Neural Networks Discovery

Understanding How Neurons Build Intelligence

Pre-Class Discovery Handout — Time: 40-50 minutes

Objective: Discover how simple neurons combine to solve complex problems through hands-on exploration. **No prior knowledge required!**

Part 0: What is a Neuron? Building from Zero (15 minutes)

The Party Decision Problem

Your Friend Alex's Decision

Alex needs to decide: Should I go to this party? Two factors matter:

- **Distance** (km): How far is the party?
- **Friends**: How many friends are going?

Alex's personal formula for decisions:

Decision Score = $(-2 \times \text{Distance}) + (3 \times \text{Friends})$ - 5

If Score $> 0 \rightarrow$ **GO!** — If Score $\leq 0 \rightarrow$ **STAY HOME**

Your task: Calculate Alex's decision for these parties. Fill in the table:

Distance	Friends	Score Calculation	Decision
1 km	2	$(-2 \times 1) + (3 \times 2) - 5 =$ _____	GO / STAY
5 km	1	$(-2 \times 5) + (3 \times 1) - 5 =$ _____	GO / STAY
2 km	3	_____	GO / STAY
3 km	4	_____	GO / STAY

Reflection Questions:

1. The number **-2** (distance weight) means: _____
2. Why is friends weight **+3** positive but distance **-2** negative? _____
3. The **-5** is called the "bias." What does it tell you about Alex? _____

This is a Neuron!

Key Takeaway

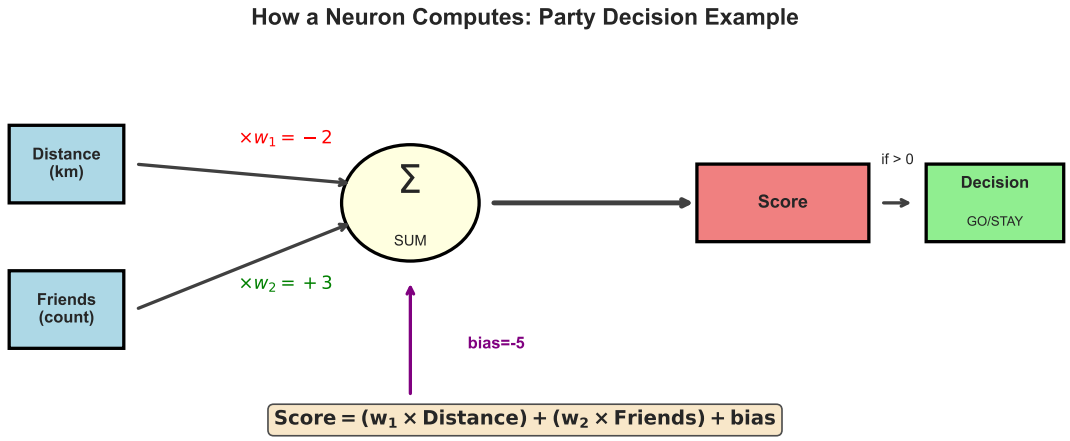
Congratulations! You just computed a **NEURON**. Here's the general formula:

$$\text{Output} = (w_1 \times \text{input}_1) + (w_2 \times \text{input}_2) + \text{bias}$$

Terminology:

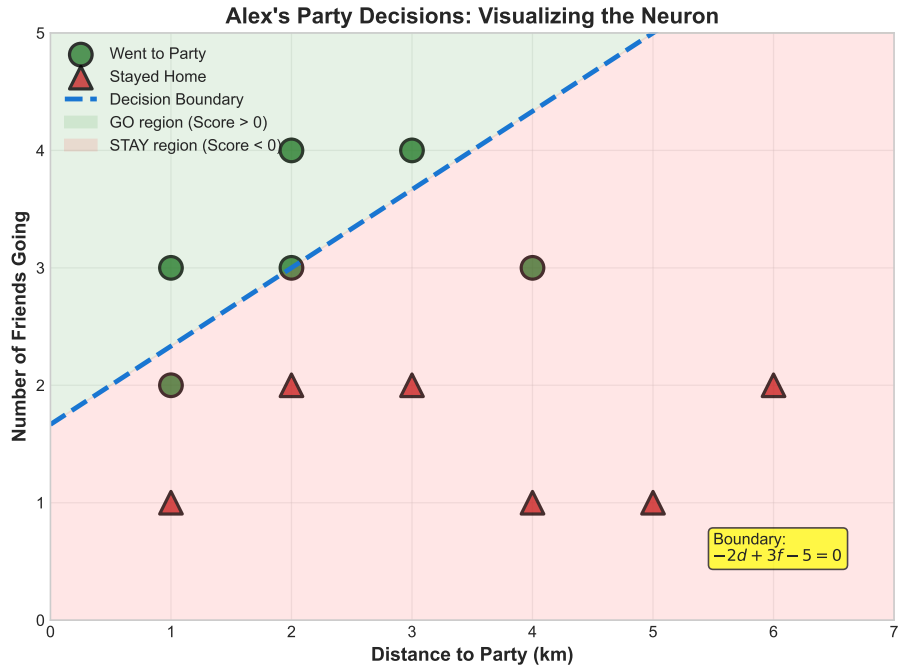
- w_1, w_2 = **WEIGHTS** (importance of each input)
- inputs = **DATA** (distance, friends, or any numbers)
- bias = **BASELINE** (starting preference before considering inputs)
- output = **SCORE** (final decision value)

Intuition: Positive weights mean "more is better," negative weights mean "less is better."



Visualizing Alex's Decisions

Now let's plot all of Alex's past party decisions on a graph:

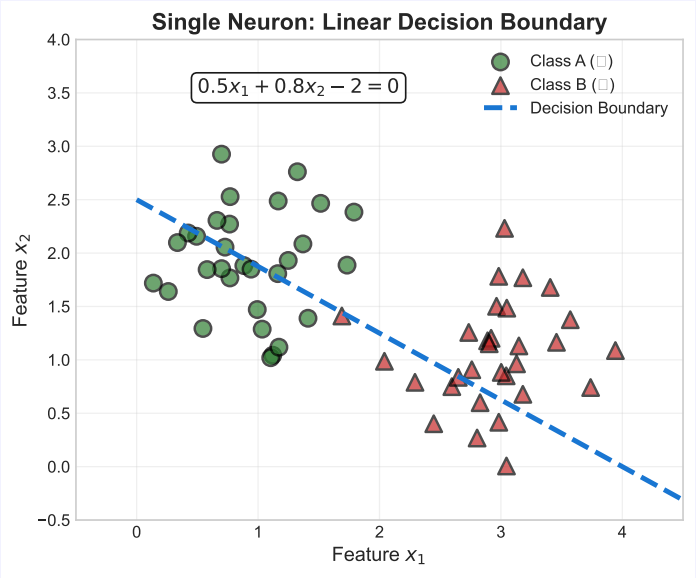


Discovery: The line $-2d + 3f - 5 = 0$ perfectly separates GO from STAY decisions! This is called the **decision boundary**, and it's always a _____ line (straight/curved).

Part 1: The Limitation - Only Straight Lines (8 minutes)

Understanding Linear Boundaries

Remember Alex's formula from Part 0? Let's see what happens with mathematical notation:



A single neuron computes: Output $= w_1x_1 + w_2x_2 + b$
When Output $= 0$, we get the boundary: $w_1x_1 + w_2x_2 + b = 0$ (this is a line!)
Critical Question:
Can this neuron separate data arranged in a **circular pattern** (inner circle vs outer circle)?
☐ Yes - I can draw one straight line to separate them
☐ No - I would need a curve, not a line
What's the fundamental limitation? _____

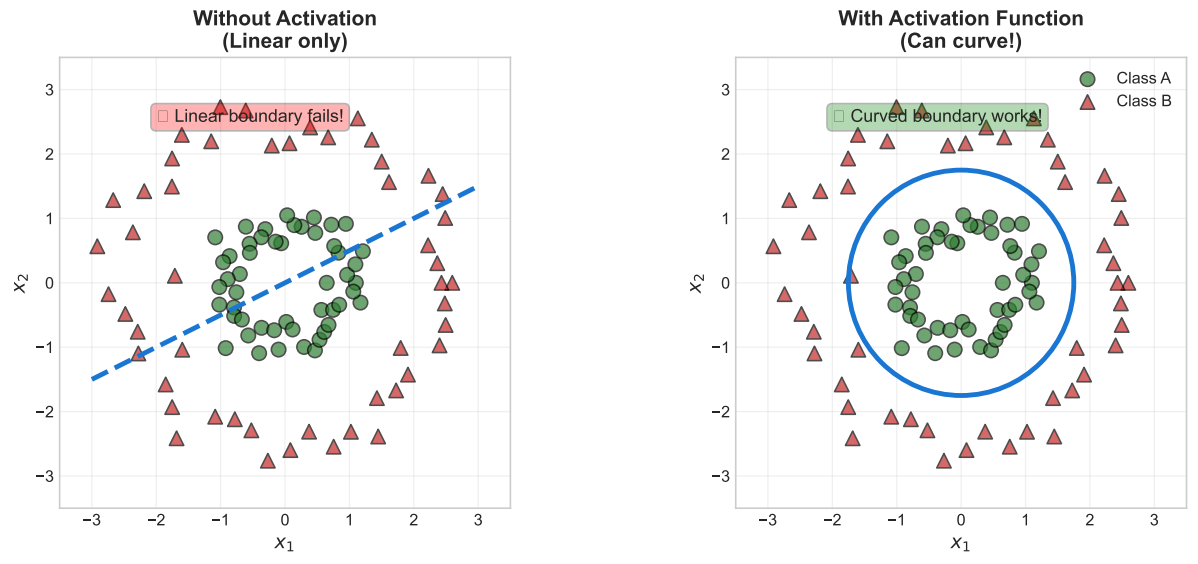
Key Takeaway

Without activation functions: A single neuron can **ONLY** draw straight lines. It cannot create curved boundaries!

Part 2: The Solution - Adding Curves (8 minutes)

From Lines to Curves with Activation Functions

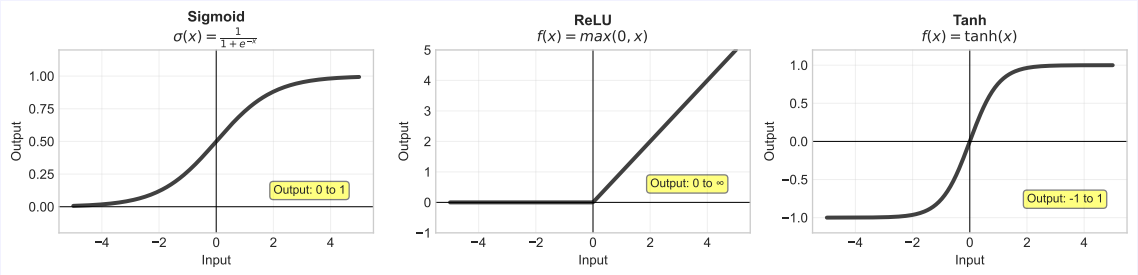
To handle circular or complex patterns, we add an **activation function** that "bends" the output.



Fill in the comparison:

Property	Without Activation	With Activation
Boundary shape	_____	_____
Can make curves?	Yes / No	Yes / No
Can separate circles?	Yes / No	Yes / No

Common Activation Functions:



Question: Which activation function outputs values between 0 and 1? _____

Part 3: Two Neurons Are Better Than One - The XOR Problem (12 minutes)

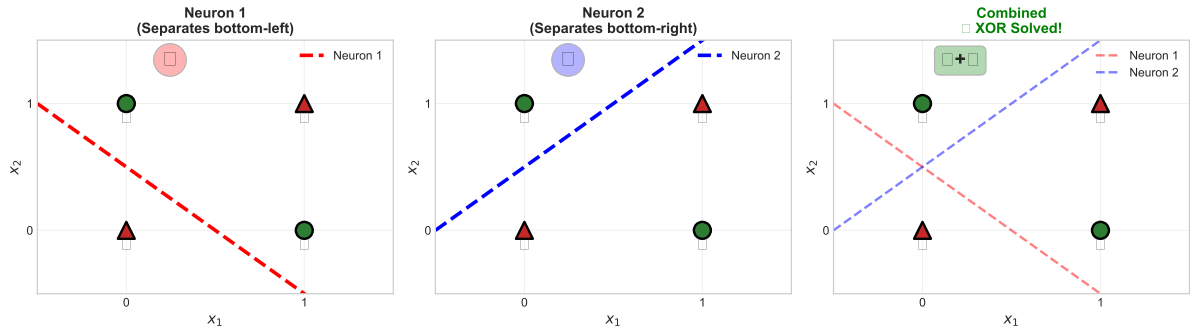
Why One Neuron Isn't Enough

The XOR (exclusive OR) problem: Output 1 if inputs are *different*, 0 if *same*.

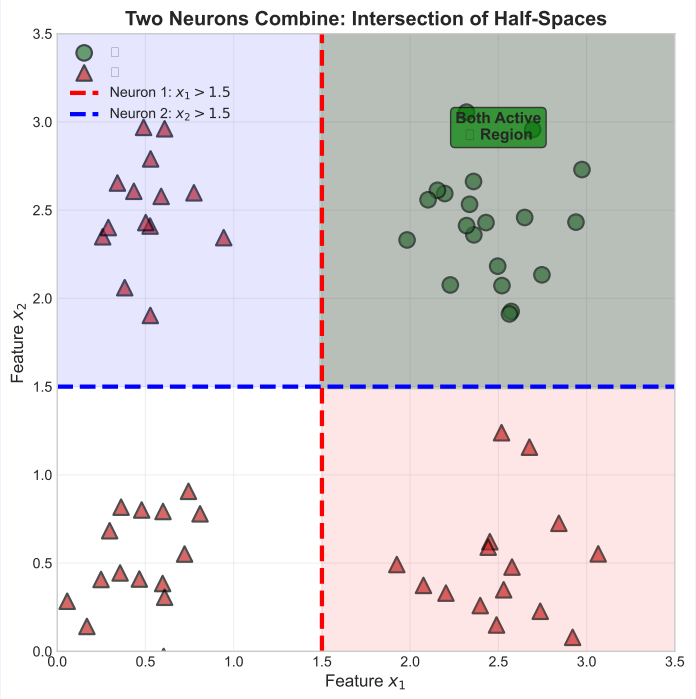
XOR Truth Table:

x_1	x_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Challenge: Try drawing **ONE** straight line that separates the 1's from 0's on the plot below.
Result: Impossible with one line!
Solution: Use **TWO** neurons!



Key Insight: Neuron 1 creates one boundary, Neuron 2 creates another. Together, their *intersection* solves XOR!



Trace Through Calculation: For XOR input $(x_1 = 1, x_2 = 0)$, assume:

- Hidden Neuron 1: weights=[1.0, 1.0], bias=-0.5
- Hidden Neuron 2: weights=[1.0, 1.0], bias=-1.5

a) Neuron 1: $z_1 = (1.0 \times 1) + (1.0 \times 0) - 0.5 =$ _____

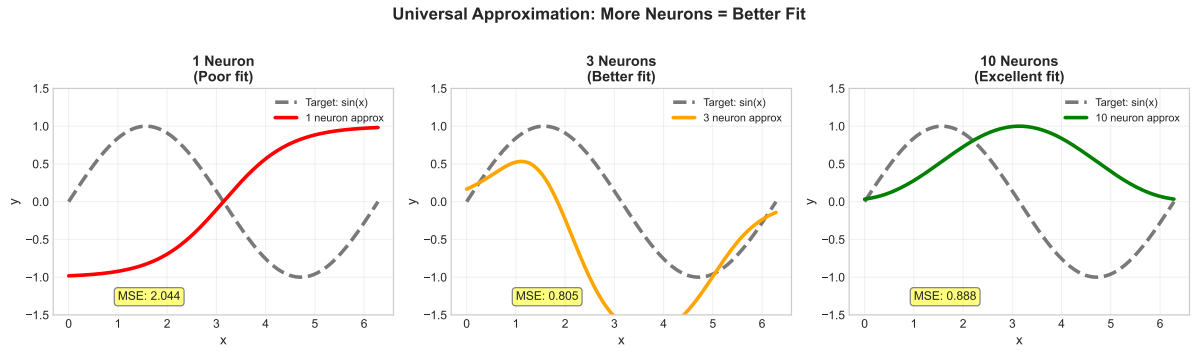
b) After sigmoid: $h_1 = \sigma(z_1) \approx$ _____ *(use 0.62 if $z_1 = 0.5$)*

c) Neuron 2: $z_2 = (1.0 \times 1) + (1.0 \times 0) - 1.5 =$ _____

Part 4: Many Neurons = Any Function! (5 minutes)

The Universal Approximation Theorem

As we add more neurons, we can approximate *any* smooth function to *any* accuracy!



Observations:

1. With 1 neuron, the fit is _____ (poor/excellent).
2. With 10 neurons, the approximation is _____ (poor/nearly perfect).
3. **Pattern:** More neurons = _____ approximation.

True or False:

☐ Neural networks with enough neurons can approximate any continuous function.

☐ One neuron is always enough to solve any problem.

☐ Activation functions are optional for neural networks.

Key Takeaway

Universal Approximation Theorem (Cybenko, 1989): A neural network with enough hidden neurons can approximate *any* continuous function to *any* desired accuracy. This is why neural networks are so powerful!

Summary: What You Discovered

Fill in the blanks to consolidate your learning:

1. A neuron computes: Output = _____
2. **Weights** control the _____ of each input.
3. **Bias** represents the _____ before considering inputs.
4. **Without activation:** Neurons can only make _____ boundaries.
5. **With activation:** Neurons can create _____ shapes.
6. **Multiple neurons:** Can solve problems like _____ that single neurons cannot.
7. **Many neurons:** Can approximate _____.

Before Class - Think About:

- How do we actually *learn* the right weights?
- What happens when we stack many layers of neurons?
- What are the limitations of neural networks?

Answers will be revealed in class. Bring your questions!