Deep Learning for Perception

Lecture 01: Foundations of Deep Learning

Neural Networks & Perceptrons

Topics Covered in This Lecture:

- AI vs Machine Learning vs Deep Learning
- Supervised vs Unsupervised Learning
- Why Deep Learning?
- Biological Inspiration: Neurons
- The Perceptron Model
- Perceptron Learning Algorithm
- Common Activation Functions
  - Threshold (Step) Function
  - Sigmoid Function
  - Hyperbolic Tangent (tanh)
  - Rectified Linear Unit (ReLU)

Instructor: Aqsa Younas

Lecturer, Department of Computer Science
FAST-NU, CFD Campus

Contents

Advance Organizer — What You'll Learn

Learning Objectives: By the end of this lecture, you will be able to:

1. **Distinguish** between AI, Machine Learning, and Deep Learning
2. **Compare** supervised and unsupervised learning paradigms
3. **Explain** why deep learning has become so powerful
4. **Describe** the biological inspiration behind artificial neurons
5. **Implement** the perceptron learning algorithm
6. **Apply** different activation functions and understand their properties

Prior Knowledge Activation: Before we begin, think about these questions:

- Have you used any AI-powered applications? (Siri, Google Translate, Netflix recommendations?)
- What do you think makes computers "intelligent"?
- How do you think the human brain processes information?

1 AI vs Machine Learning vs Deep Learning

Why It Matters

Understanding the relationship between AI, ML, and DL is fundamental because these terms are often confused. Each represents a different level of sophistication in making machines "intelligent." This knowledge helps you choose the right approach for different problems.

1.1 The Nested Hierarchy

Definition

Artificial Intelligence (AI) is the broadest concept—any technique that enables machines to **mimic human behavior** and intelligence, including rule-based systems, expert systems, and learning algorithms.

Definition

Machine Learning (ML) is a **subset of AI** that uses statistical methods to enable machines to **improve with experience**. Instead of explicit programming, ML algorithms learn patterns from data.

Definition

Deep Learning (DL) is a **subset of ML** that uses **multi-layer neural networks** to learn hierarchical representations of data. It excels at processing images, text, and sound.

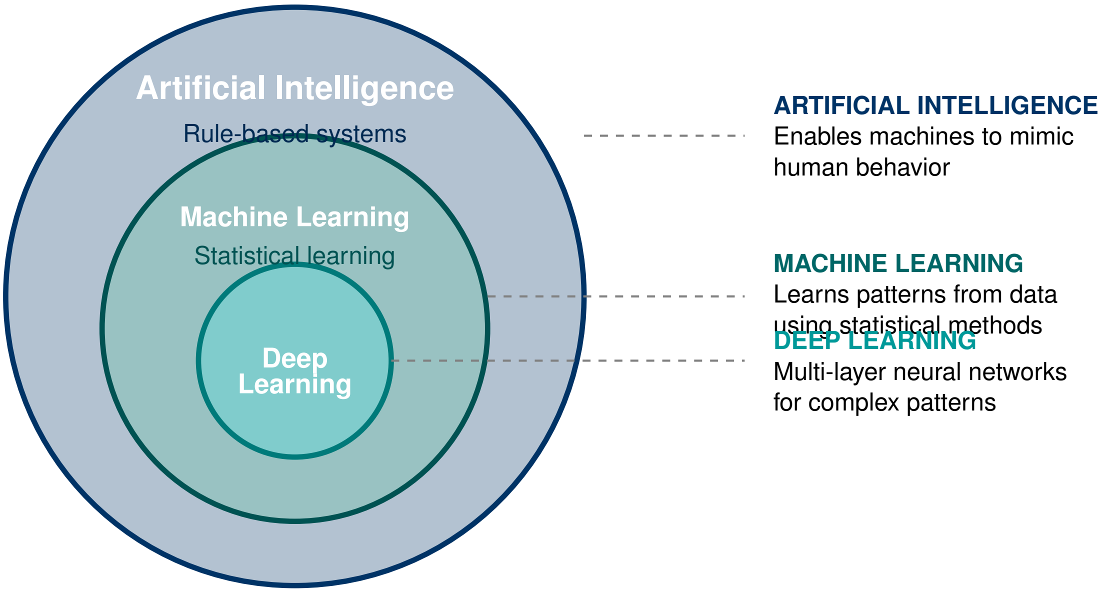

Figure 1: The Nested Relationship: AI $\supset$ ML $\supset$ DL

Figure Explanation: This diagram shows how Deep Learning is the innermost circle—it's the most specialized. Every DL system is also ML, and every ML system is also AI. But not all AI uses ML (some use rules), and not all ML uses deep neural networks.

1.2 ML vs DL: Key Differences

Key Comparison

Aspect	**Machine Learning**	**Deep Learning**
Feature Engineering	Manual—humans select features	Automatic—network learns features
Data Requirements	Works with **small data**	Requires **large data** (thousands+)
Training Time	**Less time** to train	**More time** to train
Testing Time	**More time** for inference	**Less time** for inference
Hardware	Can run on CPU	Often requires **GPU/TPU**
Interpretability	Often interpretable	Often "black box"

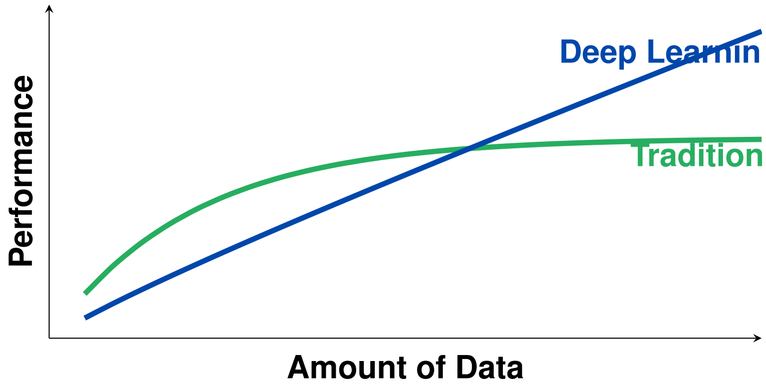

Figure 2: Performance vs Amount of Data: Deep Learning scales better with more data

Key Insight: Traditional ML algorithms (like decision trees, SVM) reach a performance plateau—adding more data doesn't help much. Deep Learning **continues to improve** as you feed it more data, which is why it dominates in the era of big data.

Memory Hook — Remember This!

Quick Memory Aid:

- **ML** = "Learn with experience, no spoon-feeding" — needs **small data**
- **DL** = "Inspired by brain structure" — needs **huge data**
- **Training**: ML is fast, DL is slow
- **Testing**: ML is slow, DL is fast

Self-Test — Check Your Understanding

Quick Check:

1. Is every Deep Learning system also a Machine Learning system? *(Yes—DL is a subset of ML)*
2. You have only 100 training samples. Should you use DL or traditional ML? *(Traditional ML—DL needs more data)*
3. Which requires more training time: ML or DL? *(DL)*

2 Supervised vs Unsupervised Learning

Why It Matters

The type of learning paradigm determines what kind of data you need and what problems you can solve. Supervised learning needs expensive labeled data but can make precise predictions. Unsupervised learning works with raw data but discovers hidden patterns.

2.1 Supervised Learning

Definition

Supervised Learning uses **labeled data** [data with known correct answers] to train models. The algorithm learns to map inputs to outputs by studying many input-output pairs.
Analogy: Like a student learning with a teacher who provides correct answers for practice problems.

Analogy — Think of It Like This

Imagine teaching a child to recognize animals. You show pictures and say "This is a cat," "This is a dog." The child learns from these **labeled examples**. Later, when shown a new picture, the child can identify the animal—this is supervised learning!

Types of Supervised Learning Tasks:

- **Classification** [predict discrete categories]: Spam detection, disease diagnosis, image recognition
- **Regression** [predict continuous values]: House price prediction, temperature forecasting, stock prices

2.2 Unsupervised Learning

Definition

Unsupervised Learning works with **unlabeled data**—no correct answers are provided. The algorithm must discover patterns, structures, or relationships on its own.
Analogy: Like a child sorting toys without being told how—they might group by color, size, or type naturally.

Types of Unsupervised Learning Tasks:

- **Clustering** [group similar items]: Customer segmentation, document grouping
- **Anomaly Detection** [find outliers]: Fraud detection, network intrusion
- **Dimensionality Reduction** [compress data]: Data visualization, feature extraction

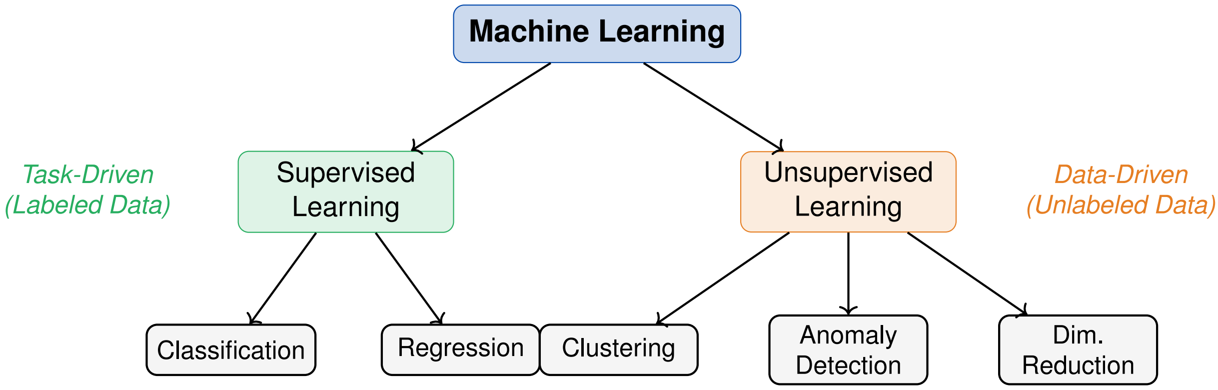

Figure 3: Machine Learning Taxonomy

Key Comparison

Aspect	Supervised Learning	Unsupervised Learning
Data Type	Labeled (input + correct output)	Unlabeled (input only)
Goal	Predict known outcomes	Discover hidden patterns
Feedback	Has “teacher” providing answers	No teacher, self-discovers
Examples	Classification, Regression	Clustering, Anomaly Detection
Real-World	Spam filter, Medical diagnosis	Customer segmentation, Fraud detection

3 Why Deep Learning?

Why It Matters

Deep Learning has revolutionized AI because it solves problems that were previously impossible. Understanding **why** DL works helps you know when to use it and when simpler methods suffice.

3.1 The Power of Deep Learning

Definition

Deep Learning is a machine learning technique that learns **features and tasks directly from data** (images, text, sound) using neural networks with **multiple layers** (hence "deep").

Key Advantages of Deep Learning:

1. **Automatic Feature Learning:** Traditional ML requires humans to design features. DL **learns features automatically** from raw data.
2. **Scales with Data:** Performance keeps improving as you add more training data.
3. **Handles Complex Patterns:** Can learn hierarchical representations—simple features combine into complex ones.
4. **Transfer Learning:** Pre-trained models can be adapted to new tasks with less data.

3.2 Real-World Applications

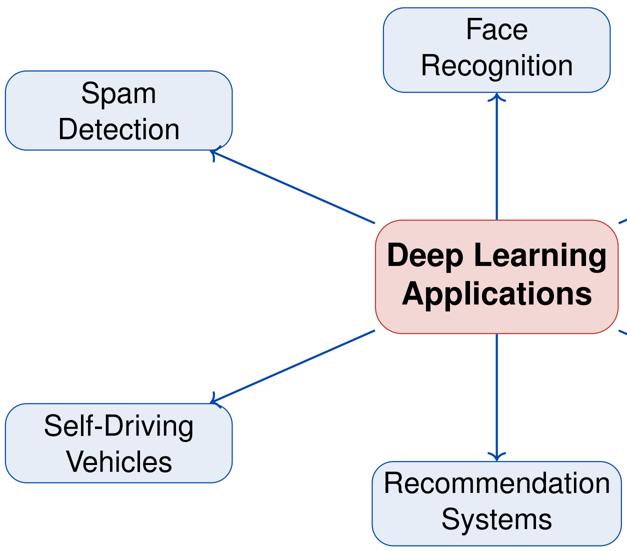

Figure 4: Real-World Deep Learning Applications

Memory Hook — Remember This!

Remember: Deep Learning excels when you have:

- **Lots of data** (thousands to millions of examples)
- **Complex patterns** (images, speech, natural language)
- **Computational resources** (GPUs/TPUs available)

Use traditional ML when data is limited or interpretability is critical.

4 Biological Inspiration: The Neuron

Why It Matters

Understanding biological neurons helps you intuit how artificial neurons work. The brain's architecture—billions of simple units working together—inspired the design of neural networks.

4.1 The Biological Neuron

Definition

A **biological neuron** is a nerve cell that:

- **Receives** signals from other neurons via **dendrites**
- **Processes** signals in the **cell body** (soma)
- **Transmits** signals to other neurons via the **axon**
- **Fires** (sends output signal) when input exceeds a **threshold**

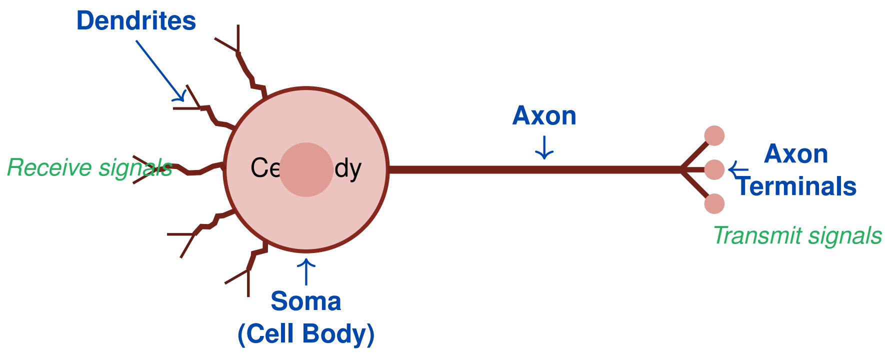

Figure 5: Structure of a Biological Neuron

How It Works:

1. **Dendrites** receive electrical signals from other neurons
2. Signals are **summed** in the cell body
3. If the sum exceeds a **threshold**, the neuron "fires"
4. The **axon** transmits the output signal to other neurons

This is an "all-or-none" process—the neuron either fires or doesn't.

Memory Hook — Remember This!

Neuron Firing Rule:

"When input signals exceed a threshold within a short time, the neuron FIRES—it's all-or-nothing!"

5 The Perceptron

Why It Matters

The perceptron is the **building block** of all neural networks. Understanding it deeply gives you the foundation for understanding complex deep learning architectures.

5.1 What is a Perceptron?

Definition

A **Perceptron** is the simplest artificial neural network, introduced by **Frank Rosenblatt in 1958**. It's a **binary linear classifier** that:

- Takes multiple inputs
- Computes a **weighted sum**
- Applies an **activation function**
- Produces a single output (0 or 1)

Analogy — Think of It Like This

A Perceptron is like a Judge:
Imagine a judge deciding whether to approve a loan. The judge considers multiple factors (income, credit score, employment) but gives different **weight** (importance) to each. If the weighted evidence exceeds a threshold, the loan is approved (output = 1); otherwise, it's rejected (output = 0).

5.2 Perceptron Architecture

Components of a Perceptron:

- **Inputs** $(x_1, x_2, \ldots, x_n)$: Feature values from your data
- **Weights** $(w_1, w_2, \ldots, w_n)$: Learnable parameters showing importance
- **Bias** (b): Shifts the decision boundary (allows flexibility)
- **Net Input**: Weighted sum $z = \sum_{i=1}^{n} w_i x_i + b$
- **Activation Function**: Converts net input to output

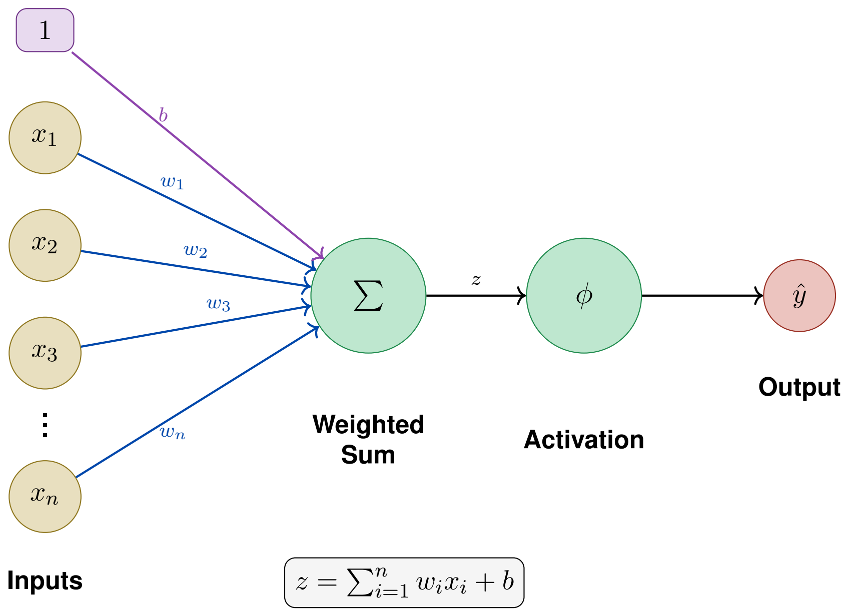

Figure 6: Perceptron Architecture: Inputs → Weighted Sum → Activation → Output

5.3 Mathematical Formulation

The perceptron computes:

$$\boxed{z = \sum_{i=1}^{n} w_i x_i + b = \mathbf{w} \cdot \mathbf{x} + b} \tag{1}$$

Then applies an activation function ϕ:

$$\boxed{\hat{y} = \phi(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}} \tag{2}$$

Memory Hook — Remember This!

Why do we need bias?
Without bias, the decision boundary **must pass through the origin**. The bias allows the boundary to **shift anywhere**, giving the model more flexibility to fit the data.

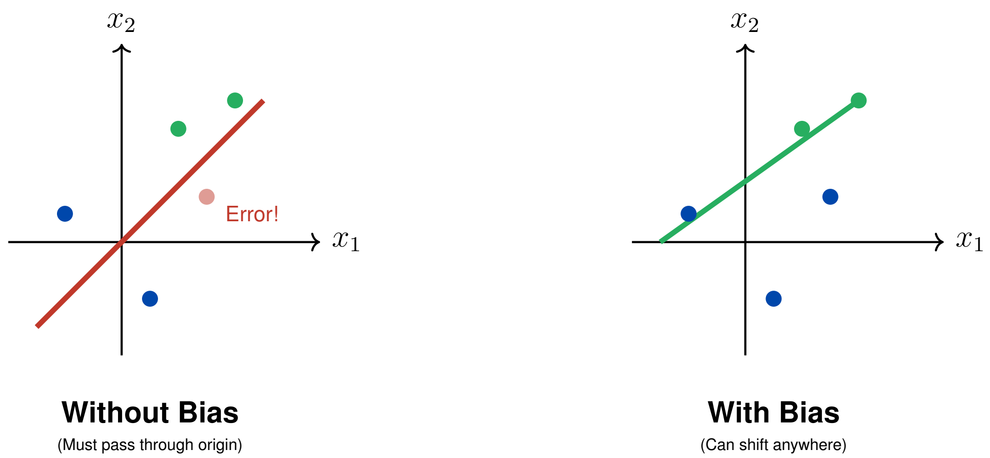

Figure 7: The Importance of Bias: Allows the decision boundary to shift

6 Perceptron Learning Algorithm

Why It Matters

The learning algorithm is **how** the perceptron finds the right weights. Understanding this iterative process is fundamental to understanding how all neural networks learn.

6.1 The Learning Process

Definition

The **Perceptron Learning Algorithm** iteratively adjusts weights and bias based on classification errors. If the output is wrong, it updates the parameters to reduce the error.

Algorithm

Perceptron Learning Algorithm

1. **Initialize:** Set all weights $w_i = 0$ (or small random values) and bias $b = 0$
2. **For each training sample** $(\mathbf{x}, y)$:
   a) **Compute output:**

   $$\hat{y} = \text{step}(\mathbf{w} \cdot \mathbf{x} + b) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

   b) **Update if misclassified:**

   $$\boxed{w_i \leftarrow w_i + \eta(y - \hat{y})x_i}$$

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

   where η is the **learning rate** [controls step size]
3. **Repeat until:**
   - All samples are correctly classified, **OR**
   - Maximum iterations reached

6.2 Understanding the Update Rule

The term $(y - \hat{y})$ can only be:

- **0**: Correct prediction $\rightarrow$ No update needed
- **+1**: Predicted 0, actual 1 $\rightarrow$ Increase weights (to make output larger)
- **-1**: Predicted 1, actual 0 $\rightarrow$ Decrease weights (to make output smaller)

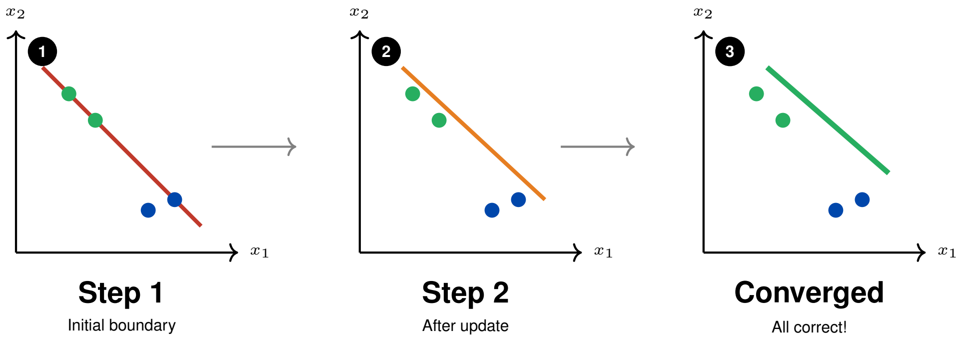

Figure 8: Perceptron Learning: Decision boundary adjusts iteratively until all points are classified correctly

Memory Hook — Remember This!

Learning Rate (η):

- **Too large**: Overshoots optimal solution, may never converge
- **Too small**: Takes too long to converge
- **Typical values**: 0.01 to 0.1

Self-Test — Check Your Understanding

Troubleshooting:

- **Model not learning?** Check learning rate—try smaller or larger values
- **Never converges?** Data may not be linearly separable—perceptron can only learn linear boundaries!

7 Common Activation Functions

Why It Matters

Activation functions introduce **non-linearity**, enabling neural networks to learn complex patterns. Without them, a neural network would just be a linear transformation, no matter how many layers you add.

Definition

An **Activation Function** decides whether a neuron should be "activated" (fired) based on the weighted sum of inputs. It performs a **non-linear transformation** on the input signal.

Analogy — Think of It Like This

Think of an activation function as a **volume knob** with special properties. It takes the raw signal and transforms it—some functions squash everything between 0 and 1, others cut off negative values entirely.

7.1 Why Non-Linearity?

Memory Hook — Remember This!

Why do we need non-linear activation functions?
Real-world data is rarely a straight line! Without non-linearity:

- Multiple layers would collapse into a single linear transformation
- Network could only learn linear decision boundaries
- Complex patterns (images, speech) would be impossible to learn

7.2 Threshold (Step) Function

Definition

The **Threshold Function** (or Step Function) outputs 1 if input is positive, 0 otherwise. Used in the **original perceptron**.

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

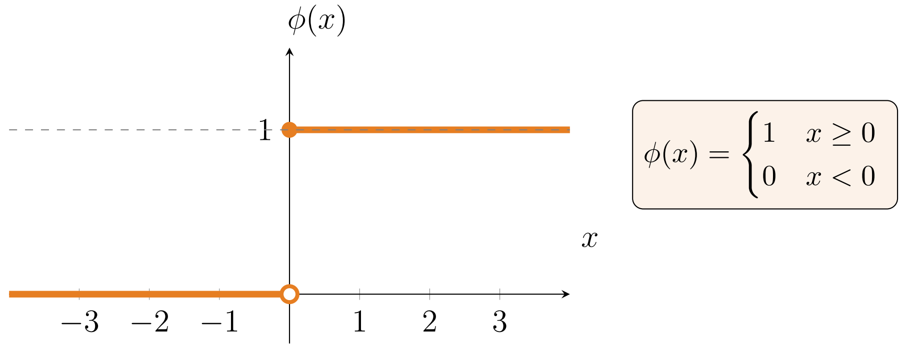

Figure 9: Threshold (Step) Function

Characteristics:

- **Output range**: $\{0, 1\}$ (binary)
- **Pros**: Simple, intuitive
- **Cons**: Not differentiable at 0 (can't use gradient descent)

7.3 Sigmoid Function

Definition

The **Sigmoid Function** squashes any input to a value between 0 and 1, creating an S-shaped curve.

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

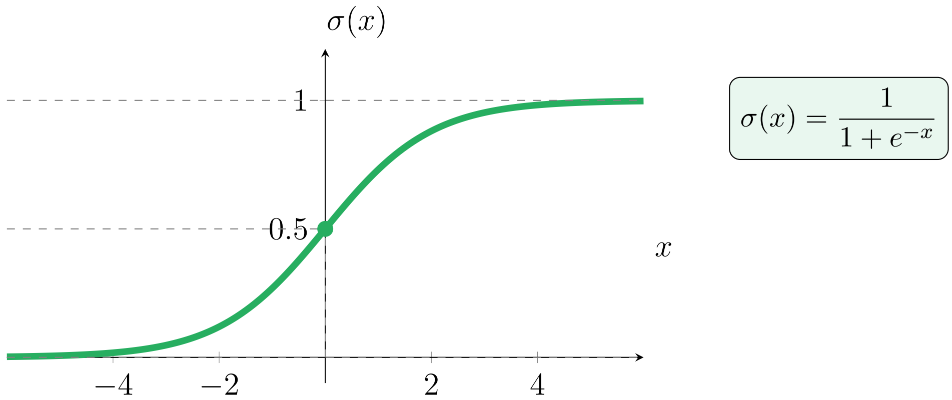

Figure 10: Sigmoid Function — S-shaped curve mapping to (0, 1)

Characteristics:

- **Output range**: $(0, 1)$
- **Pros**: Smooth, differentiable, outputs like probabilities
- **Cons**: **Vanishing gradient** problem—gradients become very small for large/small inputs
- **Use case**: Output layer for binary classification

7.4 Hyperbolic Tangent (tanh)

Definition

The **Hyperbolic Tangent** (tanh) is similar to sigmoid but outputs values between -1 and 1.

$$\boxed{\phi(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}}}$$

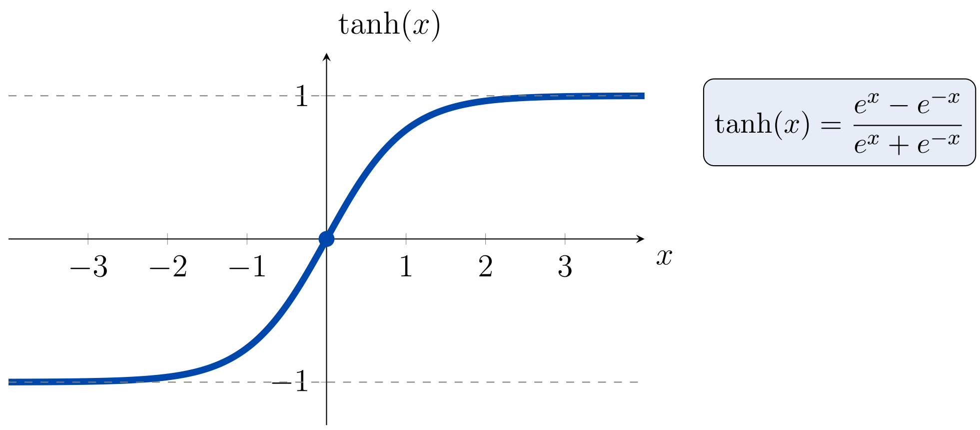

Figure 11: Hyperbolic Tangent (tanh) — Zero-centered output

Characteristics:

- **Output range**: $(-1, 1)$
- **Pros**: **Zero-centered**—easier for optimization than sigmoid
- **Cons**: Still has vanishing gradient problem
- **Use case**: Hidden layers (better than sigmoid)

7.5 Rectified Linear Unit (ReLU)

Definition

ReLU (Rectified Linear Unit) is the **most widely used** activation function today. It outputs the input if positive, otherwise outputs 0.

$$\boxed{\phi(x) = \max(0, x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}}$$

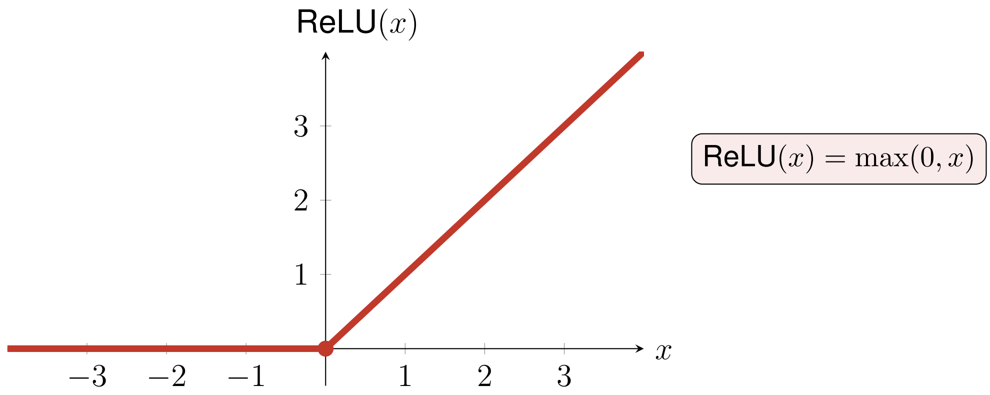

Figure 12: ReLU — The most popular activation function

Characteristics:

- **Output range**: $[0, \infty)$
- **Pros**:
  - Computationally efficient (just comparison)
  - **No vanishing gradient** for positive values
  - Enables sparse activation (many zeros)
- **Cons**: “Dying ReLU” problem—neurons can get stuck outputting 0
- **Use case**: Default choice for hidden layers in modern networks

7.6 Comparison of Activation Functions

Key Comparison

Function	Range	Formula	Best For
Step	$\{0, 1\}$	$\begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$	Original perceptron (historical)
Sigmoid	$(0, 1)$	$\frac{1}{1+e^{-x}}$	Binary classification output
Tanh	$(-1, 1)$	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	Hidden layers (zero-centered)
ReLU	$[0, \infty)$	$\max(0, x)$	Hidden layers (default choice)

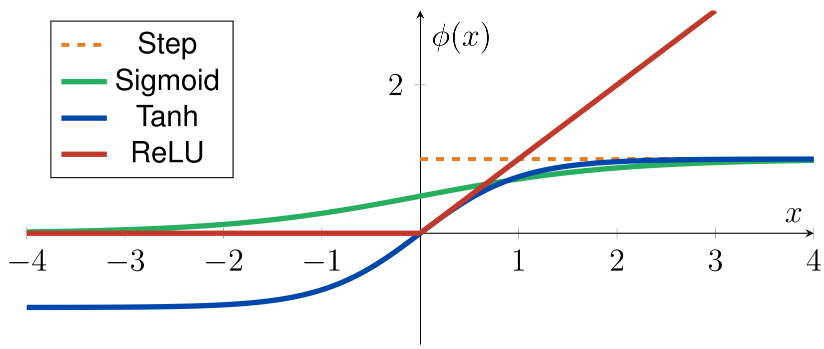

Figure 13: Comparison of All Activation Functions

Memory Hook — Remember This!

Quick Decision Guide:

- **Hidden layers**: Use **ReLU** (default) or tanh
- **Binary classification output**: Use **Sigmoid**
- **Multi-class classification output**: Use **Softmax** (not covered here)
- **Regression output**: Use **Linear** (no activation)

8 Summary and Key Takeaways

Lecture Summary — Key Points to Remember

1. **AI $\supset$ ML $\supset$ DL**: Deep Learning is a subset of Machine Learning, which is a subset of AI
2. **Supervised vs Unsupervised**: Supervised uses labeled data for prediction; Unsupervised finds patterns in unlabeled data
3. **Why Deep Learning**: Automatic feature learning, scales with data, handles complex patterns
4. **Perceptron**: Simplest neural network—weighted sum + activation function
5. **Perceptron Learning**: Update weights when misclassified: $w \leftarrow w + \eta(y - \hat{y})x$
6. **Activation Functions**:
   - Step: $\{0, 1\}$—binary, not differentiable
   - Sigmoid: $(0, 1)$—probabilities, vanishing gradient
   - Tanh: $(-1, 1)$—zero-centered, vanishing gradient
   - ReLU: $[0, \infty)$—most popular, efficient, no vanishing gradient

Self-Test — Check Your Understanding

Final Self-Assessment Questions:

1. What is the main difference between supervised and unsupervised learning?
   Answer: Supervised uses labeled data; unsupervised uses unlabeled data
2. Why does deep learning need more data than traditional ML?
   Answer: More parameters to learn, needs data to learn automatic features
3. What is the role of bias in a perceptron?
   Answer: Allows decision boundary to shift away from origin
4. When is a perceptron's weight updated?
   Answer: Only when it makes a misclassification
5. Why is ReLU preferred over sigmoid in hidden layers?
   Answer: No vanishing gradient problem, computationally efficient
6. What is the output range of sigmoid? Of tanh?
   Answer: Sigmoid: (0,1); Tanh: (-1,1)

9 Glossary of Key Terms

Term	Definition
Artificial Intelligence	Technology enabling machines to mimic human behavior
Machine Learning	Algorithms that learn patterns from data without explicit programming
Deep Learning	ML using multi-layer neural networks
Supervised Learning	Learning from labeled data with known outputs
Unsupervised Learning	Learning patterns from unlabeled data
Perceptron	Simplest neural network—single layer, binary classifier
Weights	Learnable parameters indicating input importance
Bias	Parameter allowing decision boundary to shift
Activation Function	Non-linear function transforming neuron output
Learning Rate (η)	Controls how much weights update each step
Epoch	One complete pass through all training data
Vanishing Gradient	Problem where gradients become too small to learn