上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Lecture 14: Perceptron & MLP

Tao Huang

John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

https://taohuang.info/cs3317

https://oc.sjtu.edu.cn/courses/89538

AI tools assisted in generating some figures in these slides. All such content has been reviewed, and the instructor is responsible for its accuracy.

Objectives

By the end of this lecture, you should be able to:

- explain why a **single perceptron** is only a **linear classifier**
- explain how an **MLP** uses **hidden layers + activations** to model **nonlinear patterns**
- explain what the **universal approximation theorem** says, and why **depth still matters** in practice

1. Perceptron: Where Linearity Hits a Wall

Why Deep Learning?

We already have Logistic Regression, SVM, Trees/Forests/Boosting.

Three answers:

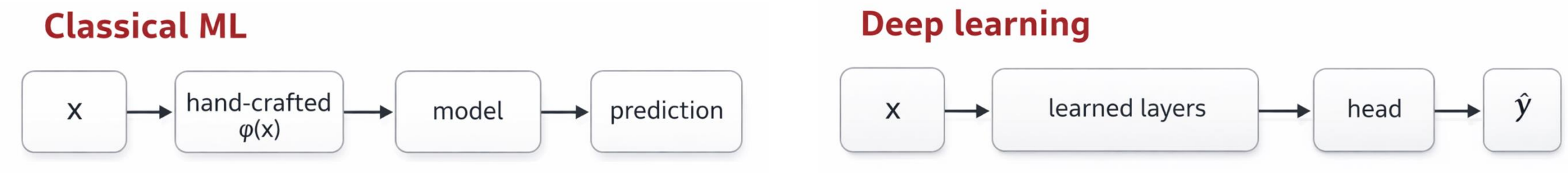
- More expressive capacity
- Automatic feature learning
- Scalability to massive data

From Classical ML to Representation Learning

- Classical approach: Map features to predictions.
- The nonlinear bottleneck: Relied on manual feature engineering $\phi(x)$ or kernels.

Fundamental question: Who designs the representation?

Deep learning paradigm: Learn representation directly from data.



Perceptron = Linear Classifier

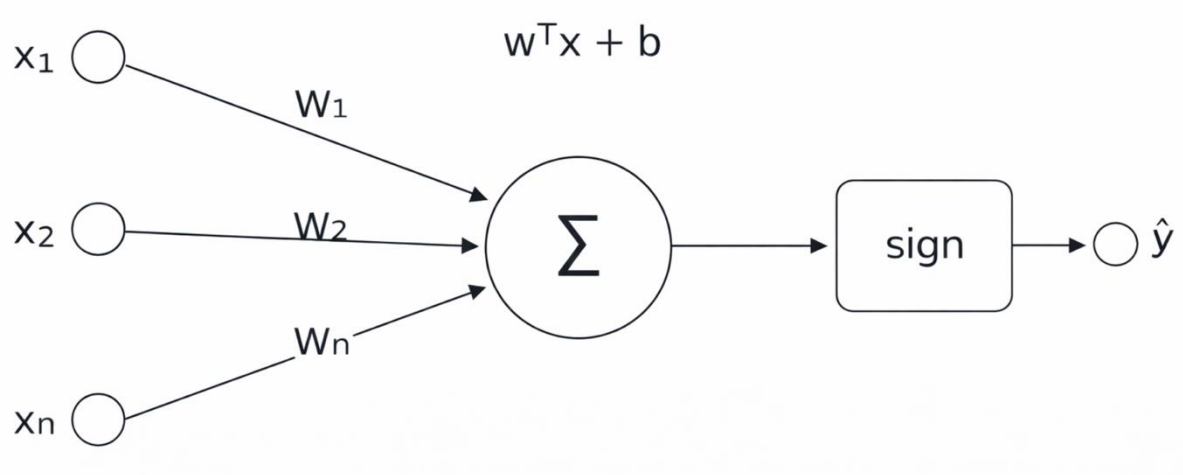
Computation:

$$\hat{y} = \text{sign}(w^T x + b)$$

Realization:

- Not a new model class
- Just linear classification in "neural" language

One neuron = One hyperplane



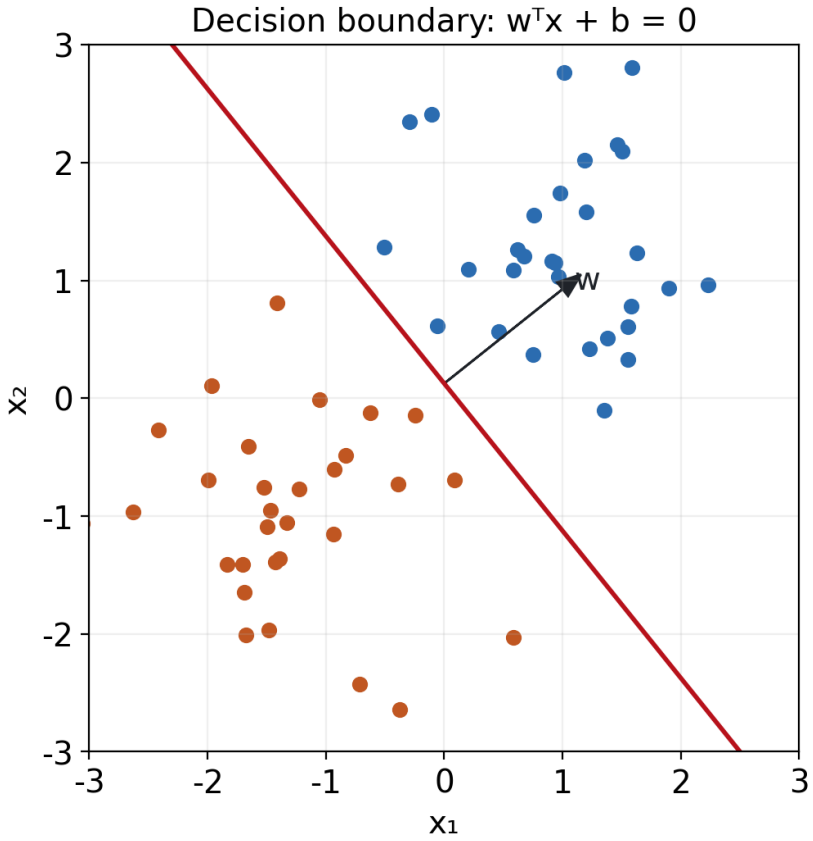
What Does a Perceptron Really Learn?

- Decision boundary:

$$w^T x + b = 0$$

- Geometry:
  - Strictly linear separator
  - Divides space into two half-spaces
- vs. Logistic Regression / SVM:
  - Different objective / optimization
  - Different raw output

Expressive power: unchanged.



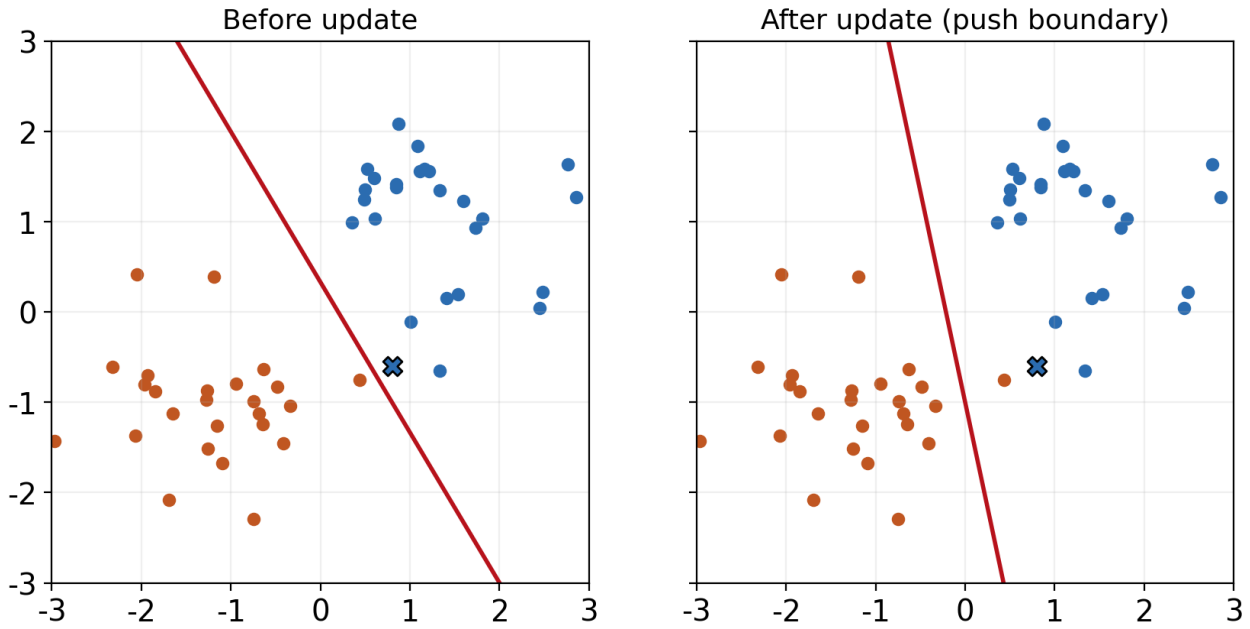
Perceptron Learning: Fixing Mistakes Online

Update rule for misclassified sample (x_i, y_i):

$$w \leftarrow w + \eta y_i x_i, \qquad b \leftarrow b + \eta y_i$$

Geometric intuition: Boundary shifts to correct error.

Optimization perspective: Precursor to SGD.



Checkpoint

Gained:

- Trainable artificial neuron
- Simple online update rule
- Historical bridge

Still Missing:

- Nonlinear boundaries
- Hierarchical features
- Expressive representations

Historically neural, functionally still linear.

2. XOR: the Crisis that Forces Hidden Layers

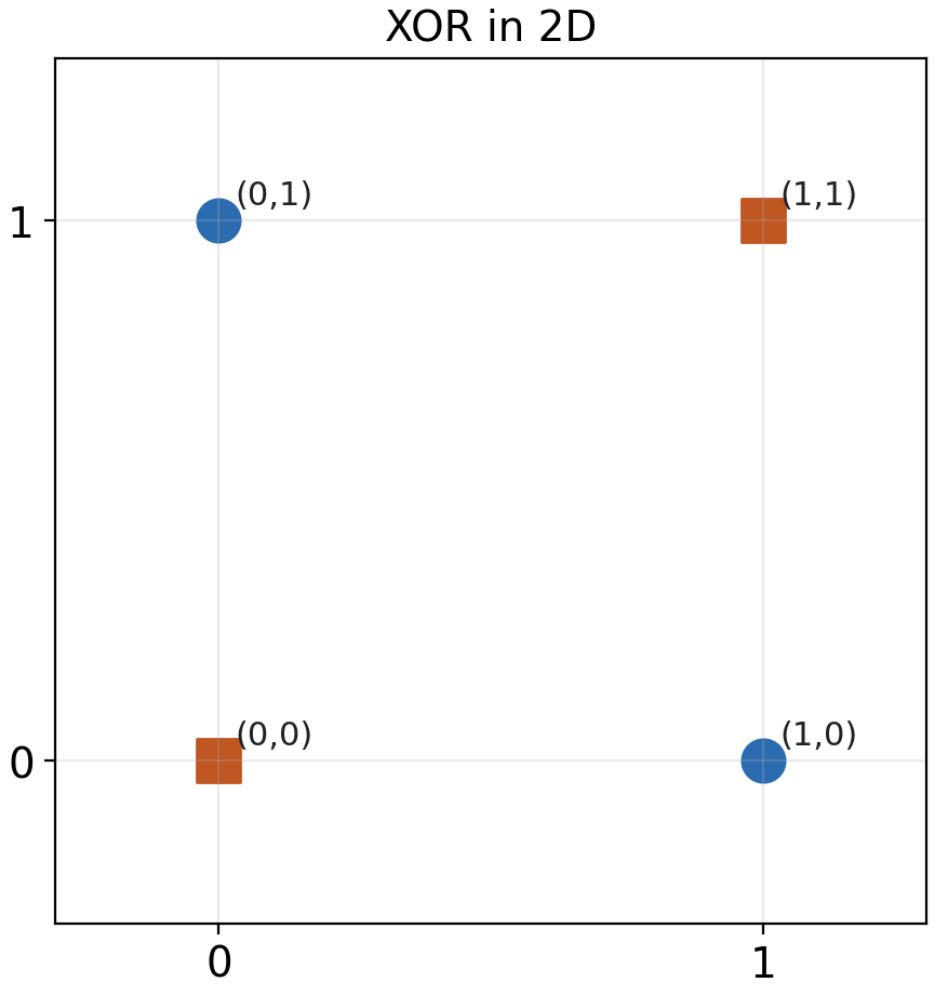
Can One Line Solve XOR?

30-Second Prompt:

Can a single line separate the + and - points?

Do not just answer “no”.

Explain why geometrically.



XOR: A Structural Failure Case

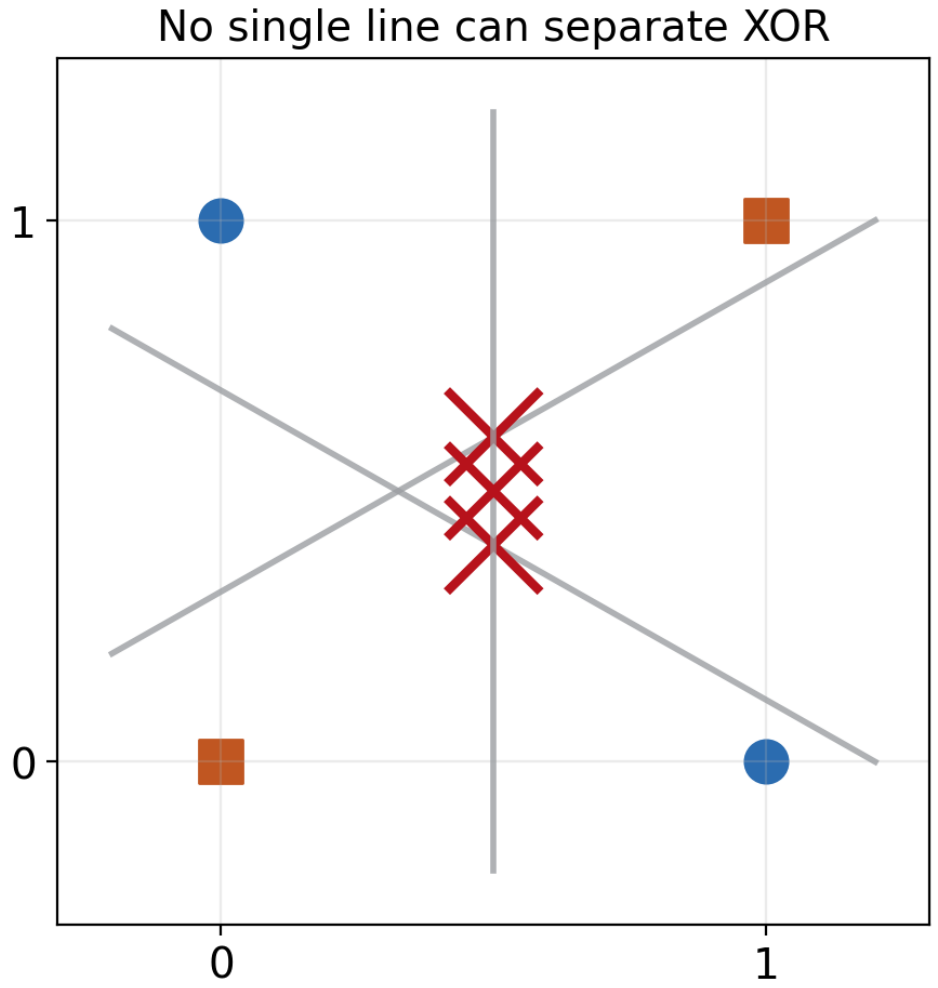
Stronger than “training failed”.

For XOR:

No choice of w and b exists.

Distinction:

- Optimization failure
- Expressivity failure (Hypothesis class limit)



Analyze the Bottleneck

Classify the main bottleneck in each case:

Scenarios

1. Perceptron on linearly separable data
2. Perceptron on XOR
3. Deep network with poor initialization

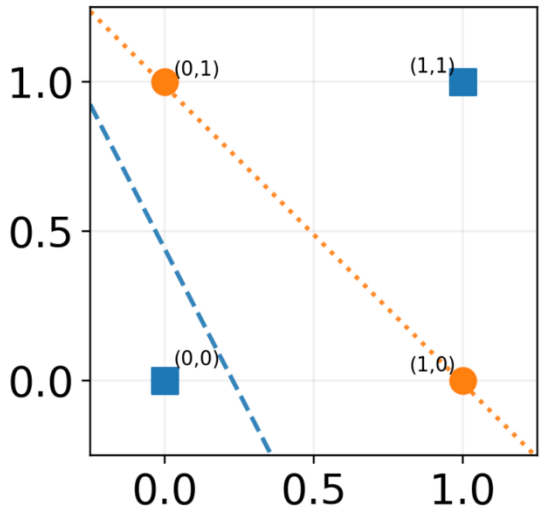
Possible Bottlenecks

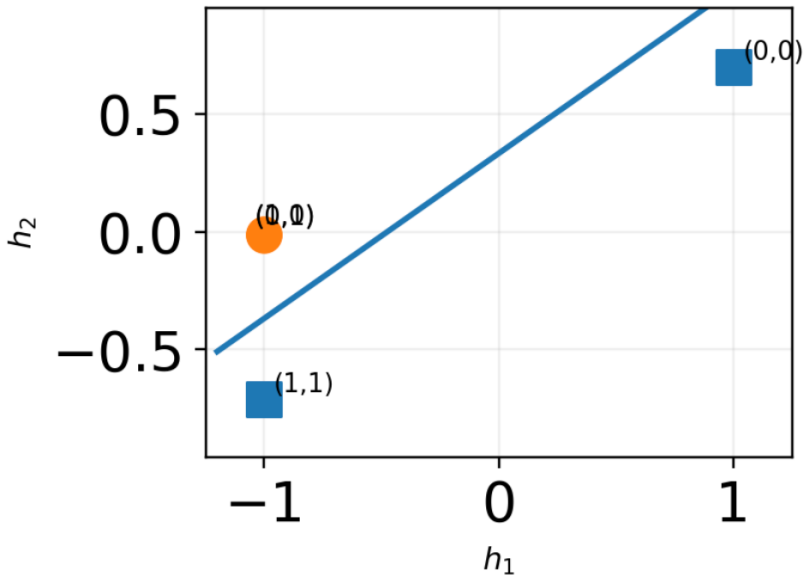
- Expressivity
- Optimization
- Data
- Generalization

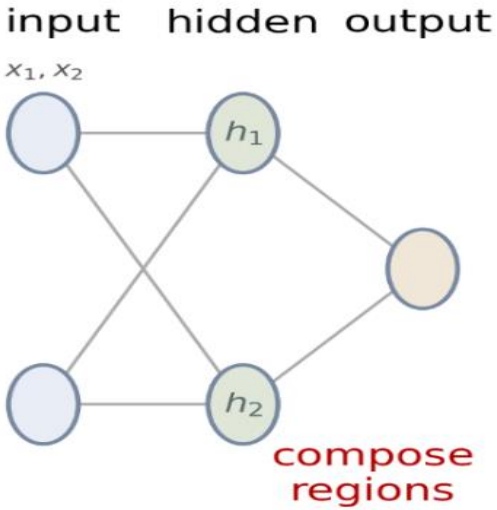
Escape XOR: Composition

- One neuron = one linear cut.
- Several neurons = several cuts.
- Later neurons combine learned tests.

Breakthrough: Composition of intermediate features.







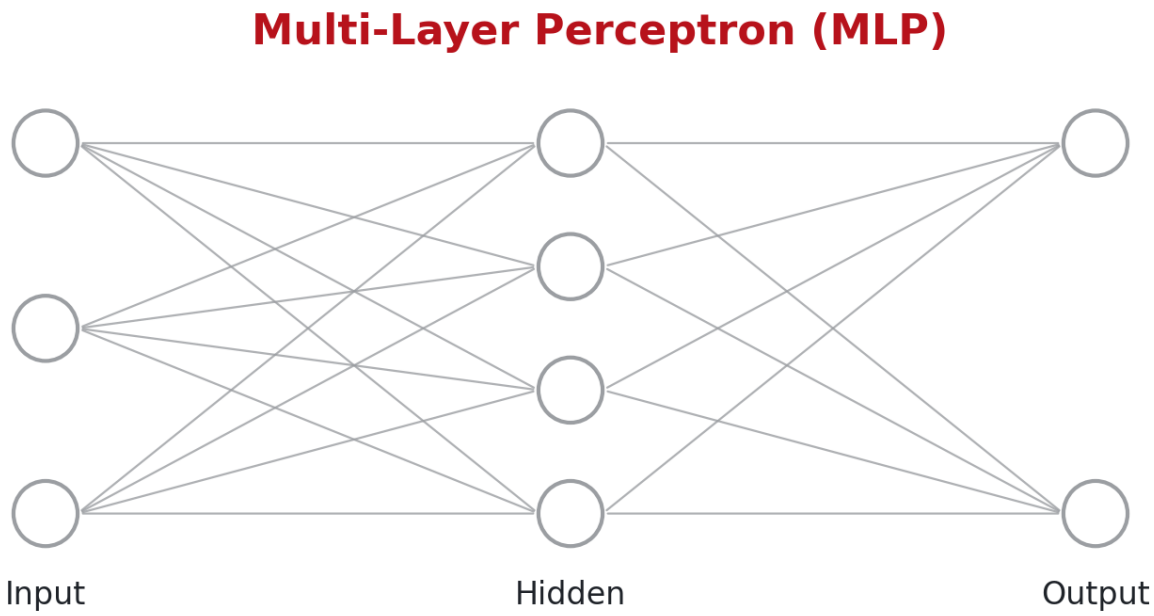
Multi-Layer Perceptron (MLP)

Input: $a^{(0)} = x$

For layer l:

- $z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$
- $a^{(l)} = \phi(z^{(l)})$

Hidden layers = learned intermediate representations.



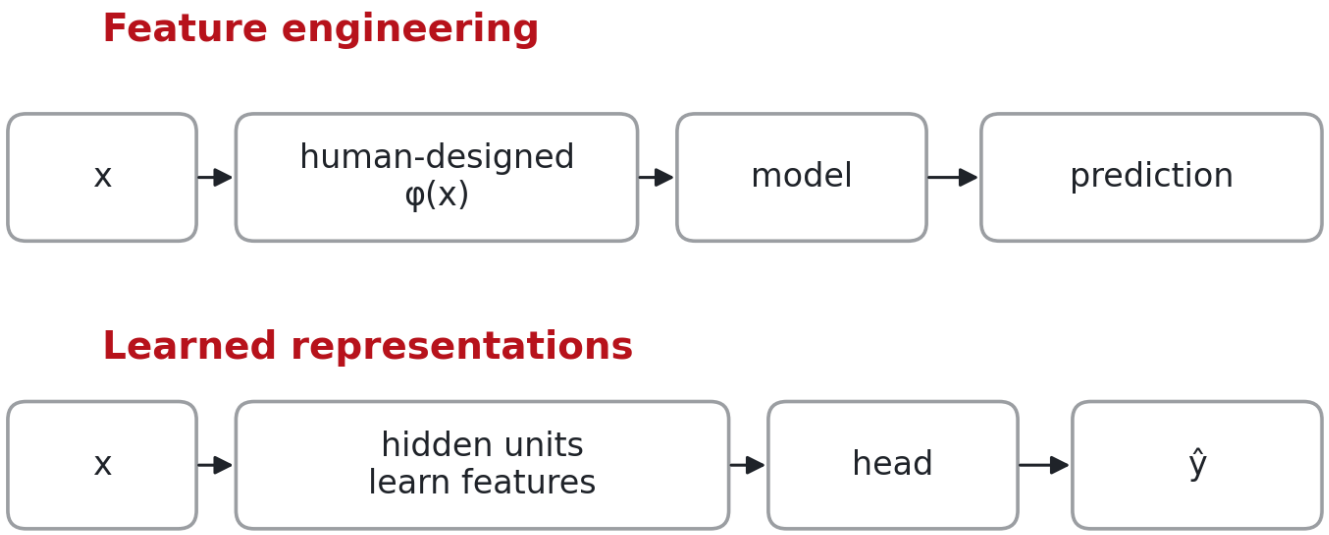
Hidden Unit

A hidden unit is a learned feature:

- $h_j = \phi(w_j^T x + b_j)$

Classical ML:

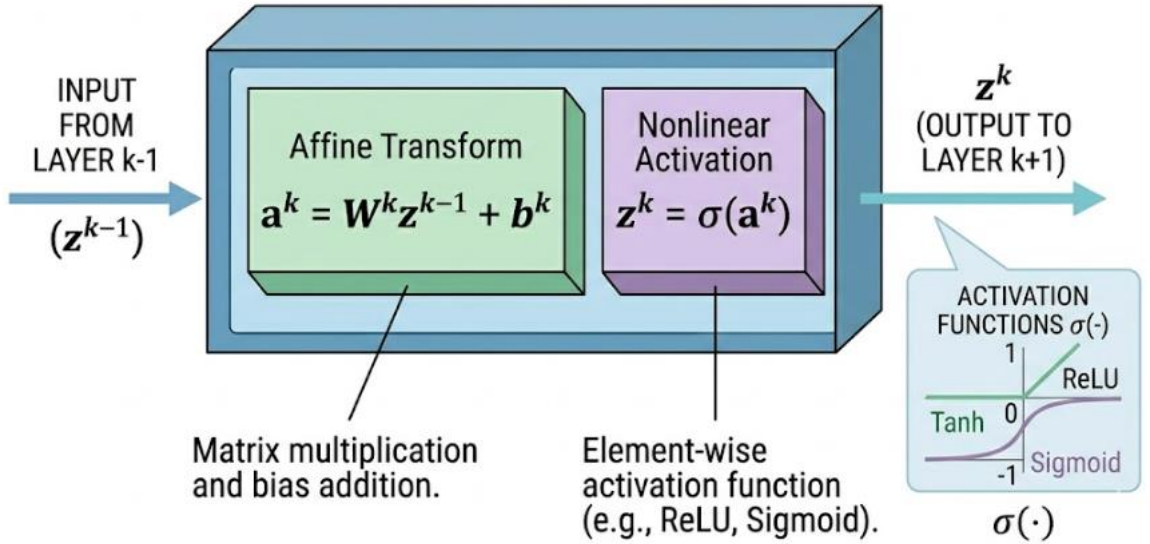
- Humans designed ϕ(x) → Model learns features automatically.

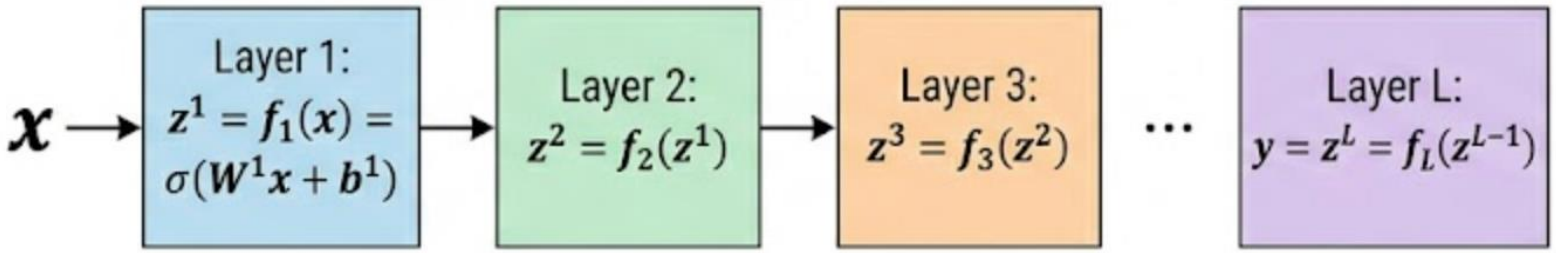


Forward Pass = Repeated Feature Transformation

- Layer operation: Affine transform + Nonlinear activation
- Network computation (Function composition):

$$f(x) = f_L(f_{L-1}(\cdots f_1(x)))$$





3. Nonlinearity: What Makes Depth Useful

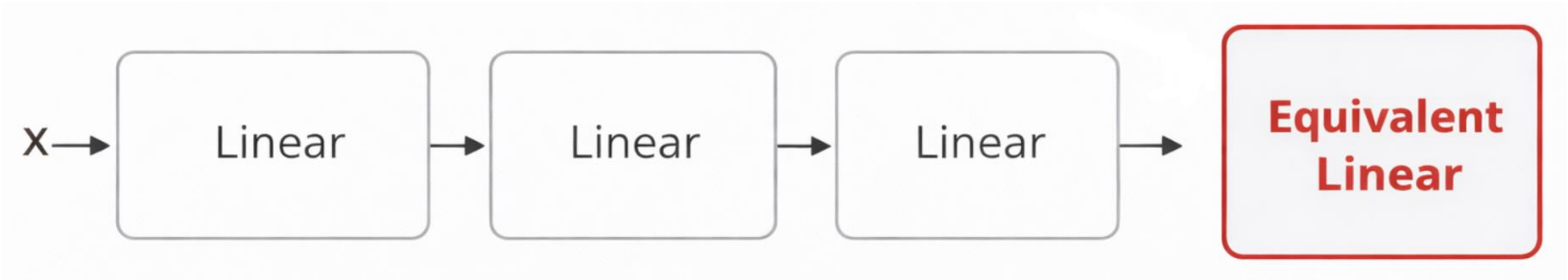
Why Nonlinearity?

If every layer were linear:

$$W_2(W_1x + b_1) + b_2 = W'x + b'$$

So:

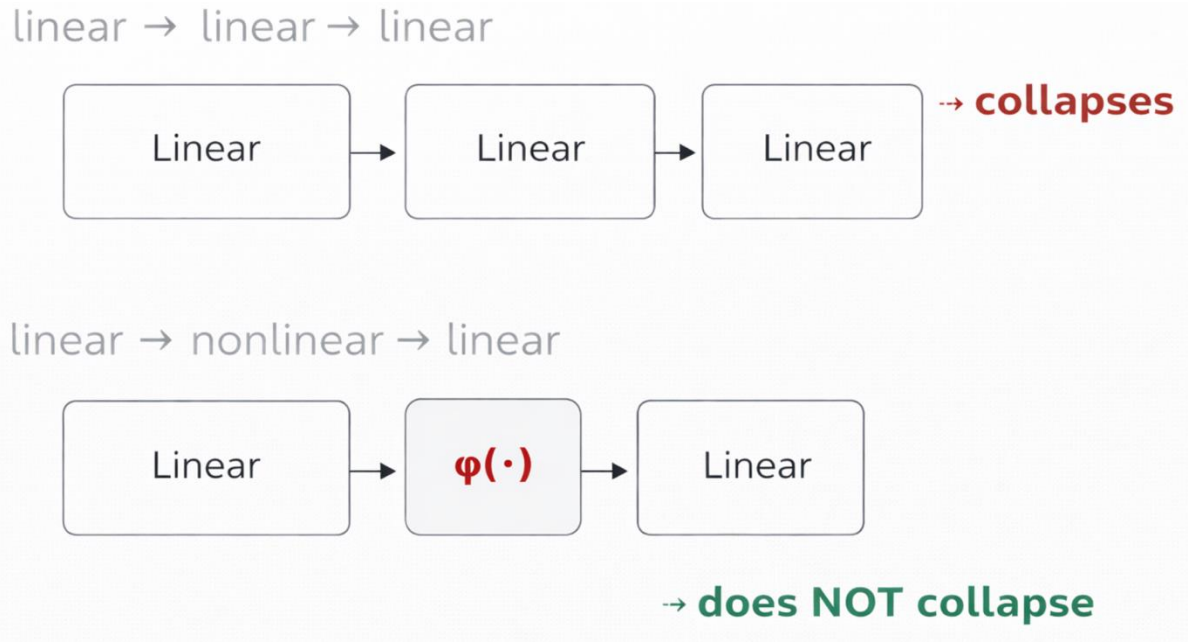
- Without nonlinearity, depth adds no expressive power.
- Core reason neural nets beat linear models.



What Exactly Is Activation Doing?

"An activation function helps because it _____."

- Breaks linear collapse
- Bends representation space
- Enables nonlinear transformations



Activation = Tradeoffs

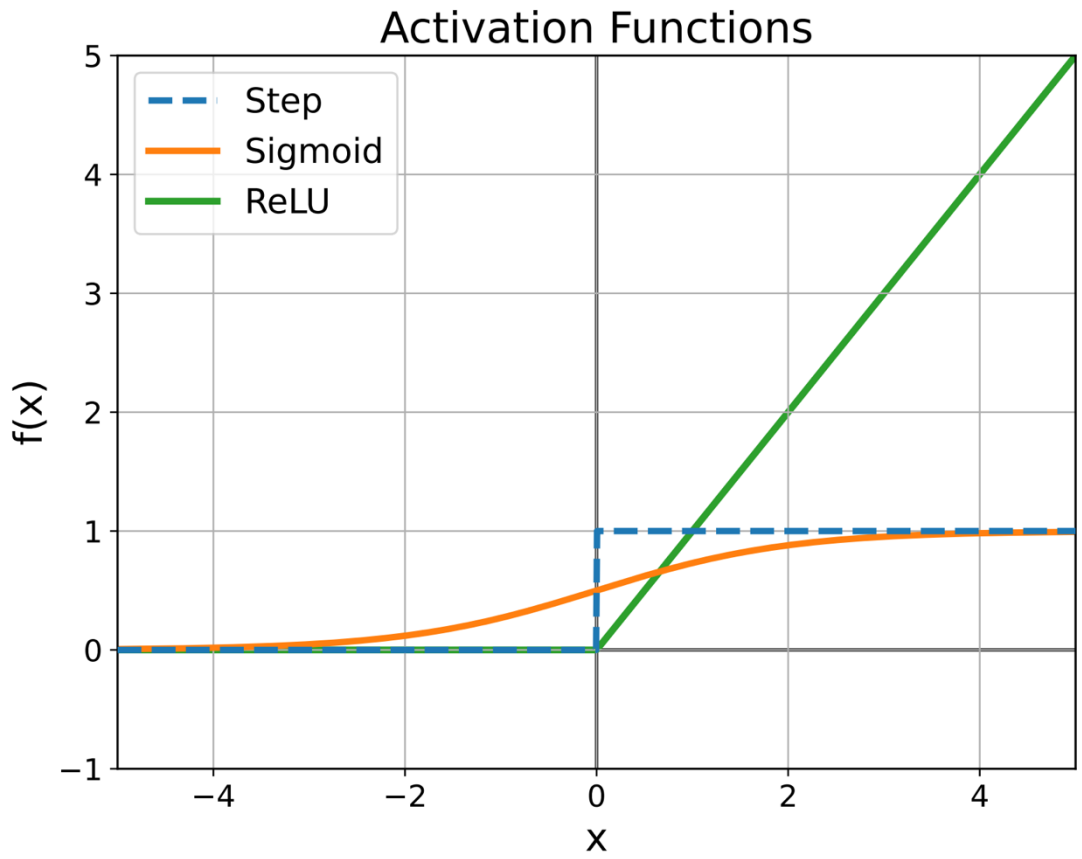
Think in tradeoffs:

For each function, analyze:

- Range / Behavior
- Key advantage
- Key weakness (e.g., vanishing grads)
- Typical usage

Functions:

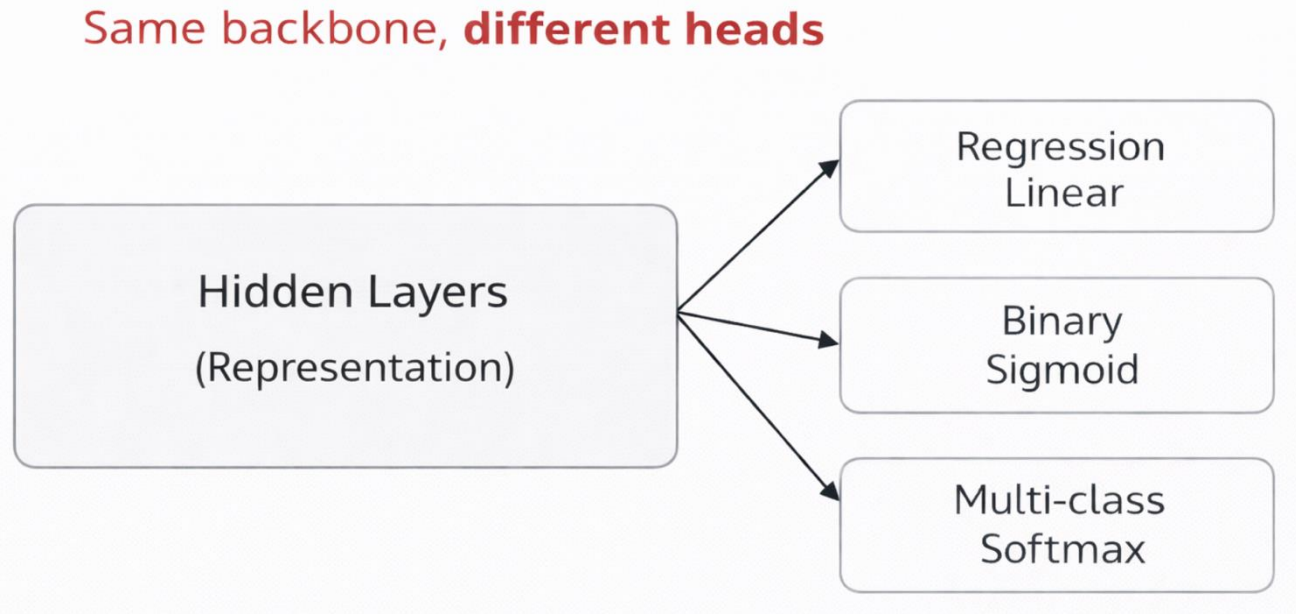
- Step, Sigmoid, ReLU



Hidden Layers vs Output Layers

Hidden-layer activation:
- Shapes representation
- Affects optimization
- Affects expressivity

Output-layer activation:
- Regression → Linear
- Binary Classification → Sigmoid
- Multi-class Classification → Softmax



Exercise: Output Head

For each task, choose output activation & justify:

1. House Prices

2. Spam Filter

3. Digit (0-9)

Question:

What assumption about the target space is encoded by each choice?

Universal Approximation

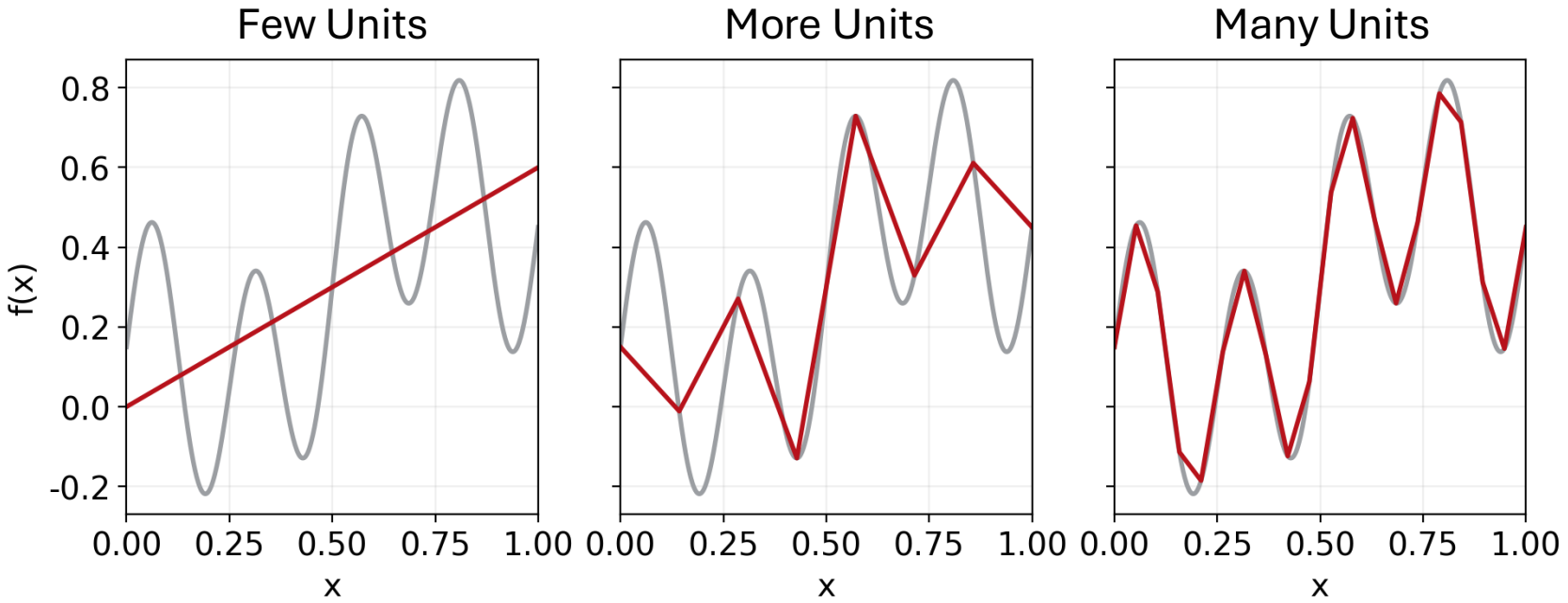
Theorem: A sufficiently wide one-hidden-layer MLP can approximate a broad class of continuous functions.

What it says (Theory):

- Shallow networks are expressive in principle.

What it does NOT say (Practice):

- Training is easy or efficient.
- Generalization is guaranteed.



[1] Hornik, Kurt; Stinchcombe, Maxwell; White, Halbert (January 1989). "Multilayer feedforward networks are universal approximators". *Neural Networks*. **2** (5): 359–366. doi:10.1016/0893-6080(89)90020-8

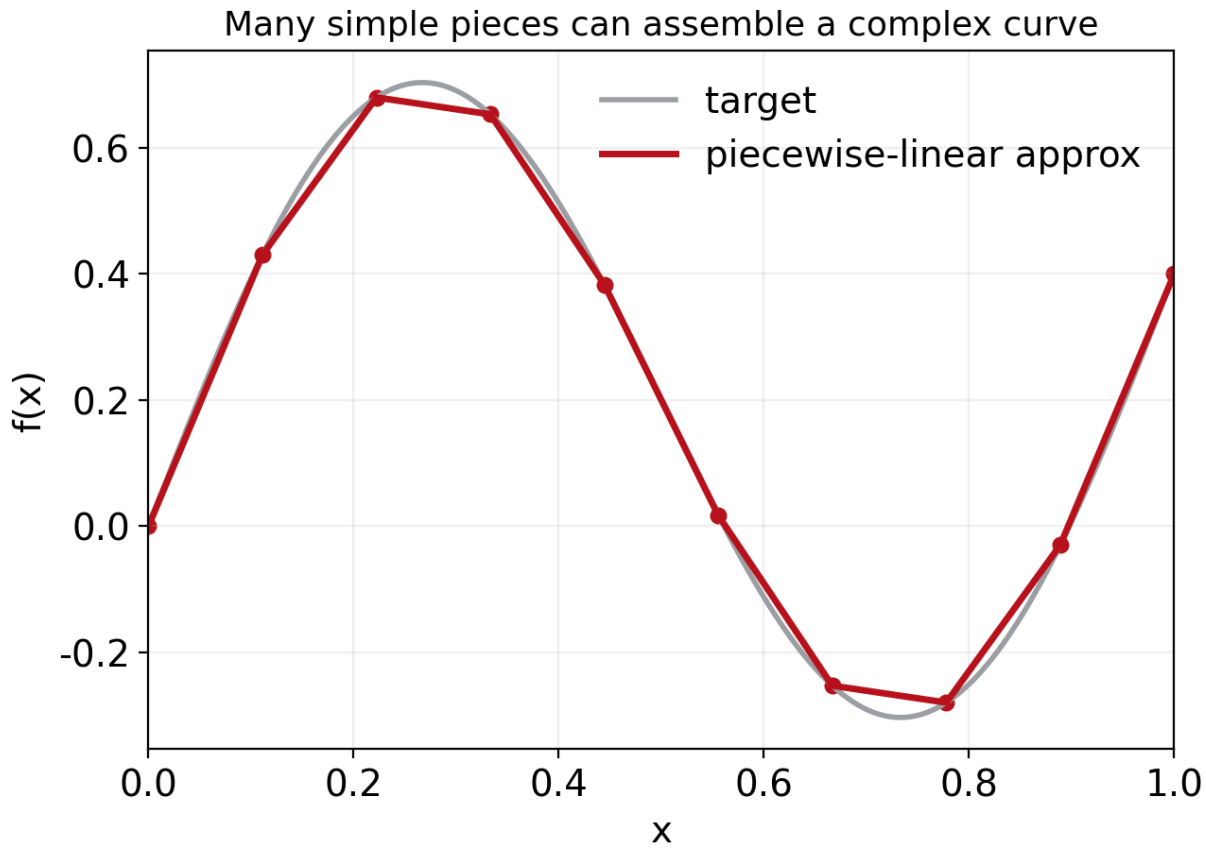
MLP Approximation

Mechanics:

- Sigmoid → soft regions
- ReLU → piecewise segments

Intuition:

- “Many simple building blocks can assemble a complex function.”



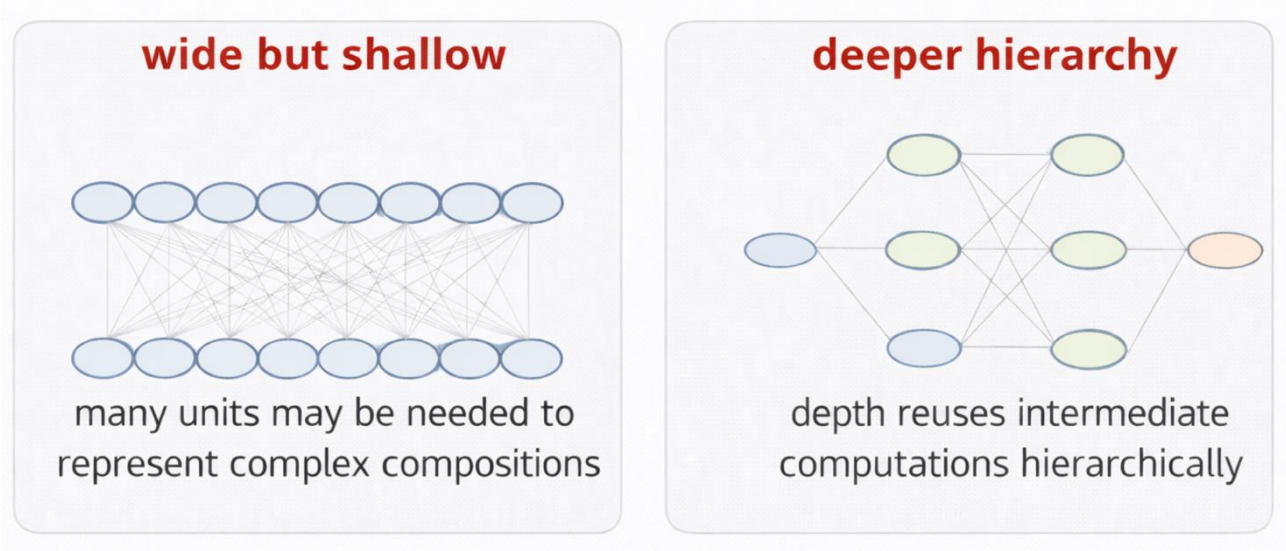
If One Hidden Layer Suffices, Why Depth?

Key answer: Representability in theory is not efficiency in practice.

Depth matters because of:

- parameter efficiency & feature reuse
- compositional structure & hierarchical abstraction

Note: Shallow models may need dramatically more hidden units.



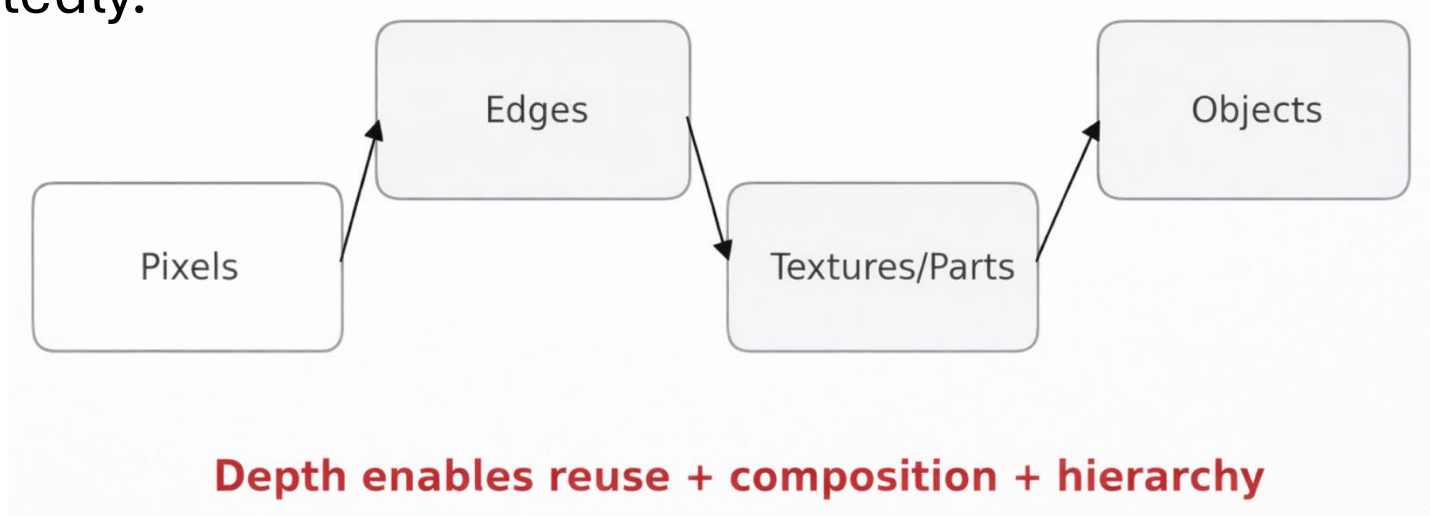
Depth = Reuse + Composition + Hierarchy

Example in vision:

- early layers: edges
- middle layers: motifs / parts
- later layers: objects

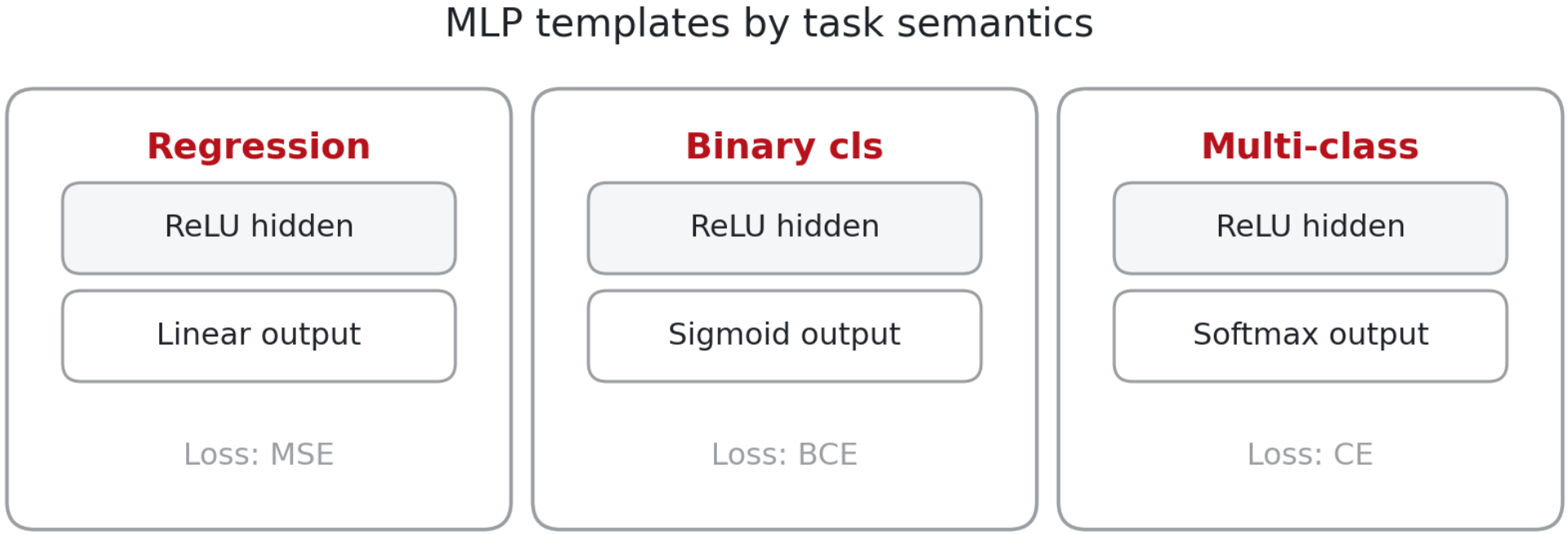
General principle:

- Deeper networks can reuse intermediate computations instead of relearning them repeatedly.



MLP as a Practical Template

Three Standard Architectures



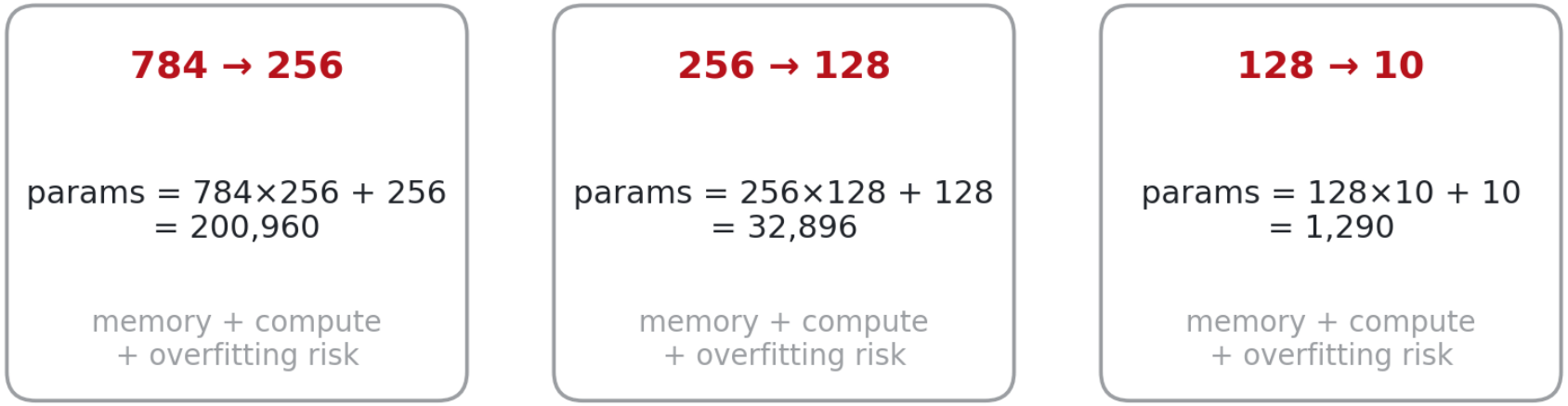
Counting Parameters: Width Is Not Free

Fully connected layer (n → m): **#params = O(nm)**

Implications of fully-dense connections:

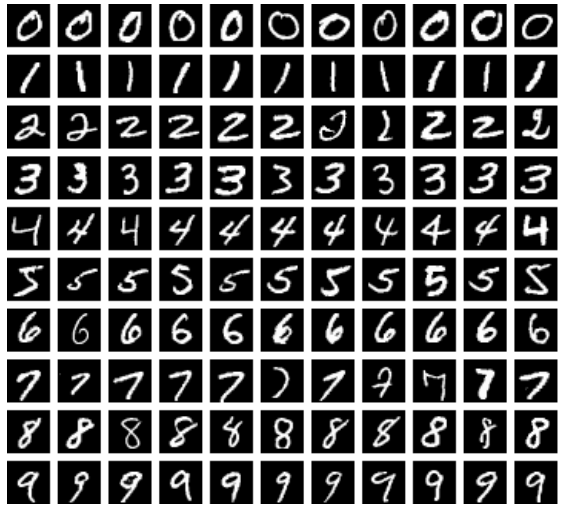
- Massive memory & compute scaling
- High overfitting risk
- Inefficient for high-dim structured inputs (e.g. images) → Motivates CNNs

Width is not free: fully-connected layers scale as nm



Example: MLP for MNIST Classification

- Input: 28 x 28 = 784
- Output: 10



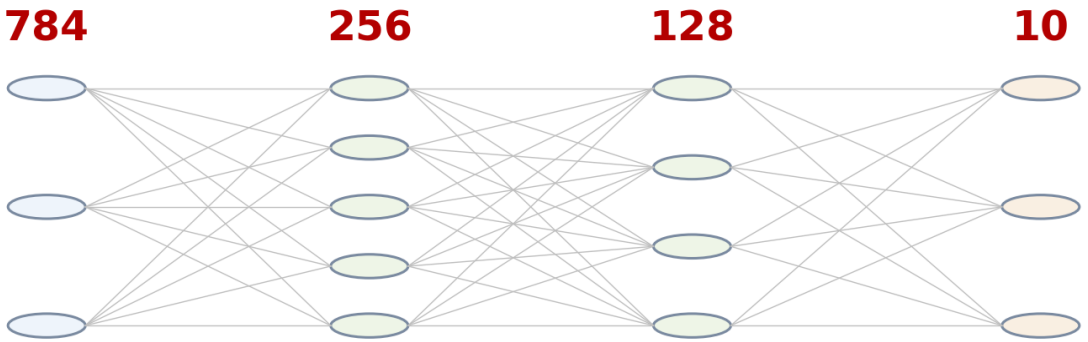
MNIST dataset

A reasonable baseline:

- 784 -> 256 -> 128 -> 10
- ReLU hidden layers, softmax output

Now critique it:

- Why this depth?
- Why these widths?
- What are its likely weaknesses?



parameters = 784×256 + 256 + 256×128 + 128 + 128×10 + 10

Summary: Three Core Concepts

1. **Why perceptron is limited:**
   Still a strictly linear hypothesis class.
2. **How MLP breaks the limit:**
   Composition (affine + activation) = learned features.
3. **Why depth is better than width:**
   Efficiency, hierarchy, and feature reuse in practice.

Recap of the intellectual arc

We Built It. How Do We Train It?

This Lecture (Architecture):

- Neural network basics
- Role of activation
- Value of depth

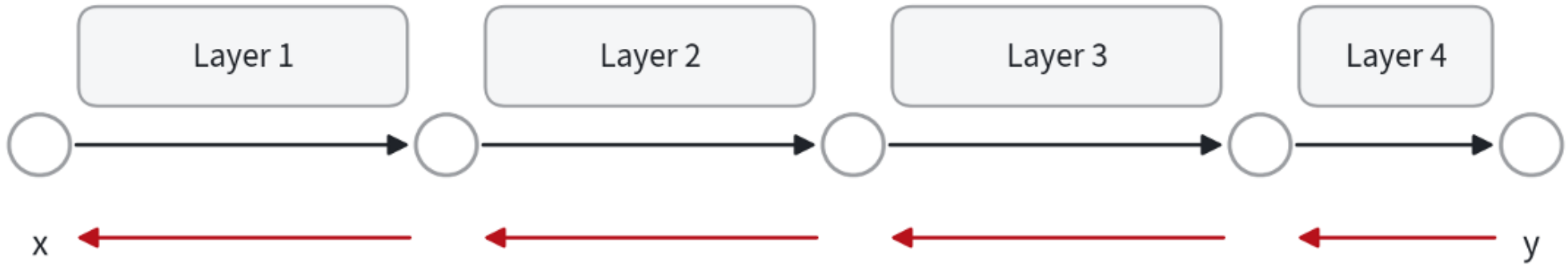
Next Lecture: Backpropagation

- Computational graphs & chain rule
- Gradient flow through layers

Major remaining question:

How to compute gradients efficiently?

Next: backpropagation = differentiate through the computation



Backward gradients