ARTIFICIAL NEURAL NETWORKS AND DEEP LEARNING

Biological Inspiration and Perceptrons

Joshua Llano

Universitat Politècnica de Catalunya

Table of contents

Introduction

Biological Neurons

- Biological neurons are the building blocks of the human brain and nervous system.
- They play a crucial role in processing information and enabling communication within the body.

Structure and Components of a Neuron

- Dendrites: Receive incoming signals from other neurons.
- Cell Body (Soma): Contains the nucleus and processes signals.
- Axon: Transmits signals away from the neuron.
- Axon Terminal: Communicates with other neurons or cells.

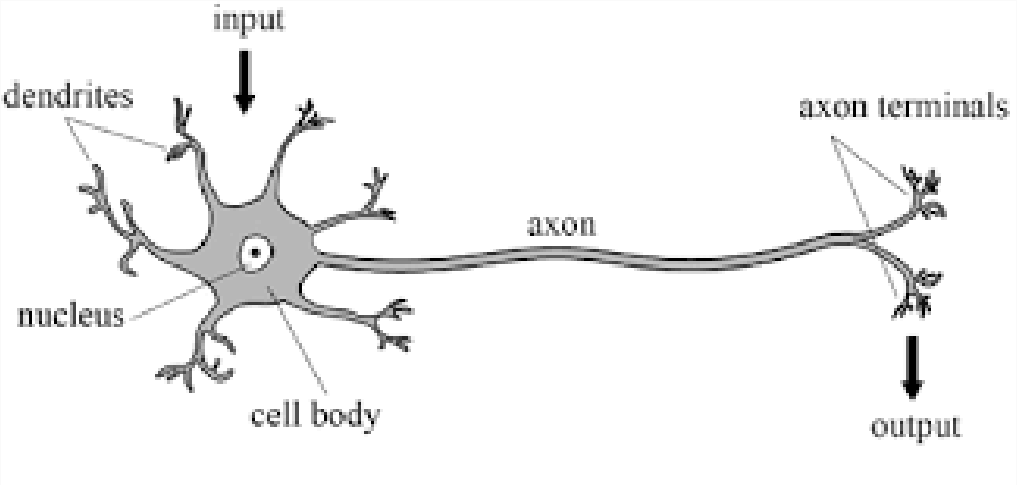


Figure 1: Anatomy of a Biological Neuron

Communication and Components of a Neuron

Neuron at Rest

- Neurons maintain a resting membrane potential.
- Ion channels regulate the flow of ions, maintaining an electrical gradient.

Action Potential

- When a stimulus is strong enough, an action potential is generated.
- It is an **all-or-nothing** electrical signal that travels down the axon.

Synaptic Transmission

- Action potentials reach the axon terminals.
- Neurotransmitters are released into the synaptic cleft.
- Neurotransmitters bind to receptors on the dendrites of the next neuron.

Significance of Understanding Neurons

- Understanding how biological neurons work is fundamental to neuroscience.
- It also serves as inspiration for artificial neural networks in AI and machine learning.

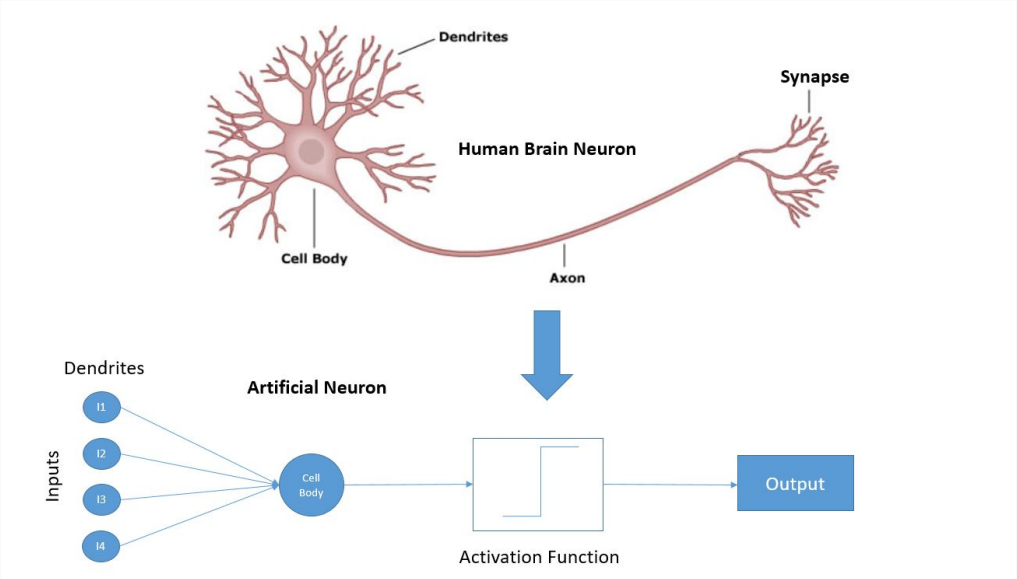


Figure 2: Biological Neuron and Artificial Neuron

Perceptrons

Introduction to Perceptrons

- Perceptrons are the fundamental building blocks of artificial neural networks.

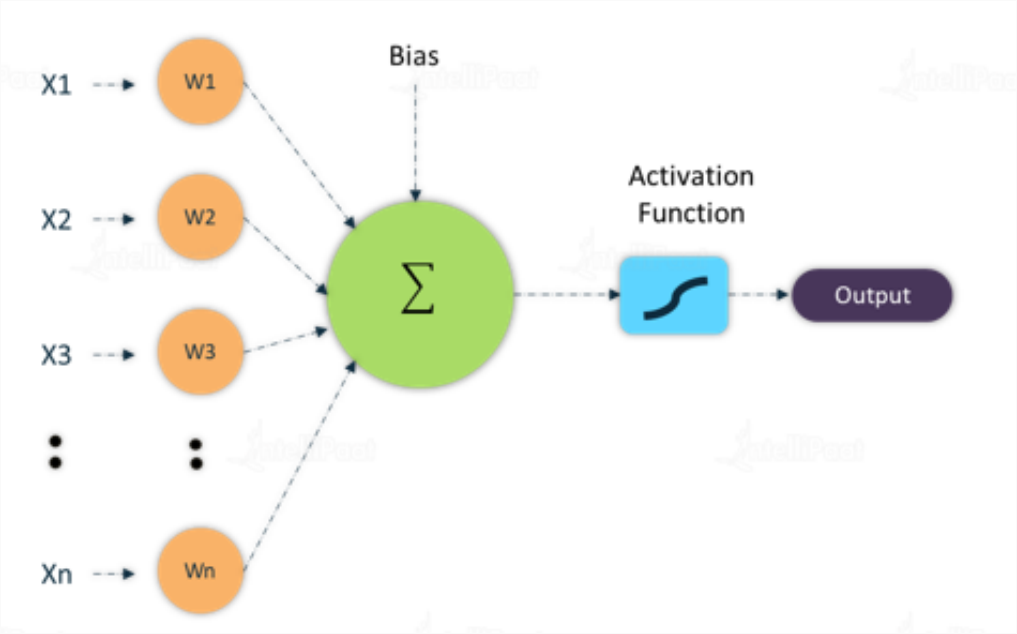


Figure 3: Components of a Perceptron

Components of a Perceptron

- Inputs: The perceptron receives input signals.
- Weights: Each input is associated with a weight, which represents its importance.
- Bias: The bias term is an offset, allowing the perceptron to account for certain biases.
- Activation Function: The activation function processes the weighted sum and produces an output.

Perceptron Operation

- The perceptron computes a weighted sum of its inputs.
  $weighted\_sum = \sum_{i=1}^{n}(input_i \cdot weight_i)$
- The bias term is added to the weighted sum.
  $z = weighted\_sum + bias$.
- The result is passed through an activation function. $f(z)$

Activation functions

Biological Neurons Are Non-Linear

- Biological neurons exhibit non-linear behavior.
- They don’t act in a purely linear way.

Role of Activation Functions

- Activation functions introduce non-linearity into neural networks.
- They enable neural networks to model complex relationships and patterns.

Types of Activation Functions

- Common Activation Functions:
  - Step function
  - Sigmoid function
  - Rectified Linear Unit (ReLU)
  - Hyperbolic Tangent (tanh)

Visualization of Activation Functions

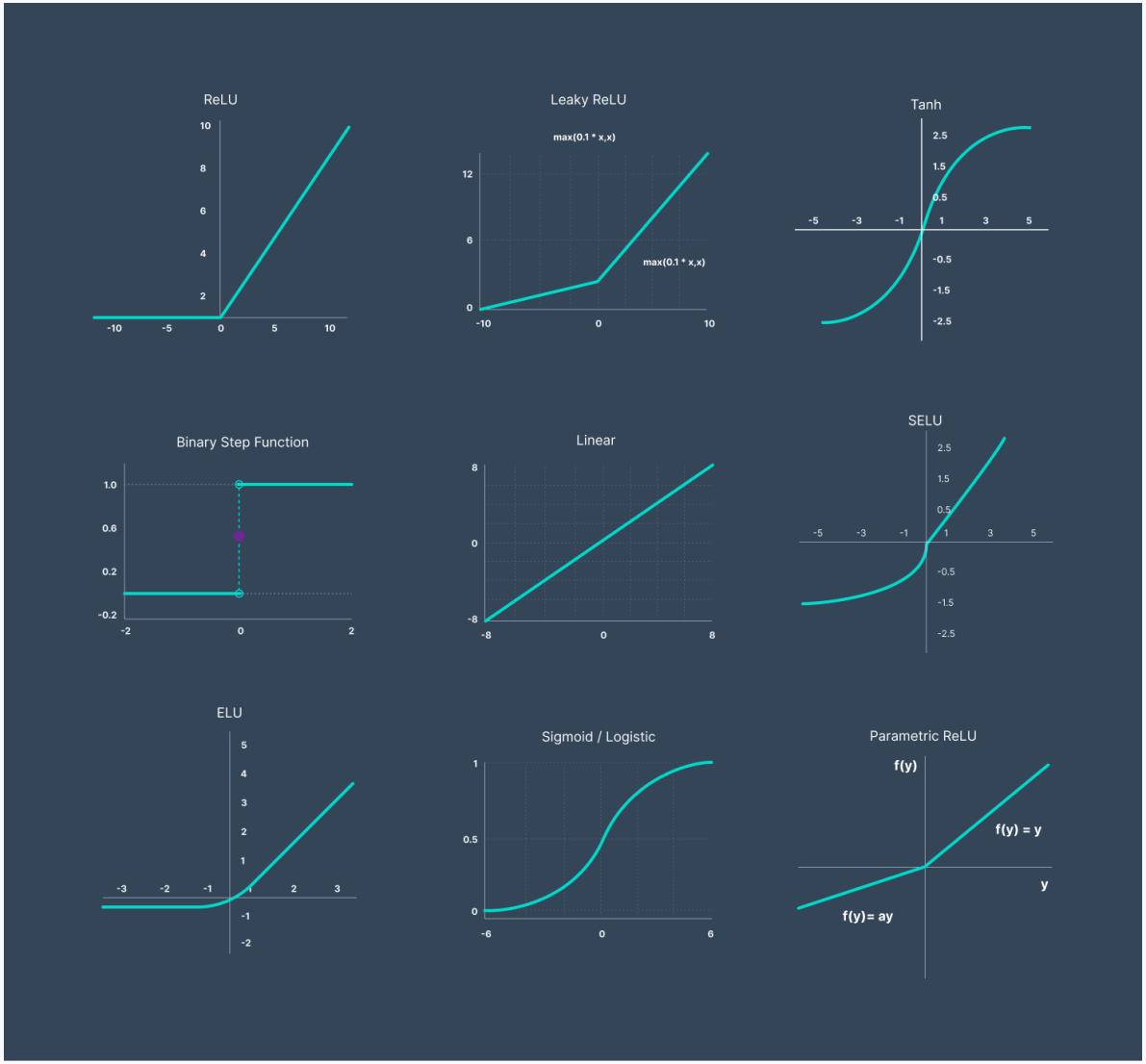


Figure 4: Visualization of Common Activation Functions

Perceptron: simple example

Perceptron Input and Weights

- Inputs: $x_1 = 0.5, x_2 = 0.2$
- Weights: $w_1 = 0.8, w_2 = -0.5$

Weighted Sum Calculation

- Calculate the weighted sum: $z = w_1 x_1 + w_2 x_2$
- $z = 0.8 \cdot 0.5 + (-0.5) \cdot 0.2 = 0.4 - 0.1 = 0.3$

Perceptron: simple example

Activation Function

- Apply the step function as the activation function.
- If z is greater than or equal to 0, the perceptron outputs 1; otherwise, it outputs 0.
- $f(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases}$
- $f(0.3) = 1$

Perceptron Output

- The perceptron's output is 1 in this example.
- It's a simple binary classification decision based on the weighted sum and the activation function.

Summary

- Perceptrons are the basic units of artificial neural networks.
- They consist of inputs, weights, bias, and activation functions.
- Activation functions play a crucial role in introducing non-linearity, similar to biological neurons.

Perceptron Learning Algorithm

Perceptron Learning Algorithm

Steps involved in perceptron learning:

1. Initialization
2. Input Presentation
3. Weighted Sum Calculation
4. Activation Function
5. Output Comparison
6. Weight Adjustment
7. Repeat

Perceptron Learning Process

Initialization

- The weights and bias are randomly initialized.
- Display the initial values (e.g., w1 = 0.2, w2 = -0.3, bias = 0.1).

Input Presentation

- Present a specific input vector, e.g., (x1 = 0.5, x2 = 0.8).
- Mention the target output (expected classification).

Weighted Sum Calculation

- Calculate the weighted sum:
  $z = (0.2 * 0.5) + (-0.3 * 0.8) + 0.1 = 0.03$

Activation Function

- Apply the Activation Function (step function in this example).
- In this case, f(0.03) = 1.

Perceptron Learning Process

Output Comparison

- Compare the perceptron's output (1) to the target output (expected classification).

Weight Adjustment (Learning)

- Weights are adjusted if outputs don't match.
- The adjusted weights are calculated using delta rule:
  $\Delta\omega_i = \alpha \cdot (t_i - f(z_i)) \cdot x_i$;
  $\Delta b = \alpha \cdot (t_i - f(z_i))$;
  where α is the learning rate; t_i is the target; $f(z_i)$ is the output after the activation function

Repeat and Convergence

- The learning process is repeated for multiple input samples.
- Over time, the perceptron learns to correctly classify inputs.

Learning Rate Overview

- The learning rate controls the size of weight updates during training.
- Define the learning rate as a small positive constant (e.g., 0.1, 0.01, 0.001).

Learning Rate Overview

Learning Rate: High (e.g., 0.1)

- Large weight updates occur, and the learning process may be unstable.

Learning Rate: Medium (e.g., 0.01)

- The weight updates are moderate, allowing for stable learning.

Learning Rate: Low (e.g., 0.001)

- The weight updates are very small, leading to slow convergence.

Learning Rate Overview

- The learning rates' impact on convergence speed and stability.
- The choice of learning rate depends on the specific problem and dataset.

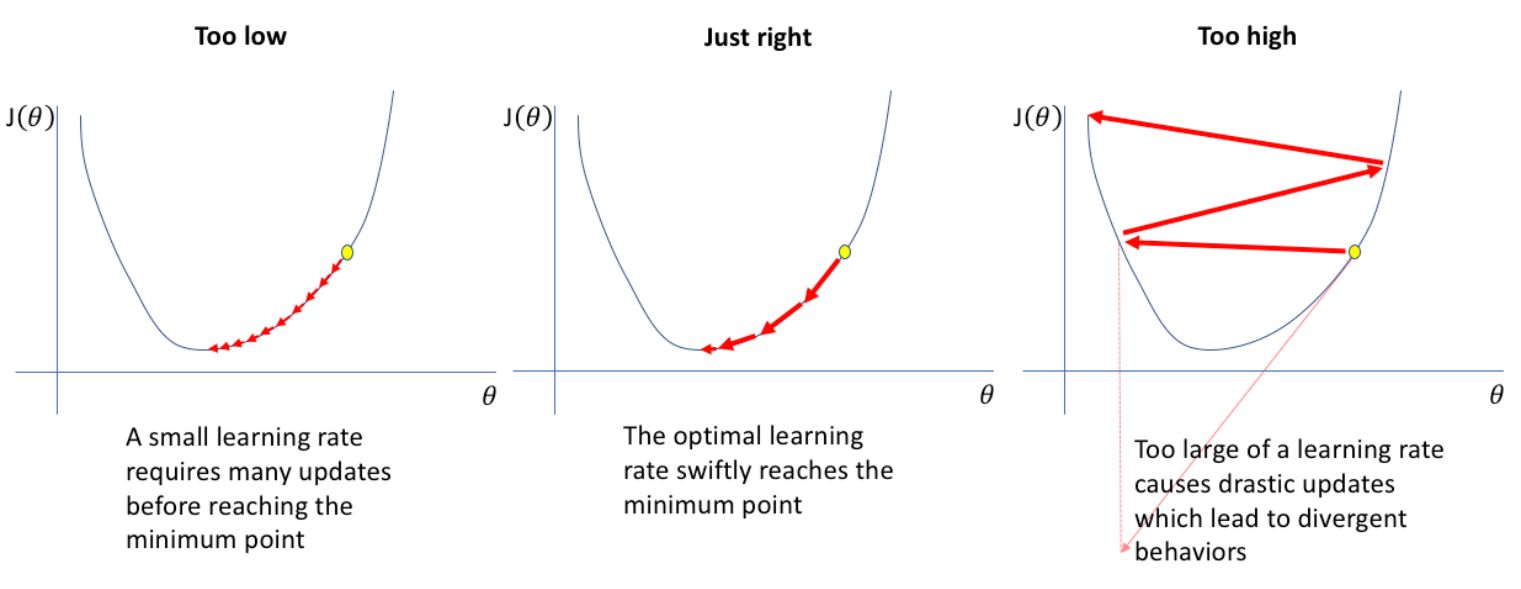
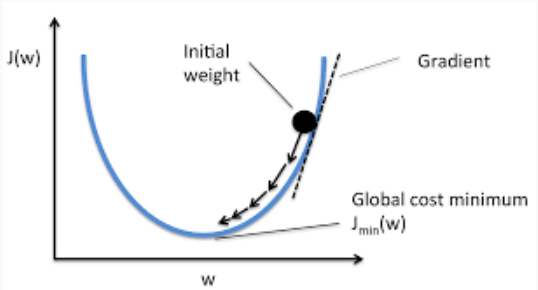


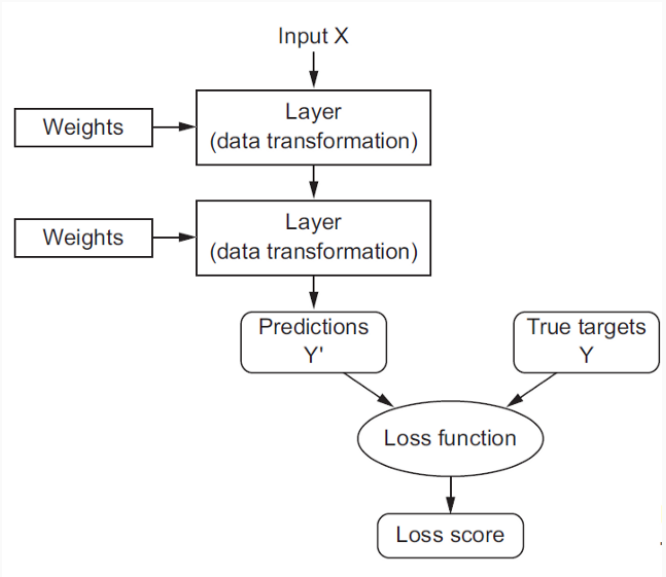
Figure 5: Different learning rates

Loss Function

- We convert the learning problem into an optimization problem. We define a loss function and then optimize the algorithm to minimize the loss function.
- A loss function measures the difference between the predicted output of a model and the actual output.



(a) Global cost



(b) Computing the Loss score

Common Types of Loss Functions

- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual values, commonly used in regression tasks.

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

- **Binary Cross-Entropy Loss (Log Loss):** Quantifies the difference between predicted class probabilities and true binary labels, used for binary classification.

$$\text{Log Loss} = -\sum_{i=1}^{n}\left(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right) \tag{2}$$

Common Types of Loss Functions

- **Categorical Cross-Entropy Loss:** Measures the difference between predicted class probabilities and true one-hot encoded labels, suitable for multi-class classification.

$$\text{Categorical Cross-Entropy Loss} = -\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij} \log(\hat{y}_{ij}) \tag{3}$$

- **Hinge Loss (SVM Loss):** Encourages a margin of separation between classes, often used in support vector machines (SVMs) and some classification problems.

$$\text{Hinge Loss} = \max(0, 1 - y\hat{y}) \tag{4}$$

Common Types of Loss Functions

- **Huber Loss:** A robust loss function that combines qualities of mean squared error and mean absolute error, used in regression tasks.

$$\text{Huber Loss} = \begin{cases} 0.5 \cdot (y - \hat{y})^2, & \text{for } |y - \hat{y}| \leq \delta \\ \delta \cdot |y - \hat{y}| - 0.5 \cdot \delta^2, & \text{otherwise} \end{cases} \tag{5}$$

- **Kullback-Leibler Divergence (KL Divergence):** Measures the difference between two probability distributions, often used in generative models.

$$\text{KL Divergence} = \sum_{i=1}^{n} y_i \log \left(\frac{y_i}{\hat{y}_i} \right) \tag{6}$$

Common Types of Loss Functions

- **Poisson Loss:** Applied to Poisson regression, which models count data, quantifying the difference between actual counts and predicted counts.

$$\text{Poisson Loss} = \hat{y} - y \log(\hat{y}) \tag{7}$$

- **Dice Loss:** Common in image segmentation, it measures the overlap between predicted and actual segmentations.

$$\text{Dice Loss} = 1 - \frac{2 \sum_{i=1}^{n} y_i \hat{y}_i}{\sum_{i=1}^{n} y_i + \sum_{i=1}^{n} \hat{y}_i} \tag{8}$$

Limitations

Perceptron Convergence Theorem

- If the training data used to train a perceptron is linearly separable, the inputs are bounded (i.e., they don't grow without limit), and the learning rate is finite, then the perceptron learning algorithm will converge and find a solution.
- It assures us that if the data is linearly separable, the perceptron algorithm will find a separating hyperplane.

Linear Separability

- Single-layer perceptrons can only solve linearly separable problems.

Example: XOR Problem

- XOR problem is a classic example of non-linearity.

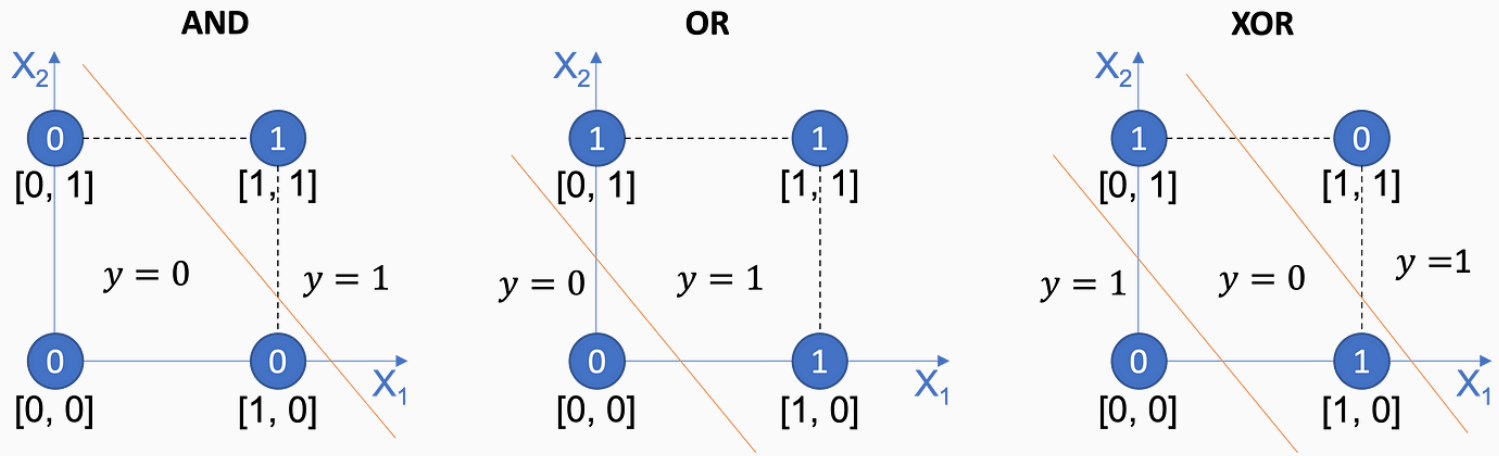


Figure 7: XOR is not linearly separable

Overcoming Limitations

Hidden Layers

- Hidden layers can model complex, hierarchical features.
- Use multiple layers: Multi-layer perceptrons (MLPs).
- MLPs with hidden layers can handle non-linear problems and achieve complex tasks.

Multi-Layer Perceptrons (MLPs)

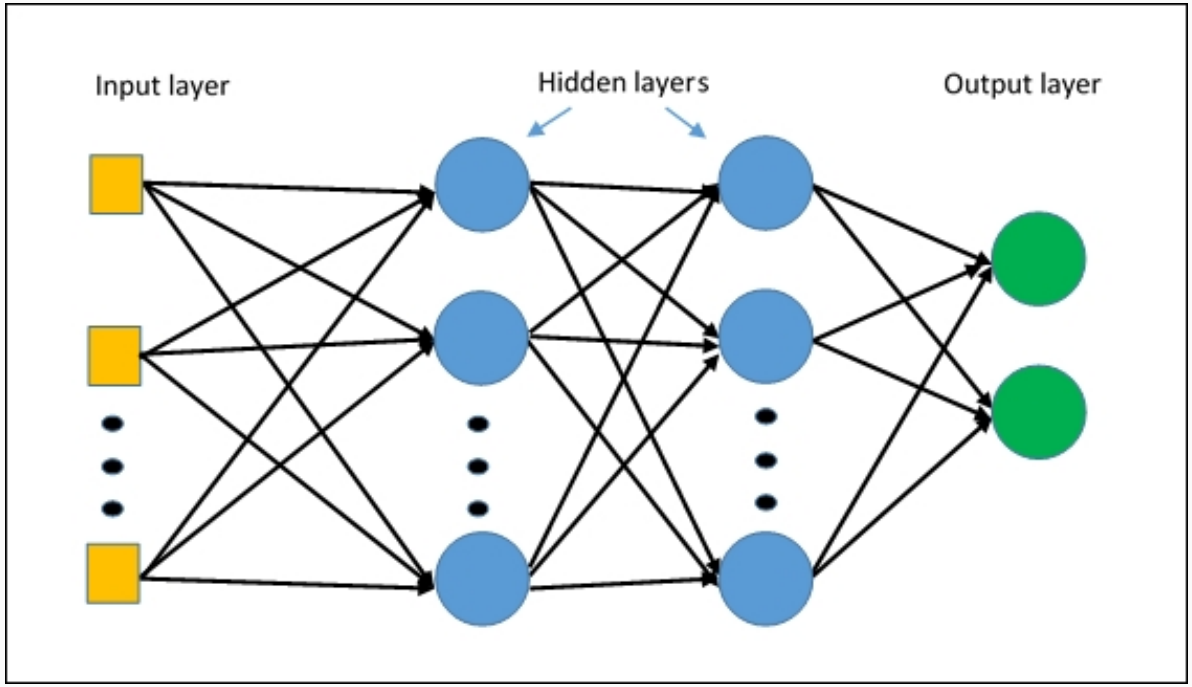


Figure 8: Multi-Layer Perceptron

Conclusion

Conclusion

- Perceptrons are powerful tools for binary classification.
- Trade-off between convergence speed and stability.
- MLPs with hidden layers can handle non-linear problems and achieve complex tasks.

Exercises

Exercise 1

Apply the perceptron learning algorithm for the following sample set until convergence. Use the samples in the given order cyclically. For each step of perceptron learning write down the applied sample, the classification result and the update of the weight vector.

- Sample 1 $[4, 3, 6]$ and sample 4 $[4, 2, 3]$ belong to class -1.
- Sample 2 $[2, -2, 3]$ and sample 3 $[1, 0, -3]$ belong to class 1.
- Learning rate, $\alpha = 0.5$
- Initial weights = 0; initial bias = 1
- Activation function:

$$f(z) = \begin{cases} -1 & \text{if } z < 0 \\ 1 & \text{if } z >= 0 \end{cases}$$

Exercise 1

Hint

Use delta rule to update the weights and the bias:

$\Delta\omega_i = \alpha \cdot (t_i - f(z_i)) \cdot x_i$;

$\Delta b = \alpha \cdot (t_i - f(z_i))$;

where α is the learning rate; t_i is the target; $f(z_i)$ is the output after the activation function

Exercise 2

- Plot the weights after each epoch
- Chose a Loss Function. Plot it after each epoch
- Explore different learning rates
- Explore different Activation functions