Neural Networks Class 1

Foundations & Perceptron

Learning Objectives:

- Understand biological motivation and historical context
- Master mathematical foundations of single neurons
- Learn perceptron mechanics and learning algorithm

From Biology to Mathematics

Biological vs Artificial Neurons

Biological Neuron:

- **Dendrites:** Receive signals from other neurons
- **Cell body:** Processes and integrates signals
- **Axon:** Transmits output when threshold reached
- **Synapses:** Control signal strength

From Biology to Mathematics

Biological vs Artificial Neurons

Artificial Neuron:

- **Inputs** $(x_1, x_2, \ldots, x_n)$: Represent dendrites
- **Weights** $(w_1, w_2, \ldots, w_n)$: Represent synaptic strengths
- **Weighted sum + bias:** Cell body processing
- **Activation function:** Axon firing threshold

Historical Context

The Journey from McCulloch-Pitts to Modern AI

1943: McCulloch-Pitts neuron

- First mathematical model of biological neuron
- Binary threshold activation

1957: Rosenblatt's Perceptron

- Learning algorithm introduced
- Pattern recognition applications

Historical Context

The Journey from McCulloch-Pitts to Modern AI

1969: Minsky & Papert critique

- Showed limitations (XOR problem)
- AI winter begins

1980s: Backpropagation revival

- Multi-layer networks solve limitations
- Modern deep learning foundation

Mathematical Foundations

Linear Algebra Refresher

Vectors: Column vectors represent data points

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Dot Product: Core operation in neural networks

$$\mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^{n} w_i x_i = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

Mathematical Foundations

Linear Algebra Refresher

Matrix-Vector Multiplication:

$$\mathbf{W}\mathbf{x} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_m^T \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{w}_1^T \mathbf{x} \\ \mathbf{w}_2^T \mathbf{x} \\ \vdots \\ \mathbf{w}_m^T \mathbf{x} \end{bmatrix}$$

The Perceptron Model

Mathematical Definition

Core Equation:

$$y = f(\mathbf{w} \cdot \mathbf{x} + b)$$

Where:

- $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$: Input vector
- $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$: Weight vector
- b: Bias term, $f(\cdot)$: Activation function, y: Output

Expanded Form:

$$y = f(w_1x_1 + w_2x_2 + \ldots + w_nx_n + b)$$

Activation Functions

Step Function

Mathematical Definition:

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

Properties:

- **Binary output:** Only 0 or 1
- **Non-differentiable:** Cannot use gradient-based learning
- **Historical importance:** Original perceptron activation

Geometric Interpretation:

Creates hard decision boundary at $\mathbf{w} \cdot \mathbf{x} + b = 0$

Activation Functions

Sigmoid Function

Mathematical Definition:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Properties:

- **Output range:** $(0, 1)$
- **Smooth and differentiable**
- **Derivative:** $\sigma'(z) = \sigma(z)(1 - \sigma(z))$

Advantages:

- Probabilistic interpretation

Activation Functions

Hyperbolic Tangent (tanh)

Mathematical Definition:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{2}{1 + e^{-2z}} - 1$$

Properties:

- **Output range:** $(-1, 1)$, **Zero-centered**, **Derivative:** $\tanh'(z) = 1 - \tanh^2(z)$

Relationship to Sigmoid:

$$\tanh(z) = 2\sigma(2z) - 1$$

Advantage: Zero-centered output helps with learning dynamics

Activation Functions

ReLU (Rectified Linear Unit)

Mathematical Definition:

$$\mathrm{ReLU}(z) = \max(0, z) = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

Properties:

- **Output range:** $[0, \infty)$, **Computationally efficient**
- **Derivative:** $\mathrm{ReLU}'(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$

Advantages:

- Solves vanishing gradient problem, Sparse activation (many zeros), Fast computation

Decision Boundaries

Geometric Interpretation

Linear Decision Boundary:

The perceptron creates a hyperplane: $\mathbf{w} \cdot \mathbf{x} + b = 0$

2D Case: Line equation $w_1 x_1 + w_2 x_2 + b = 0$

Classification Rule:

- **Class 1:** $\mathbf{w} \cdot \mathbf{x} + b > 0$
- **Class 0:** $\mathbf{w} \cdot \mathbf{x} + b < 0$

Weight Vector Properties:

- $\mathbf{w}$ is perpendicular to decision boundary
- $|\mathbf{w}|$ affects margin width, b shifts boundary away from origin

Perceptron Learning Algorithm

Goal: Find weights that correctly classify all training examples, **Learning rate:** $\eta > 0$ controls step size

Algorithm:

1. Initialize weights randomly: $\mathbf{w} = \mathbf{0}$ or small random values
2. For each training example $(\mathbf{x}_i, y_i)$:
   - Compute prediction: $\hat{y}_i = f(\mathbf{w} \cdot \mathbf{x}_i + b)$
   - If $\hat{y}_i \neq y_i$, update weights:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta(y_i - \hat{y}_i)\mathbf{x}_i$$

$$b \leftarrow b + \eta(y_i - \hat{y}_i)$$

3. Repeat until convergence or max iterations

Mathematical Derivation - Weight Update

Error Definition: $e_i = y_i - \hat{y}_i$

Update Rules:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta e_i \mathbf{x}_i$$

$$b^{(t+1)} = b^{(t)} + \eta e_i$$

Intuition:

- **Correct prediction** ($e_i = 0$): No update
- **False positive** ($e_i = -1$): Move boundary away from $\mathbf{x}_i$
- **False negative** ($e_i = +1$): Move boundary toward $\mathbf{x}_i$

Geometric Effect: Each update rotates the decision boundary to better classify the current example

Convergence Theorem

Theoretical Guarantees

Perceptron Convergence Theorem: If the training data is linearly separable, the perceptron learning algorithm will converge to a solution in finite steps.

Key Conditions:

1. **Linear separability:** $\exists \mathbf{w}^*, b^*$ such that all examples are correctly classified
2. **Finite margin:** Minimum distance between examples and decision boundary > 0

Convergence Theorem

Theoretical Guarantees

Convergence Bound:

$$\text{Number of mistakes} \leq \frac{R^2}{\gamma^2}$$

Where:

- R: Maximum distance of any example from origin
- γ: Margin (minimum distance to boundary)

Limitations of Single Perceptron

The XOR Problem Preview

XOR Truth Table:

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Problem: No single line can separate the XOR classes

Limitations of Single Perceptron

The XOR Problem Preview

Mathematical Proof: Assume linear separability

- $(0,0) \to 0: b < 0$
- $(0,1) \to 1: w_2 + b > 0$
- $(1,0) \to 1: w_1 + b > 0$
- $(1,1) \to 0: w_1 + w_2 + b < 0$

Contradiction: From conditions 2,3: $w_1, w_2 > -b > 0$

But condition 4 requires: $w_1 + w_2 < -b$

Summary: Class 1 Key Concepts

Biological Motivation:

- Neural networks inspired by biological neurons, Mathematical abstraction captures essential computation

Mathematical Foundations:

- Linear algebra: vectors, dot products, matrices, Perceptron model: $y = f(\mathbf{w} \cdot \mathbf{x} + b)$

Activation Functions:

- Step, sigmoid, tanh, ReLU with properties and trade-offs

Learning Algorithm:

- Perceptron learning rule with mathematical derivation, Convergence theorem for linearly separable data