How Does AI Learn - Intro to Backpropagation in Neural Networks

Watch this first (~18 minutes): youtube.com/watch?v=aircAruvnKk&vl=en (3Blue1Brown)

Group Member Names: ____

Date: __________

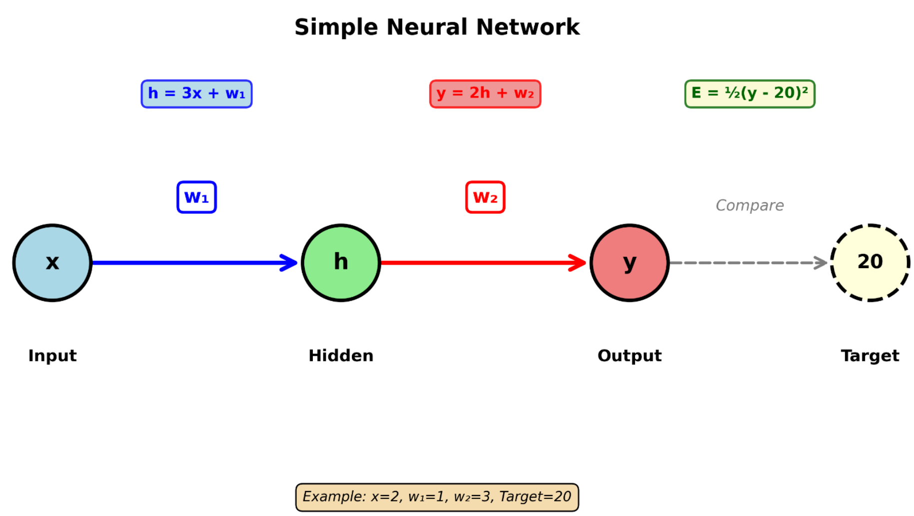
Introduction

You're going to discover how neural networks learn—using only the calculus you already know! A neural network is a mathematical function that learns from data by adjusting its internal parameters (called **weights**). Your challenge is to figure out HOW it adjusts these weights.

The Network

Here's our simple neural network:

Input (x) → [×3, then add w_1] → Hidden (h) → [×2, then add w_2] → Output (y)



Mathematical Form:

- Hidden layer: $h = 3x + w_1$
- Output layer: $y = 2h + w_2$
- Error: $E = ½(y - y\_target)^2$

Phase 1: The Forward Pass - Understanding What We Have

Given Information:

- Input: $x = 2$

- Current weights: $w_1 = 1$, $w_2 = 3$
- Target output: y_target = 20
- Our network needs to predict 20, but it won't... yet!

Task 1: Calculate the Current Output

Step 1: Calculate h (the hidden layer value)

$h = 3x + w_1 = 3(\_\_\_) + \_\_\_ = \_\_\_\_\_$

Step 2: Calculate y (the output)

$y = 2h + w_2 = 2(\_\_\_) + \_\_\_ = \_\_\_\_\_$

Step 3: Calculate the error

$E = ½(y - y\_target)^2 = ½(\_\_\_ - \_\_\_)^2 = \_\_\_\_\_$

Discussion Question: Is our network doing well? Why or why not?

Phase 2: Experimental Discovery - How Do Weights Affect Error?

The network needs to adjust w_1 and w_2 to reduce the error. Let's experiment!

Task 2: Exploring w_2 (The Output Weight)

Try changing w_2 slightly and see what happens to the error:

w_2 Value	h (stays same)	New y	New Error E	Change in E
3.0 (original)				baseline
3.1				
2.9				

Observations:

- When w_2 increases by 0.1, the error changes by approximately: __________
- When w_2 decreases by 0.1, the error changes by approximately: __________

- Should we increase or decrease w_2 to reduce the error? __________

Task 3: Exploring w_1 (The Hidden Weight)

Now try changing w_1:

w_1 Value	New h	New y	New Error E	Change in E
1.0 (original)				baseline
1.1				
0.9				

Observations:

- When w_1 increases by 0.1, the error changes by approximately: __________
- When w_1 decreases by 0.1, the error changes by approximately: __________
- Which weight (w_1 or w_2) has a bigger effect on the error? __________

Phase 3: The Calculus Connection - Making It Precise

You've discovered experimentally how weights affect error. Now let's use calculus to find the **exact** relationship.

Task 4: Direct Derivatives

Part A: How does y change with w_2?

Given: $y = 2h + w_2$ (and h doesn't depend on w_2)

dy/dw_2 = __________

Part B: How does the error change with y?

Given: $E = \frac{1}{2}(y - 20)^2$

$dE/dy = \frac{1}{2} \cdot 2(y - 20) \cdot 1 = (y - 20)$

At our current y = _____, this equals: dE/dy = __________

Task 5: The Chain Rule - Connecting the Pieces

The Big Question: How does error change with w_2?

We want: **dE/dw_2**

But notice:

- E depends on y
- y depends on w_2

This is a chain of dependencies! What calculus rule do we use?

Apply the Chain Rule:

$dE/dw_2 = (dE/dy) \cdot (dy/dw_2)$

$= (\_\_\_\_\_\_) \cdot (\_\_\_\_\_\_)$

$= \_\_\_\_\_\_\_\_\_\_$

Check Your Work: Does this value match your experimental results from Task 2?

Phase 4: The Harder Case - A Longer Chain

Task 6: Finding dE/dw_1

This one is trickier! The chain is longer:

w_1 affects h, which affects y, which affects E

Part A: Find each piece of the chain

1. How does h change with w_1?

    - Given: $h = 3x + w_1$

    $dh/dw_1 = \_\_\_\_\_\_\_\_\_\_$

2. How does y change with h?

    - Given: $y = 2h + w_2$

dy/dh = __________

3. How does E change with y?

    - We already found this!

dE/dy = __________

Part B: Chain them together!

$dE/dw_1 = (dE/dy) \cdot (dy/dh) \cdot (dh/dw_1)$

$= (\_\_\_\_\_\_) \cdot (\_\_\_\_\_\_) \cdot (\_\_\_\_\_\_)$

= __________

Check Your Work: Does this match your experimental results from Task 3?

Phase 5: Reflection and Connection

Task 7: Understanding What You Discovered

Question 1: Why was finding dE/dw_1 harder than finding dE/dw_2?

Question 2: What would happen if the network had 10 layers instead of 2? How many derivatives would you need to multiply together to find the gradient of the first weight?

Question 3: Modern neural networks like GPT-4 have billions of weights. How do you think they calculate gradients for all of them?

The Big Reveal: Backpropagation

What you just discovered is called **backpropagation** - the fundamental algorithm for training neural networks!

Key Ideas:

1. The **chain rule** lets us trace how each weight affects the final error
2. We work **backward** through the network (that's why it's called "back" propagation)
3. Once we know dE/dw for each weight, we can adjust weights to reduce error
4. This same process works for networks with millions or billions of weights!

The Update Rule:

new_weight = old_weight - learning_rate × (dE/dweight)

Task 8: Make It Better

Using your calculated gradients, update the weights to reduce the error:

Using learning_rate = 0.1:

w_2_new = w_2_old - 0.1 × (dE/dw_2)

= 3 - 0.1 × (______)

= __________

w_1_new = w_1_old - 0.1 × (dE/dw_1)

= 1 - 0.1 × (______)

= __________

Now calculate the new error with these updated weights:

h_new = 3(2) + w_1_new = __________

y_new = 2(h_new) + w_2_new = __________

E_new = ½(y_new - 20)^2 = __________

Did the error decrease? __________

Extension Challenge (If Time Permits)

Real neural networks use **nonlinear activation functions**. Here's a common one:

Sigmoid function: $\sigma(z) = 1/(1 + e^{-z})$

Its derivative: $\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z))$

Modified network:

- $h = \sigma(3x + w_1)$
- $y = 2h + w_2$
- $E = \frac{1}{2}(y - 20)^2$

Challenge: Find dE/dw_1 for this network. What's different?

Real-World Impact: Every time you use ChatGPT, Siri, or face recognition on your phone, the chain rule is working behind the scenes!