSE463: Computer Vision

Neural Networks: Basics and Functionalities

Contents

Neural Network Basics



Neural Network Basics

- A neural network is a computational model inspired by the human brain, consisting of interconnected nodes (neurons) organized in layers to process and learn from data.

- **Structure:**
  - Input Layer: Receives raw data as input.
  - Hidden Layers: Perform computations to extract patterns.
  - Output Layer: Produces the final result or prediction.

- **Key Components:**
  - Weights (w): Adjusted during learning to minimize error.
  - Bias (b): Added to ensure the model can shift predictions.
  - Activation Function ($f(x)$): Introduces non-linearity (e.g., ReLU, sigmoid).

Neural Network Functionality

- **Forward Propagation:**
  - The input data passes through layers, undergoing linear transformations and activation functions.
  - Example for one layer:

$$a^{(l)} = f(W^{(l)}a^{(l-1)} + b^{(l)})$$

    - a(l−1): Activations from the previous layer.
    - W(l): Weight matrix for layer lll.
    - b(l): Bias vector for layer lll.

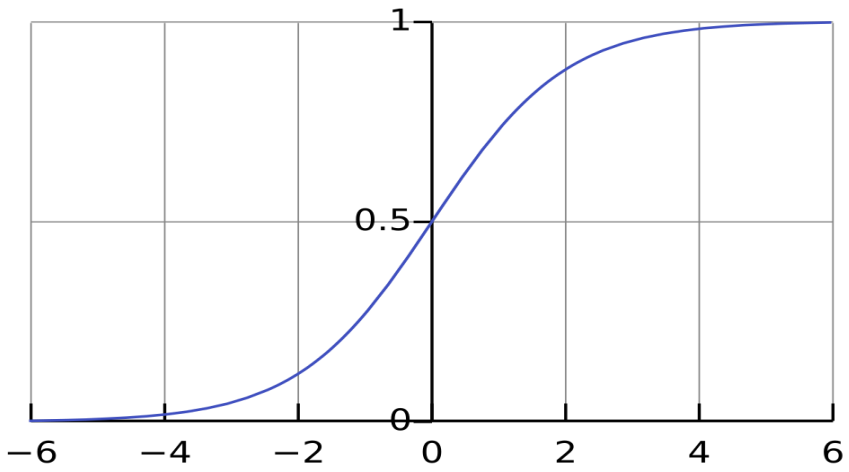
Key Features of Neural Networks

- **Learning from Data:** Neural networks adjust their weights and biases based on the input data to minimize errors.
- **Generalization:** Can predict outputs for unseen data by learning patterns during training.
- **Non-Linear Modelling:** Can model complex relationships using non-linear activation functions.
- **Parallelism:** Many computations are performed simultaneously, enabling efficient processing of large datasets.

Activation Functions

- **Purpose:** Introduces non-linearity to enable the network to learn complex patterns. Examples include-

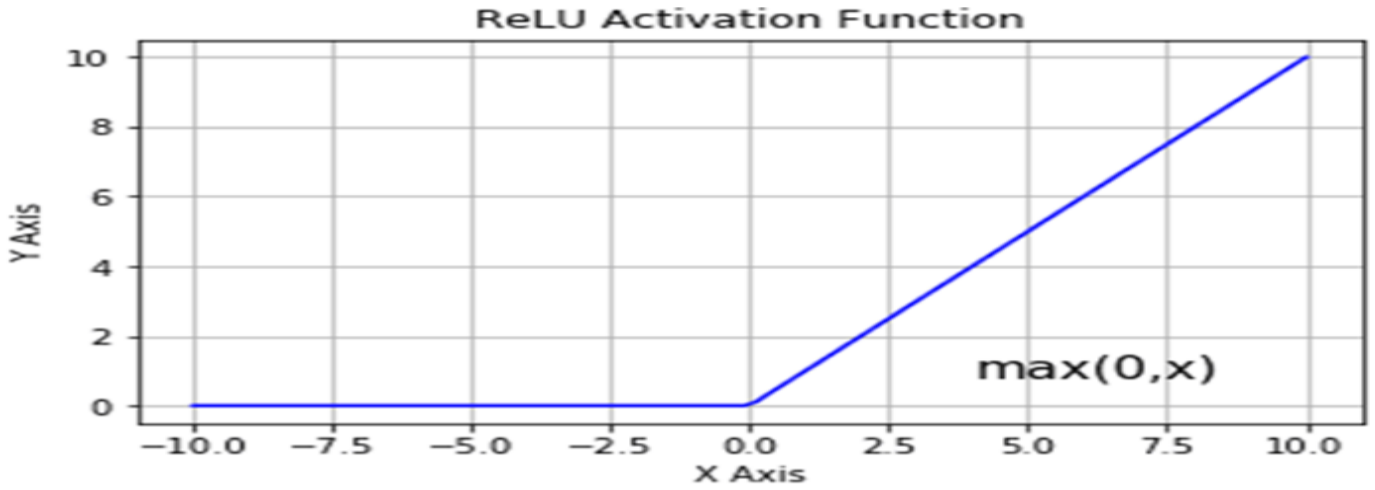
  - **Sigmoid Function:**
    - The sigmoid activation function maps input values to an output range between 0 and 1.
    - It is commonly used in binary classification problems. The function is defined as:



$$f(x) = \frac{1}{1 + e^{-x}}$$

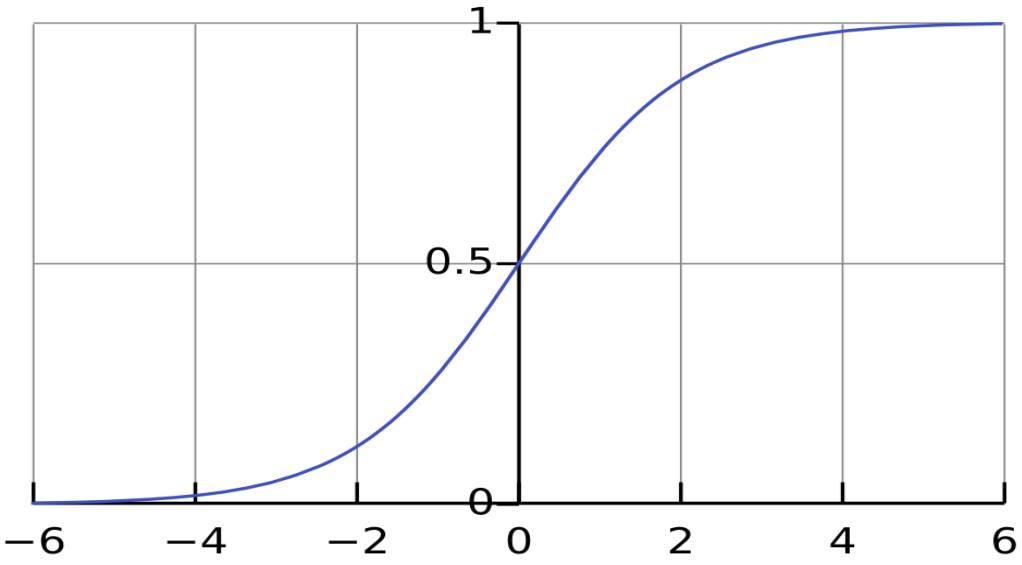
Activation Functions

- **ReLU (Rectified Linear Unit):**
  - Range: [0, ∞)
  - Efficiency: Computationally simple and fast.
  - Non-Linear: Allows the model to learn complex patterns.
  - Limitation: Can suffer from the "dying ReLU" problem, where neurons output 0 for all inputs if weights are updated poorly.
  - ReLU is defined as: $f(x) = \max(0, x)$



Activation Functions

- **Softmax Function:**
  - It is used in multi-class classification tasks to convert raw scores (logits) into probabilities.
  - Range: [0, 1]
  - It ensures the sum of probabilities across all classes equals 1.
  - Sigmoid is defined as- $$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{1+e^{x}}$$

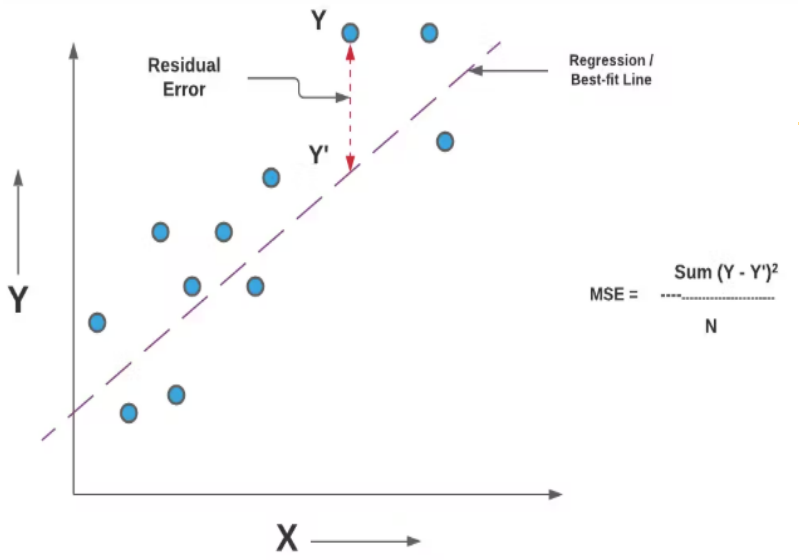


Training Neural Networks

- **Initialization:** Randomly initialize weights and biases.
- **Forward Pass:** Compute the output using current weights and biases.
- **Loss Calculation:** Evaluate error using a loss function, L(y_{true}, y_{pred}).
- **Backward Pass:** Compute gradients of the loss function with respect to weights and biases.
- **Update Parameters:** Use an optimizer (e.g., SGD, Adam) to update weights.
- **Repeat:** Continue until the loss converges or a stopping criterion is met.

Loss Function

- A Loss Function quantifies the difference between the predicted output and the actual target output. It measures the error in the model's predictions.
- Common Loss Functions
  - **Mean Squared Error (MSE)**: Used for regression tasks.



Mean Squared Error

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

n = Number of Data points

Yi = Observed Values

Y = Predicted Values

Loss Function

- **Cross-Entropy Loss**: Used for classification tasks.

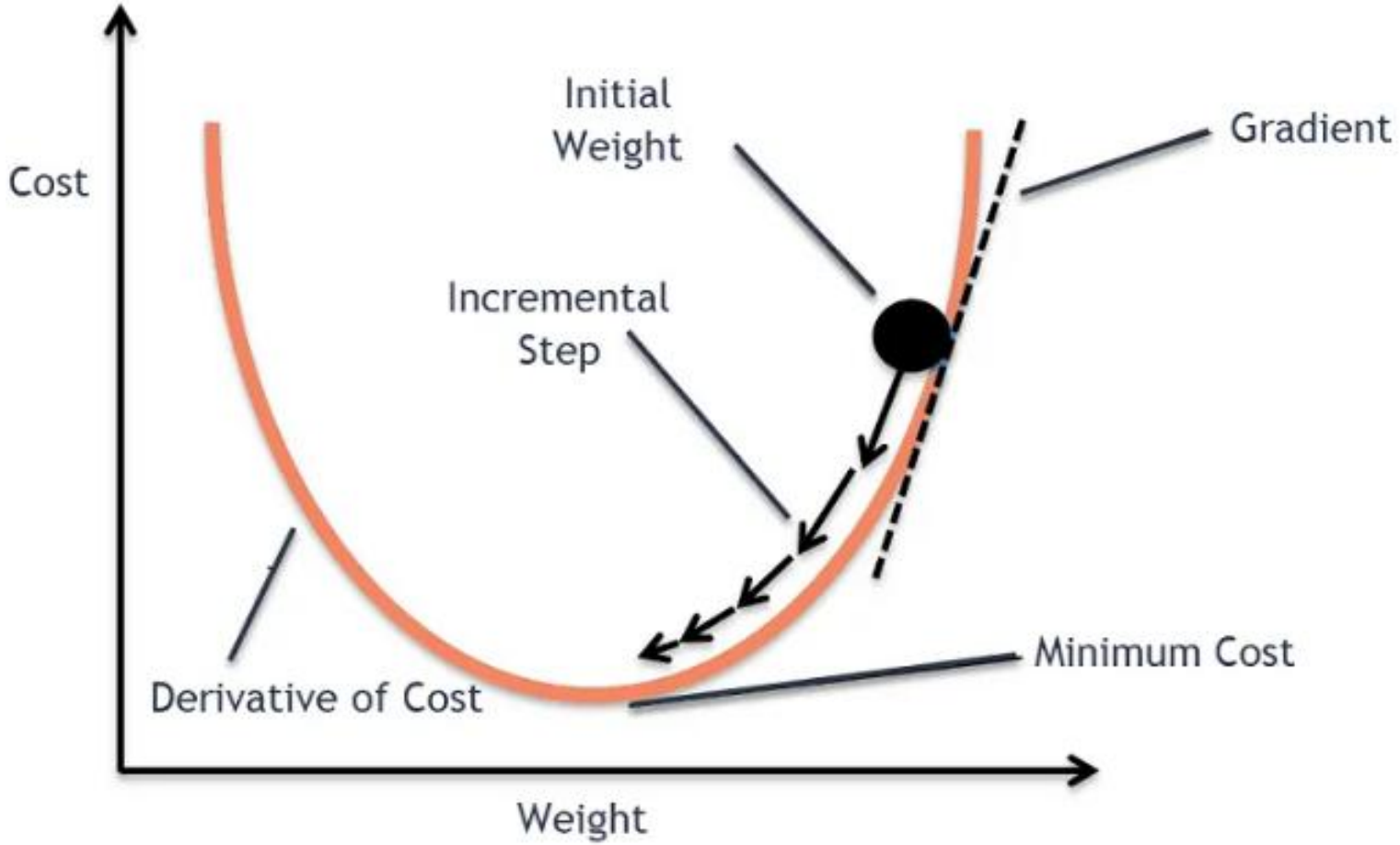
- **Formula** (for a single sample):

$$CE = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

  - yi: True label (one-hot encoded, 1 for the correct class, 0 otherwise).
  - yi^hat: Predicted probability for class i.
  - C: Number of classes.

Gradient Descent

- **Gradient Descent is a core concept for understanding how neural networks are trained.**



Gradient Descent

- Gradient Descent is an optimization algorithm used to minimize a loss function by iteratively updating the model's parameters in the direction of the negative gradient.

Core Idea

- Minimize the loss function J(θ).
- Update parameters θ to reduce J(θ):

$$\theta = \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

Here, **θ:** Model parameters (weights, biases).

η: Learning rate (step size).

∂J(θ)/∂θ: Gradient of the loss function with respect to θ.

Backpropagation

- Minimizes the error between the predicted and actual outputs using a loss function (e.g., Mean Squared Error or Cross-Entropy).

- Gradient Descent is used to update weights and biases

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial L}{\partial w_{ij}}$$

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

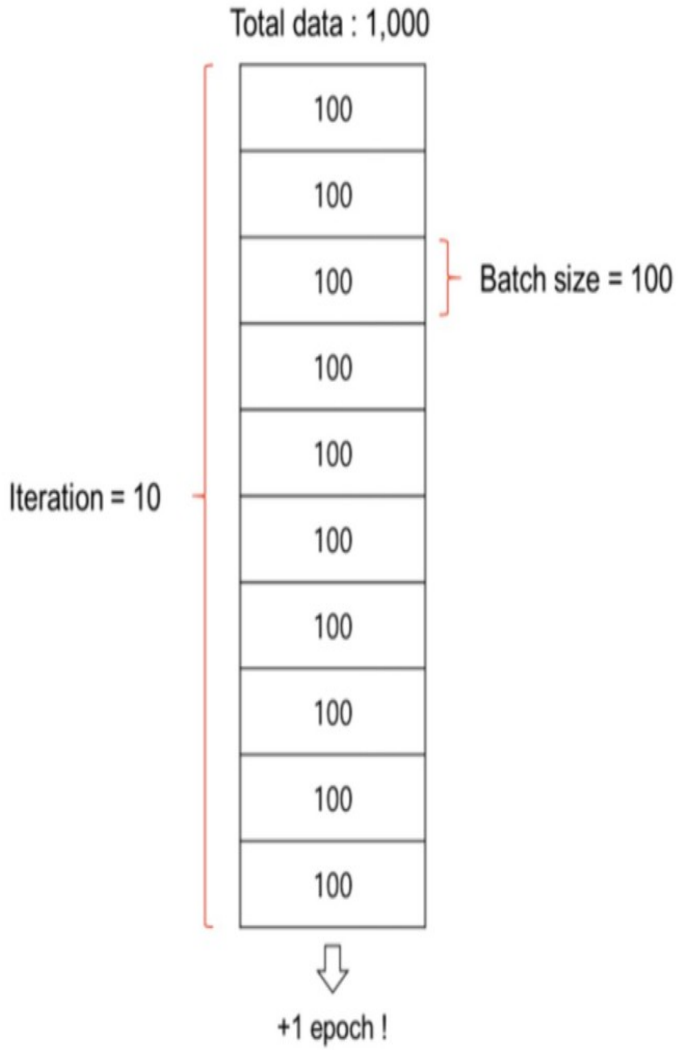
  - η: Learning rate.
  - L: Loss function.

Batches and Epochs

Epoch: Epoch is when an entire dataset is passed forward and backward through the neural network only once. An epoch describes the number of times the algorithms sees the entire data.

Batch: Batch size is total number of training examples in forward/backward pass. Batch size and number of batches is two different things. The higher the batch size the more memory space you need.

- **Iteration is the number of batches needed to complete one epoch. 1 Iteration = 1 forward pass + 1 backward pass.**



Types of Neural Networks

- **Feedforward Neural Network (FNN):**
  - Simplest type of neural network.
  - Data flows in one direction: input → hidden layers → output.
  - **Use Case:** Regression and classification tasks.

- **Convolutional Neural Network (CNN):**
  - Specializes in spatial data like images.
  - **Key Features:** Convolutional layers, pooling layers.
  - **Use Case:** Image recognition, object detection.

- **Recurrent Neural Network (RNN)**
  - Processes sequential data using loops in architecture.
  - **Key Features:** Hidden state, memory of previous inputs.
  - **Use Case:** Time series prediction, natural language processing (NLP)