How Neural Networks Learn to Draw Any Curve

Function Approximation Discovery

Pre-Class Discovery Handout — Time: 40-45 minutes

Objective: Discover how neurons combine to approximate any smooth function through hands-on exploration. Focus on **continuous curves**, not decision boundaries.

Part 0: Can Computers Learn to Draw Curves? (10 minutes)

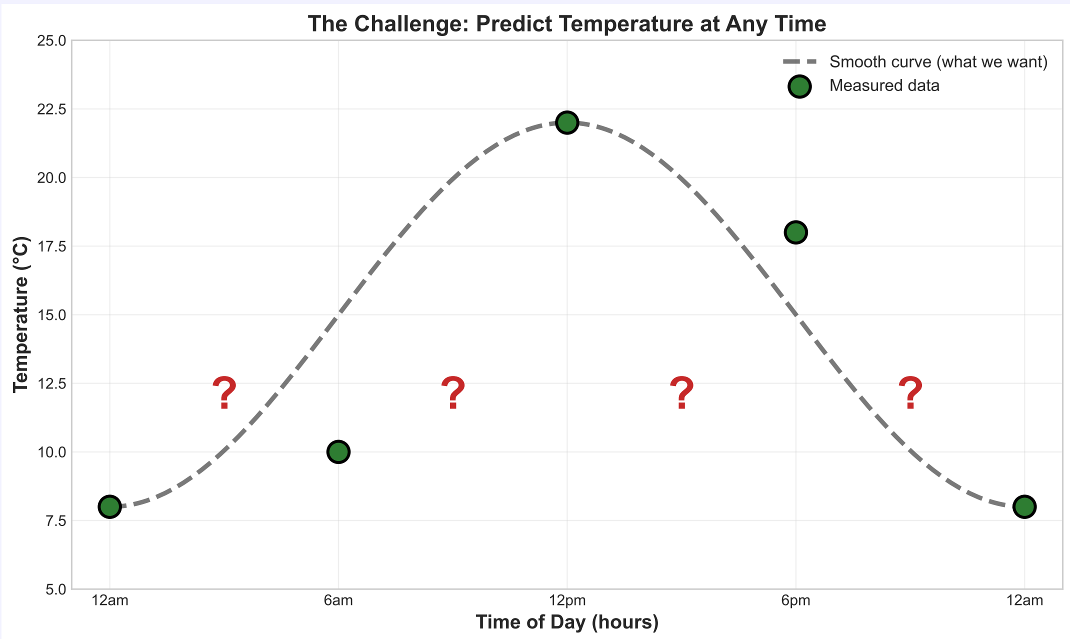
The Temperature Prediction Challenge

Real-World Problem

You want to predict temperature at *any time* during the day, but you only have a few measurements:

Time	Temperature
12am (0h)	8°C
6am	10°C
12pm (12h)	22°C
6pm (18h)	18°C
12am (24h)	8°C

Your Challenge: What is the temperature at 9am? At 3pm? At 9pm?



Questions:
1. Could you connect the dots with *straight lines*? ____________ (Yes/No)
2. Would straight lines give good predictions? ____________ (Yes/No)
3. What do you need instead? ______________________________ (smooth curves / straight lines)

Key Takeaway

Function Approximation means finding a smooth mathematical formula that fits data points. Neural networks can learn these formulas automatically!
Key difference from classification:
- Classification: "Is this A or B?" (discrete categories)
- Function approximation: "What is the exact value?" (continuous numbers)

Part 1: One Neuron Makes an S-Curve (8 minutes)

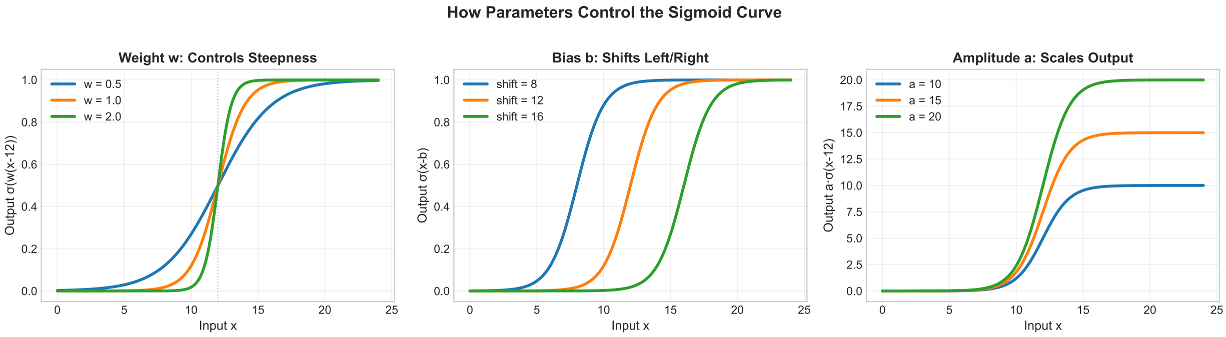
The Sigmoid Function

Remember from the first handout: A neuron uses a sigmoid function:

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

But now we can **control its shape** with three parameters:

$$\text{Output} = a \times \sigma(w \times (x - b))$$



Understanding Parameters

Parameter roles:

- **w** (weight): Controls ________________ (steepness / position / height)
- **b** (bias): Shifts curve ________________ (up-down / left-right / steeper)
- **a** (amplitude): Controls ________________ (maximum height / steepness / color)

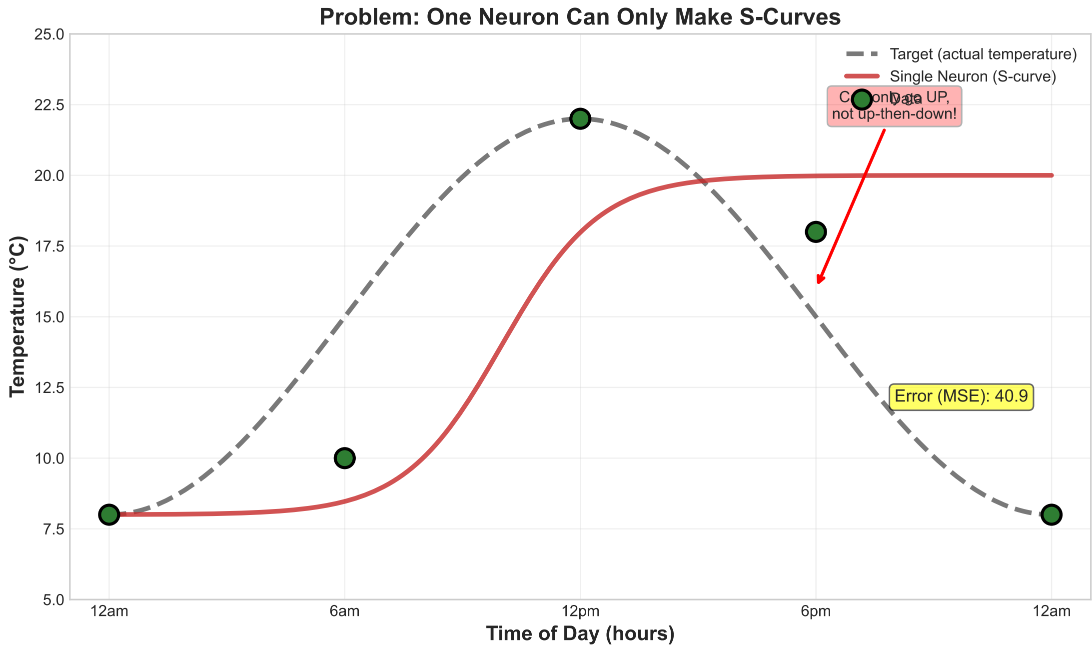
Hand Calculation: Use this approximation table for sigmoid:

Input	σ(input)	Input	σ(input)
-3	0.05	0	0.50
-2	0.12	1	0.73
-1	0.27	2	0.88
		3	0.95

Calculate: $y = 20 \times \sigma(0.5 \times (x - 12))$ for different times:

Time (x)	Calculation	Temperature (y)
6am (x=6)	$0.5 \times (6 - 12) = -3$, so $\sigma(-3) = 0.05$, thus $y = 20 \times 0.05 =$ ________	
12pm (x=12)	$0.5 \times (12 - 12) = 0$, so $\sigma(0) =$ ______, thus $y =$ ________	
6pm (x=18)	$0.5 \times (18 - 12) =$ ______, so $\sigma($______$) =$ ______, thus $y =$ ________	

The Problem: S-Curves Can't Do Everything

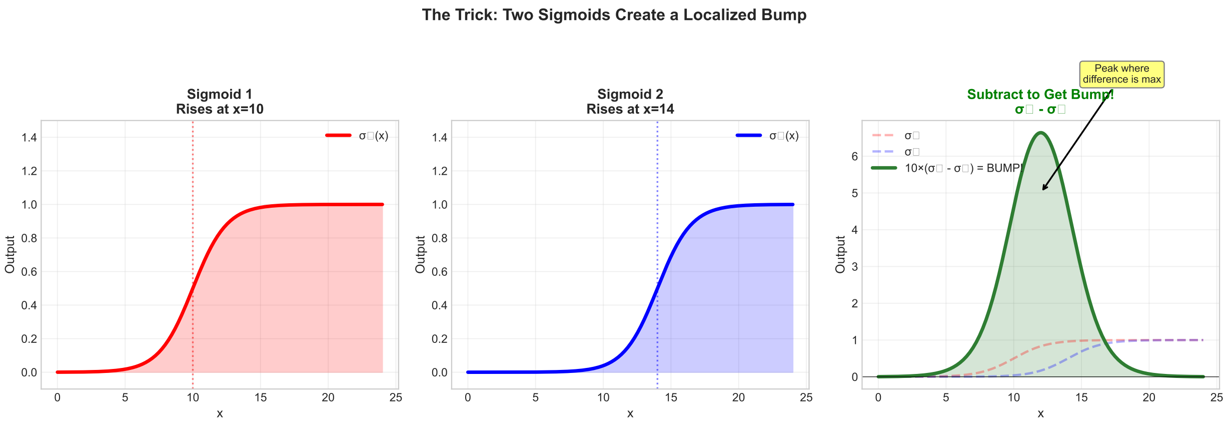


Discovery: A single neuron can only create curves that go UP (or only go DOWN). Real temperature goes up in the morning AND down in the evening!

Part 2: Two Neurons Make a Bump (10 minutes)

The Subtraction Trick

Key Insight: If we *subtract* two sigmoids, we get a localized "bump"!



Building a Bump

Given two neurons:
- Neuron 1: $\sigma_1 = \sigma(0.8 \times (x - 10))$ (rises at x=10)
- Neuron 2: $\sigma_2 = \sigma(0.8 \times (x - 14))$ (rises at x=14)

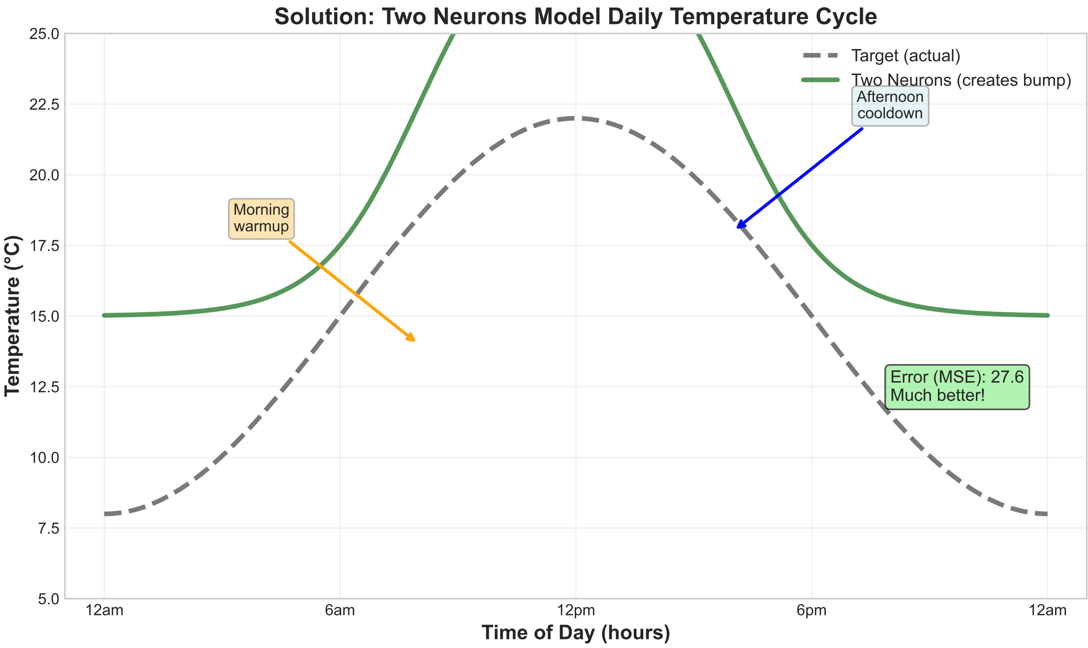
Output: $y = 10 \times (\sigma_1 - \sigma_2)$

Calculate for different times:

Time (x)	σ_1	σ_2	$y = 10 \times (\sigma_1 - \sigma_2)$
x=10	0.50	0.05	$10 \times (0.50 - 0.05) =$ ____________
x=12	0.88	0.27	$10 \times (0.88 - 0.27) =$ ____________
x=14	0.95	0.50	________________________
x=16	0.95	0.88	________________________

Observations:
1. At x=10: σ_1 is rising, σ_2 is still low → ____________ difference
2. At x=12: Both rising, but σ_1 ahead → ____________ (large/small) difference
3. At x=16: Both high (near 1) → ____________ difference
4. This creates a ____________ (straight line / bump / step) shape!

Application: Modeling Daily Temperature

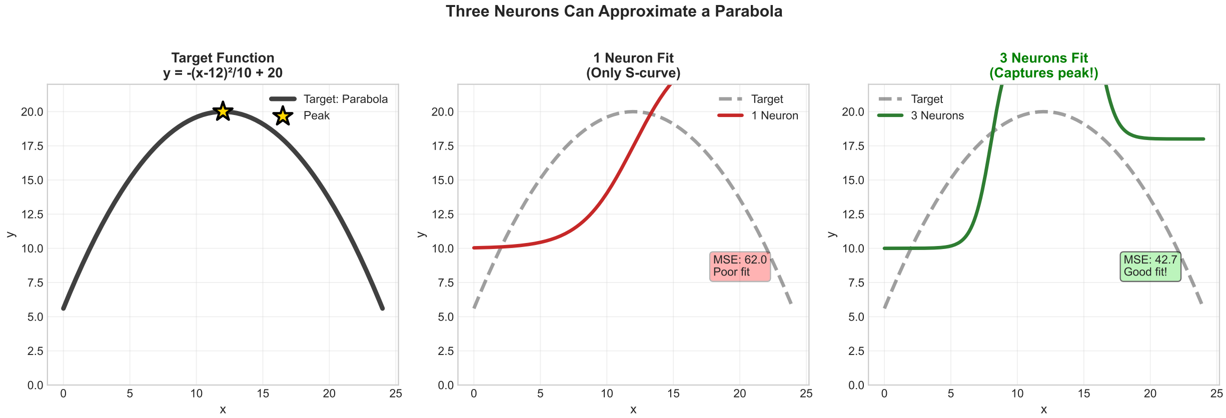


Discovery: Two neurons working together can model the *rise and fall* of temperature throughout the day!

Part 3: Three Neurons Capture Complex Patterns (10 minutes)

Approximating a Parabola (Peak Function)

Let's try to approximate a function with a clear peak: $y = -(x-12)^2/10 + 20$



Hand Calculation with 3 Neurons

The 3-neuron network:

- Baseline: $y_0 = 10$
- Neuron 1: $y_1 = 15 \times \sigma(1.5 \times (x-8))$ (handles left rise)
- Neuron 2: $y_2 = 10 \times \sigma(2.0 \times (x-12))$ (boosts peak)
- Neuron 3: $y_3 = -15 \times \sigma(1.5 \times (x-16))$ (handles right fall)

Total output: $y = y_0 + y_1 + y_2 + y_3$

Calculate at the peak (x=12):

a) Neuron 1: $1.5 \times (12-8) = 6 \rightarrow \sigma(6) \approx 1.0$, so $y_1 = 15 \times 1.0 =$ ____________

b) Neuron 2: $2.0 \times (12-12) = 0 \rightarrow \sigma(0) =$ ______, so $y_2 = 10\times$ ______ $=$ ____________

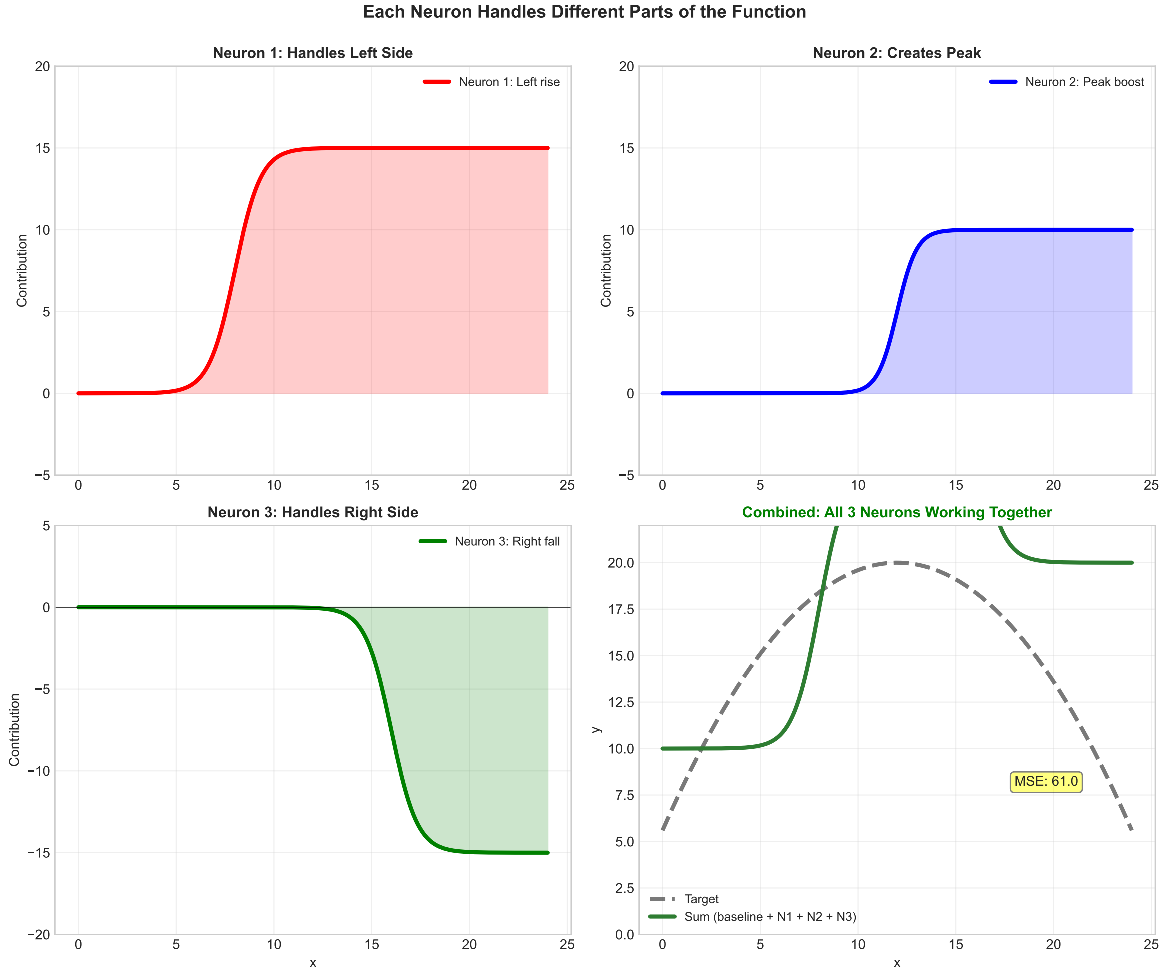
c) Neuron 3: $1.5 \times (12-16) = -6 \rightarrow \sigma(-6) \approx 0.0$, so $y_3 = -15 \times 0.0 =$ ____________

d) **Total:** $y = 10+$ ______ $+$ ______ $+$ ______ $=$ ____________

Compare to target: $y = -(12-12)^2/10 + 20 = 20$

How close is your answer? __________________

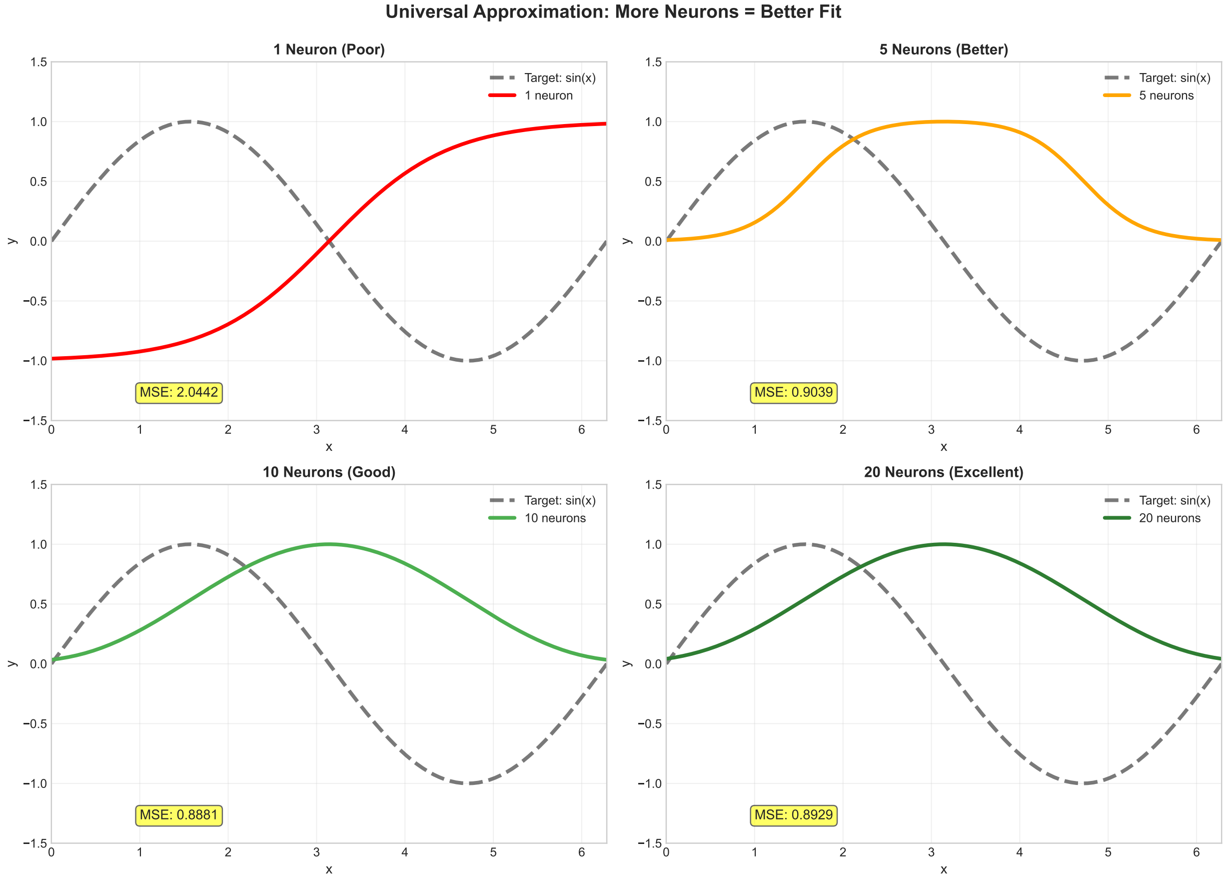
Each Neuron Has a Job



Pattern: More complex functions need more neurons, but each neuron handles a *specific part* of the curve!

Part 4: Many Neurons = Universal Approximation (7 minutes)

The Big Picture: Any Function, Any Accuracy



Observing the Pattern

Looking at the 4 panels above:

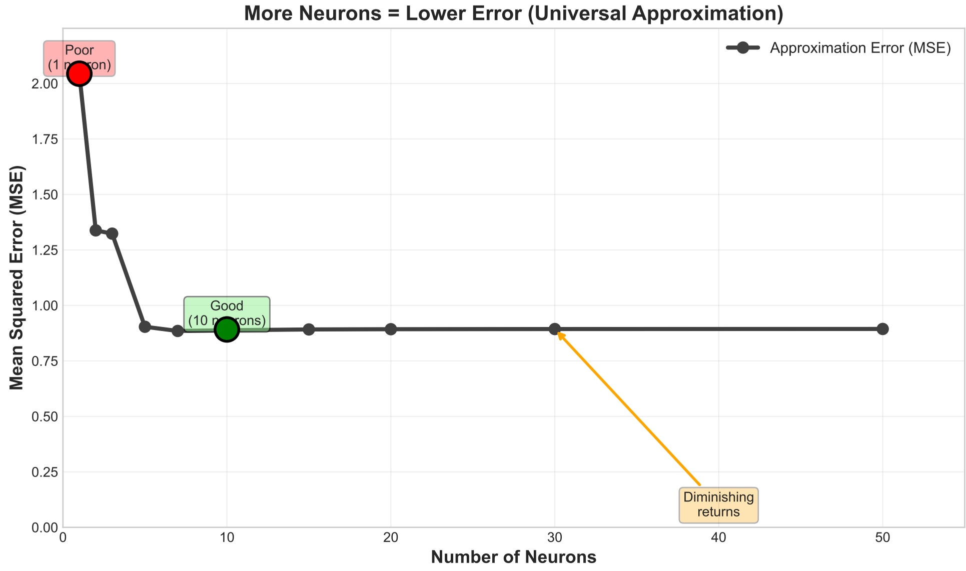
1. With 1 neuron: Error (MSE) is ____________ (high / low)
2. With 5 neurons: The fit is ____________ (worse / better) than 1 neuron
3. With 20 neurons: The approximation is ____________ (poor / nearly perfect)
4. **Pattern:** More neurons → ____________ (higher / lower) error

The Universal Approximation Theorem (Cybenko, 1989):
"A neural network with enough hidden neurons can approximate ANY continuous function to ANY desired accuracy."

What this means:

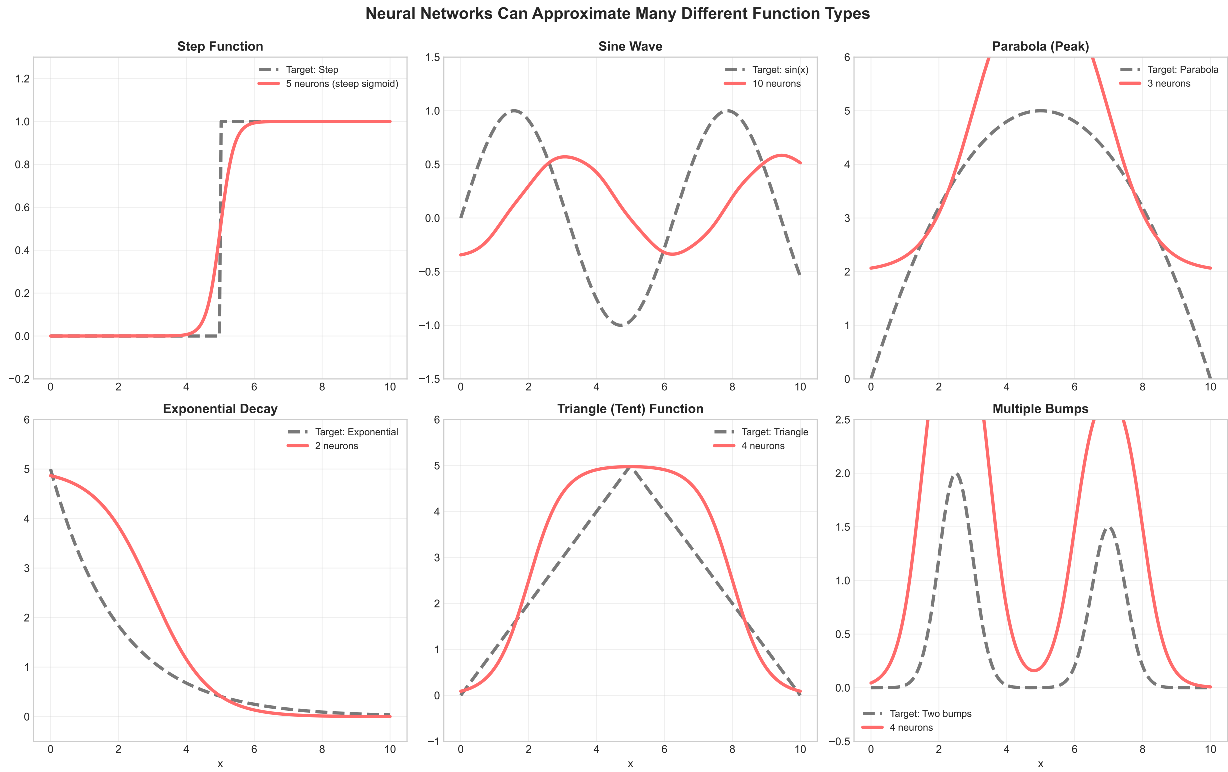
- **Universal:** Works for *any* smooth pattern (not just one type)
- **Guaranteed:** This is mathematically proven, not just hopeful
- **Practical:** Just add more neurons until fit is good enough

Error Decreases with More Neurons



Observation: After about 10-20 neurons, adding more gives *diminishing returns* (improvement slows down).

Many Different Function Types



Discovery: The same sigmoid-based neurons can approximate steps, waves, peaks, decays, triangles, and bumps!

Part 5: Summary & What's Next (5 minutes)

Fill in the blanks to consolidate your learning:

1. A sigmoid function creates an ______________-shaped curve (S / U / straight).
2. Subtracting two sigmoids creates a ______________ shape (line / bump / circle).
3. To approximate complex functions, we need ______________ (one / many) neurons.
4. More neurons means ______________ error (higher / lower).
5. The Universal Approximation Theorem guarantees that neural networks can fit ______________ (only linear / any continuous) function(s).

6. Each neuron in a network typically handles a ________________ (random / specific) part of the function.

Key Takeaway

Key Insights from This Handout:
- **Function approximation** = fitting smooth curves to data
- **Sigmoids are building blocks**: One sigmoid = S-curve, Multiple sigmoids = any shape
- **Parameters matter**: Weight (steepness), bias (position), amplitude (height)
- **Combination is key**: Neurons work together, each handling part of the function
- **Universal power**: With enough neurons, can approximate ANY smooth function

Before Class - Think About:
- How do we *automatically find* the right weights and biases for each neuron? (Hint: training/learning)
- What if we have *multiple inputs* (not just time)? Like temperature AND humidity?
- What's the downside of using 100 neurons when 10 would work? (Hint: computation, overfitting)

This handout complements the first handout (classification/decision boundaries). Together, they show neural networks can solve both discrete and continuous problems!