NYU

Lab 9

Backpropagation by Hand

DS-GA 1003 | Machine Learning | Spring 2026

2026.04.09 Presenter by Yihuai Hong

Today's Goals

1. **Trace forward & backward passes**

   with concrete numbers on a small network

2. **See how ReLU gates gradients**

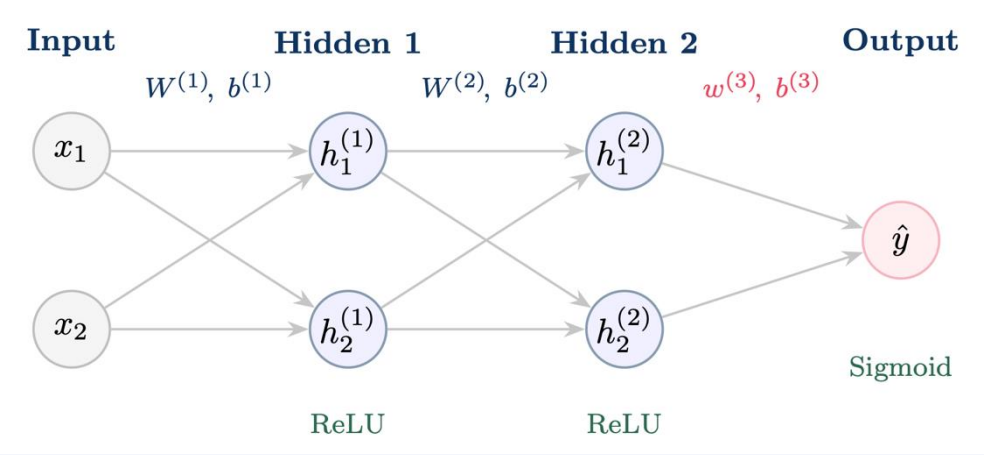
   and understand the "dying ReLU" phenomenon

3. **Practice the chain rule**

   exactly as you'll need for HW5

Network Architecture

3-layer feedforward network • 2 hidden ReLU layers • Sigmoid output • Binary cross-entropy loss



$$z^{(1)} = W^{(1)}x + b^{(1)}, \quad h^{(1)} = \mathrm{ReLU}(z^{(1)})$$

$$z^{(2)} = W^{(2)}h^{(1)} + b^{(2)}, \quad h^{(2)} = \mathrm{ReLU}(z^{(2)})$$

$$z^{(3)} = {w^{(3)}}^{\top} h^{(2)} + b^{(3)}, \quad \hat{y} = \sigma(z^{(3)})$$

$$\mathcal{L} = -\left[y \ln \hat{y} + (1-y)\ln(1-\hat{y})\right]$$

Given Parameters

Weights & Biases

$$W^{(1)} = \begin{bmatrix} 0.5 & -0.3 \\ 0.2 & 0.7 \end{bmatrix}, \quad b^{(1)} = \begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0.4 & -0.2 \\ -0.5 & 0.3 \end{bmatrix}, \quad b^{(2)} = \begin{bmatrix} 0.2 \\ 0.1 \end{bmatrix}$$

$$w^{(3)} = \begin{bmatrix} 0.6 \\ -0.4 \end{bmatrix}, \quad b^{(3)} = 0.3$$

Training Example

$$x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad y = 1$$

Quick Reference

$$\mathrm{ReLU}(z) = \max(0, z), \quad \mathrm{ReLU}'(z) = \mathbf{1}[z > 0]$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \quad \sigma'(z) = \sigma(z)(1 - \sigma(z))$$

$$\frac{\partial \mathcal{L}}{\partial z^{(3)}} = \hat{y} - y \quad \text{(BCE + sigmoid)}$$

Forward Pass — Layer 1

STEP 1.1

Compute $z^{(1)} = W^{(1)}x + b^{(1)}$, then apply ReLU element-wise.

$$z^{(1)} = \begin{bmatrix} 0.5 & -0.3 \\ 0.2 & 0.7 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.1 \end{bmatrix}$$

$$h^{(1)} = \mathrm{ReLU}\begin{pmatrix} 0.6 \\ 0.1 \end{pmatrix} = \begin{bmatrix} 0.6 \\ 0.1 \end{bmatrix}$$

Both pre-activations are positive → ReLU passes them unchanged.

Forward Pass — Layer 2

STEP 1.2

Same procedure, but watch which entry ReLU clips to zero.

$$z^{(2)} = \begin{bmatrix} 0.4 & -0.2 \\ -0.5 & 0.3 \end{bmatrix} \begin{bmatrix} 0.6 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 0.42 \\ -0.17 \end{bmatrix}$$

$$h^{(2)} = \mathrm{ReLU}\begin{pmatrix} 0.42 \\ -0.17 \end{pmatrix} = \begin{bmatrix} 0.42 \\ 0 \end{bmatrix}$$

⚠ The second neuron has $z_2^{(2)} = -0.17 < 0$ → ReLU kills it! This dead neuron will block gradient flow during backprop.

Forward Pass — Output Layer

STEP 1.3

Compute the pre-activation, apply sigmoid, and evaluate the loss.

$$z^{(3)} = \begin{bmatrix} 0.6 & -0.4 \end{bmatrix} \begin{bmatrix} 0.42 \\ 0 \end{bmatrix} + 0.3 = 0.552$$

$$\hat{y} = \sigma(0.552) \approx 0.6346$$

$$\mathcal{L} = -\ln(0.6346) \approx 0.4546$$

The network predicts ŷ ≈ 0.63, but the true label is y = 1. The loss is 0.45 — now we need gradients to improve.

The Backprop Recipe

At each layer ℓ, repeat these four steps:

1 **Error signal**

Gradient through activation

$$\delta^{(\ell)} = \frac{\partial \mathcal{L}}{\partial h^{(\ell)}} \odot \sigma'\left(z^{(\ell)}\right)$$

2 **Weight gradient**

Outer product with input

$$\frac{\partial \mathcal{L}}{\partial W^{(\ell)}} = \delta^{(\ell)} \cdot h^{(\ell-1)\top}$$

$$\frac{\partial \mathcal{L}}{\partial W^{(2)}} = \frac{\partial \mathcal{L}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial W^{(2)}}$$

3 **Bias gradient**

Equals the error signal

$$\frac{\partial \mathcal{L}}{\partial b^{(\ell)}} = \delta^{(\ell)}$$

4 **Pass backward**

Transpose multiply

$$\frac{\partial \mathcal{L}}{\partial h^{(\ell-1)}} = W^{(\ell)\top} \delta^{(\ell)}$$

Backward Pass — Output Layer

STEPS 2.1 – 2.3

Error signal:

$$\delta^{(3)} = \hat{y} - y = 0.6346 - 1 = -0.3654$$

Parameter gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(3)}} = h^{(2)} \cdot \delta^{(3)} = \begin{bmatrix} -0.1535 \\ 0 \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(3)}} = \delta^{(3)} = -0.3654$$

Gradient passed to $h^{(2)}$:

$$\frac{\partial \mathcal{L}}{\partial h^{(2)}} = w^{(3)} \cdot \delta^{(3)} = \begin{bmatrix} -0.2192 \\ 0.1462 \end{bmatrix}$$

Backward Pass — Through ReLU (Layer 2)

STEP 2.4

The ReLU derivative masks the gradient: entries where $z^{(2)} \leq 0$ become zero.

$$\delta^{(2)} = \frac{\partial \mathcal{L}}{\partial h^{(2)}} \odot \mathbf{1}[z^{(2)} > 0] = \begin{bmatrix} -0.2192 \\ 0.1462 \end{bmatrix} \odot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.2192 \\ 0 \end{bmatrix}$$

The Dying ReLU Problem

$z_2^{(2)} = -0.17 < 0 \rightarrow \text{ReLU}'(z_2^{(2)}) = 0 \rightarrow \delta_2^{(2)} = 0$

The second neuron is dead. No gradient flows through it. The entire second row of $\partial L/\partial W^{(2)}$ will be zero — those weights cannot learn from this example.

Mitigation: Leaky ReLU uses max(αz, z) with small α > 0, ensuring a nonzero gradient everywhere.

Backward Pass — Layer 2 Parameters

STEPS 2.5 – 2.6

$$\frac{\partial \mathcal{L}}{\partial W^{(2)}} = \delta^{(2)} {h^{(1)}}^{\top} = \begin{bmatrix} -0.1315 & -0.0219 \\ 0 & 0 \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(2)}} = \delta^{(2)} = \begin{bmatrix} -0.2192 \\ 0 \end{bmatrix}$$

Entire second row is zero — the dead ReLU blocks all gradient to its incoming weights.

Gradient passed to $h^{(1)}$:

$$\frac{\partial \mathcal{L}}{\partial h^{(1)}} = {W^{(2)}}^{\top} \delta^{(2)} = \begin{bmatrix} -0.0877 \\ 0.0438 \end{bmatrix}$$

Backward Pass — Layer 1

STEPS 2.7 – 2.8

Through ReLU (both neurons active — full gradient passes):

$$\delta^{(1)} = \frac{\partial \mathcal{L}}{\partial h^{(1)}} \odot \mathbf{1}[z^{(1)} > 0] = \begin{bmatrix} -0.0877 \\ 0.0438 \end{bmatrix}$$

Parameter gradients:

$$\frac{\partial \mathcal{L}}{\partial W^{(1)}} = \delta^{(1)} x^{\top} = \begin{bmatrix} -0.0877 & 0 \\ 0.0438 & 0 \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \delta^{(1)} = \begin{bmatrix} -0.0877 \\ 0.0438 \end{bmatrix}$$

Since $x_2 = 0$, the second column of $\partial L/\partial W^{(1)}$ is zero. Gradient magnitudes scale with previous-layer activations.

Gradient Summary

Parameter	Shape	Gradient
$b^{(3)}$	scalar	-0.3654
$w^{(3)}$	2×1	$[-0.1535, 0]^T$
$b^{(2)}$	2×1	$[-0.2192, 0]^T$
$W^{(2)}$	2×2	$[[-0.1315, -0.0219], [0, 0]]$
$b^{(1)}$	2×1	$[-0.0877, 0.0438]^T$
$W^{(1)}$	2×2	$[[-0.0877, 0], [0.0438, 0]]$

Parameter update (SGD):

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}$$

e.g. with $\eta = 0.1$: $b^{(3)} \leftarrow 0.3 - 0.1 \times (-0.3654) = 0.3365$

Discussion Questions

Q1 Why did the second row of $\partial L/\partial W^{(2)}$ end up all zeros?
What's the general implication for training deep networks with ReLU?

Q2 If we replaced ReLU with sigmoid in the hidden layers, how would Step 2.4 change?
Would the dead neuron problem still occur?

Q3 If we used $x = [0, 1]^T$ instead, which neurons (if any) would become dead?
Try to predict before computing.

Q4 After computing gradients, we do $\theta \leftarrow \theta - \eta \cdot \partial L/\partial \theta$.
With $\eta = 0.1$, what is the updated value of $b^{(3)}$?

Key Takeaways

Chain rule, layer by layer

Backprop is just the chain rule applied systematically from the loss back to the inputs.

The recipe at each layer

① Error signal δ through activation ② Weight gradient = outer product ③ Bias gradient = δ ④ Pass backward via transpose

ReLU gates gradients

Active neurons pass gradient unchanged (derivative = 1). Dead neurons block it entirely (derivative = 0).

Activations matter

Gradient magnitudes depend on both the upstream error signal and the forward-pass activations.