Neural Networks from Scratch

Neural Networks from Scratch Pt 1

Caleb Hallinan

01/06/2026

Announcements

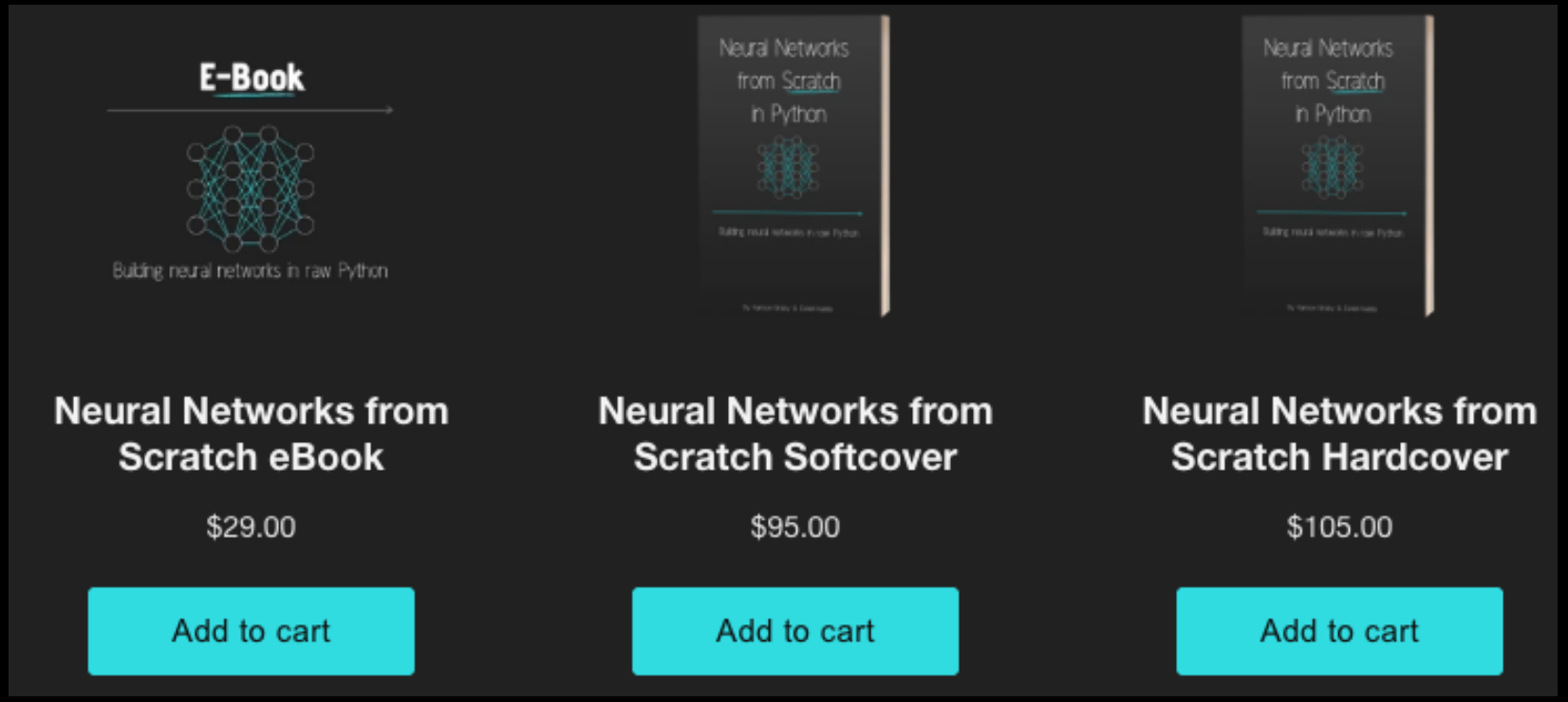
- Lecture from yesterday is on GitHub
- Let me know if you need help with your GitHub!

Outline for today

- Begin going through neural networks from scratch – forward propogation
- And brief review on derivatives and chain rule

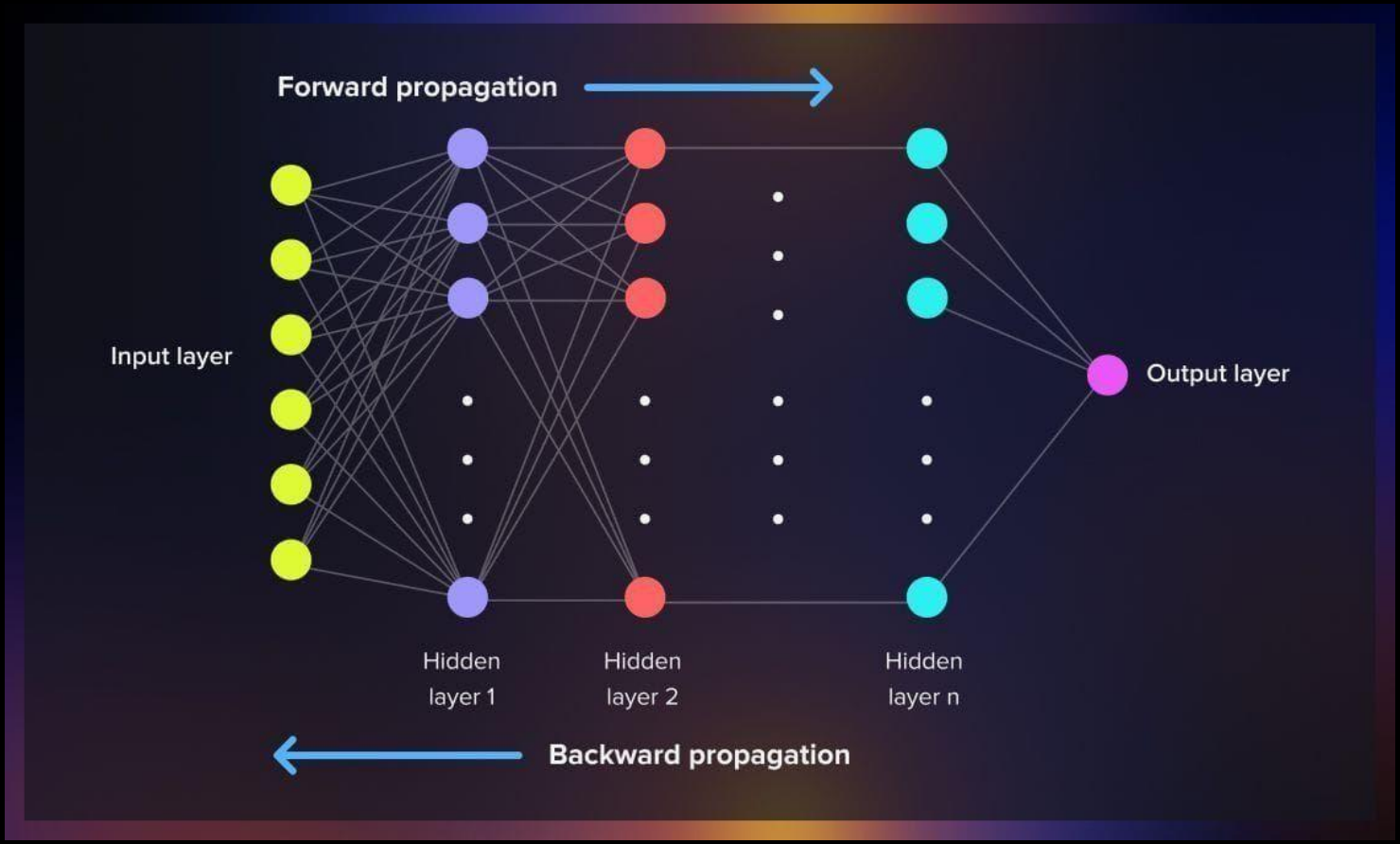
Neural Networks from Scratch in Python

- https://nnfs.io/order
- https://github.com/Sentdex/nnfs_book/tree/main



Today’s notes and coding

How do NN do forward propagation?



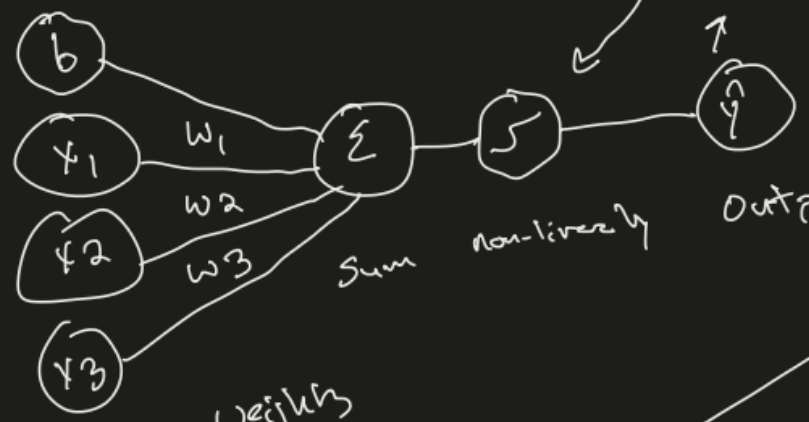
Next class: How do NN do backward propagation?

01 perceptron and layers

Friday, January 2, 2026 10:56 AM

Perceptron

$\hat{y} = g(b + \sum_{i=1} x_i w_i)$



$\hat{y} = g(b + X^T W)$

where $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ + $W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$

What does this mean? ⇧ Review of Linear Algebra!

Dot Product

$a = [a_1, a_2, a_3]$ + $b = [b_1, b_2, b_3]$

$np.dot(a,b) = a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$

$= np.dot(b,a)$

This is a neuron ⇧

Vector + Matrix

$a = [1,2,3]$ + $b = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$

3x1

What is the rule of matrix multiplication?

$np.dot(a,b) = [1,2,3] \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$ 3x3 (b_1 b_2 b_3)

1x3 · 3x3

must be the same!

So output will be 1x3

$= \begin{bmatrix} np.dot(a,b_1), np.dot(a,b_2) \\ np.dot(a,b_3) \end{bmatrix}$

$= [4+14+30=48, 54, 60]$

However, np.dot(a,b) ≠ np.dot(b,a)

$np.dot[b,a] = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} [1,2,3]$

3x3 1x3

Can someone explain transposing?

Transpose a^T

$a^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

$= \begin{bmatrix} np.dot(a^T, b_1) \\ np.dot(a^T, b_2) \\ np.dot(a^T, b_3) \end{bmatrix}$

$= \begin{bmatrix} 32 \\ 50 \\ 68 \end{bmatrix}$

These are Layers

Matrix x Matrix

$$a = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad \begin{matrix} A_1 \\ A_2 \\ A_3 \end{matrix}$$

3×3

$$b = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \quad \begin{matrix} B_1 & B_2 & B_3 \end{matrix}$$

3×3

$np.dot(a, b) = 3 \times 3 \cdot 3 \times 3$

$$= \begin{bmatrix} A_1B_1 & A_1B_2 & A_1B_3 \\ A_2B_1 & A_2B_2 & A_2B_3 \\ A_3B_1 & A_3B_2 & A_3B_3 \end{bmatrix}$$

Why might this be useful? Why would We use this

This is Batching!

Go to next notes on layers

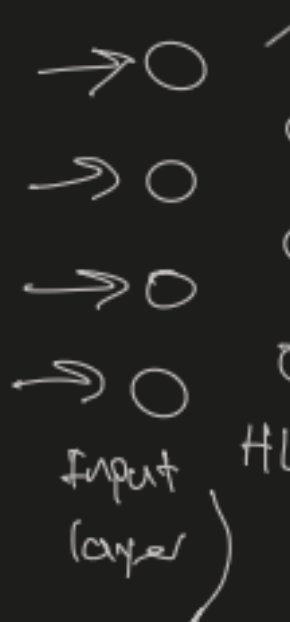
Adding more layers

$$X = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

3x4

output = np.dot(X, W_1^T) + B_1

output = np.dot(f_1, W_2^T) + B_2



What shape will the weights be?

What shape will these weights be?

$$W_1 = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

3x4

$$W_2 \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ \vdots & \vdots & \vdots \end{bmatrix}$$

3x3

$$B_1 = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

So, then

$$output_1 = np.dot(x, w_1^T) + B_1$$

$$output_2 = np.dot(f_1, w_2^T) + B_2$$

$$\vdots$$

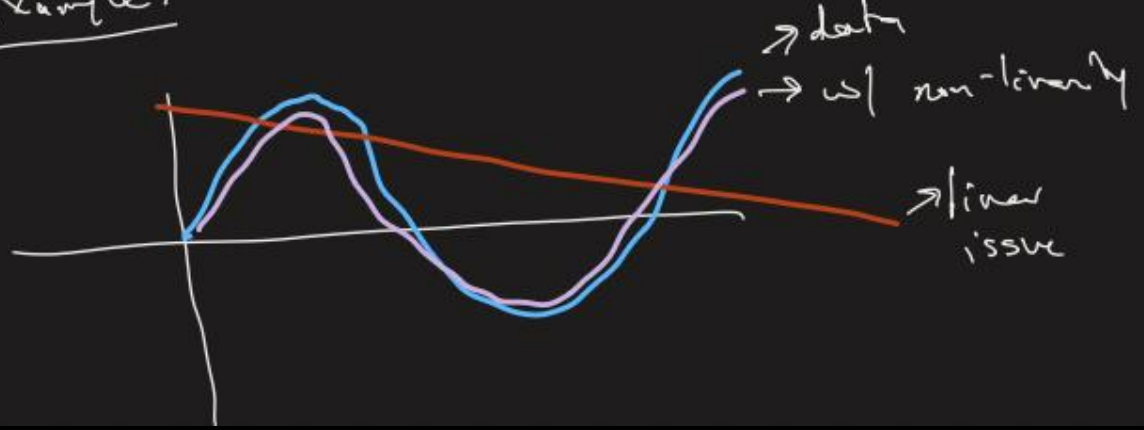
$$output_2 = np.dot(f_{2-1}, w_2^T) + B_2$$

Activation functions

input sum AF

Activation functions add non-linearity. Added every layer, Stacking them compounds their effect to learn more complex functions

Example:



ReLU

$y = \max(0, x)$

Sigmoid

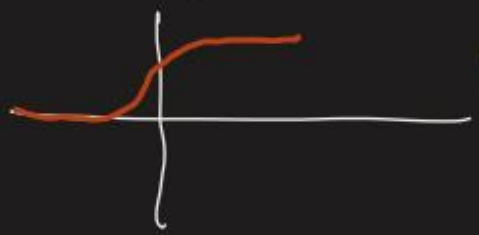
$y = 1 / (1 + e^{-x})$

Binary

Softmax

$$\sigma(z)_i = e^{z_i} / \sum_{j=1}^{K} e^{z_j}$$

K = classes

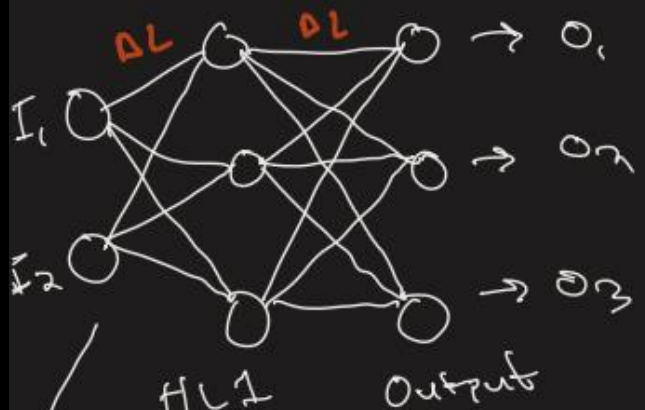


multi-class

Show video
https://nnfs.io/mvp/

Watch Video on Python notebook

A full forward pass



Think of this as class prediction probabilities

Layer_Dense → ReLU → Layer_Dense → Softmax

Ex. Predict flower based on length & width

$f_1 = [10, 5]$ $f_2 = [20, 10]$

c_1 = rose
c_2 = daisy
c_3 = tulip

← so is c_2

But that isn't a super strong prob. So how do we train the NN to predict this?

Loss function

What is the purpose of LF?

$f_1 = [10, 5]$ $f_2 = [20, 10]$

⇒ this isn't great

Categorical cross entropy loss

$$CE = -\sum \text{true} \times \log(\text{predicted})$$

ex. pred: $\{.2, .6, .2\}$
true: $\{0, 1, 0\}$

then

$$CE = -(0 \times \log(.2) + 1 \times \log(.6) + 0 \times \log(.2))$$
$$= -\log(.6)$$
$$= .22$$

what do you notice here?
incorrect do not matter!

① Why do we take the negative of the log? - Want loss to be positive

② Why do we take the log?
- Want $\log(1) = 0$ bc there is no loss

ex. Correct prob.	$-\log(p)$	meaning
.9	0.105	good ✓
.5	0.693	uncertain
.1	2.302	bad

One hot encoding

C_1 = rose C_2 = daisy C_3 = tulip

x_1 x_2

S_1 = rose w/ 10, 5

$y = [1, 0, 0] \Rightarrow \{C_1, C_2, C_3\}$

true

x_1 x_2

S_2 = tulip w/ 20, 10

$y = [0, 0, 1]$

Mean Square Error Loss

$$MSE = \frac{1}{n} \Sigma (y - \hat{y})^2$$

This is for non-categorical answers
ex. What age are you?

Input: height + weight

predicted: 21 true: 26

$$MSE = \frac{1}{1} * \Sigma (26-21)^2$$
$$= 1 * \Sigma 5^2$$
$$MSE = 25$$

if it was 22 instead

$$MSE = \frac{1}{1} * \Sigma (22-21)^2$$
$$= 1 * 1^2$$
$$MSE = 2$$

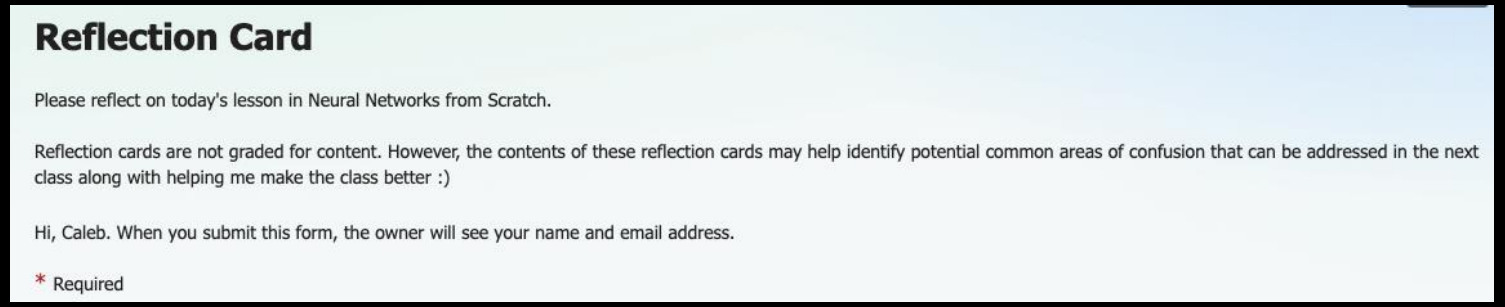
There are a few more loss functions as well!

A Neural Network Playground

https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.89302&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

Will show this before and after code

Reflection Cards



- Essentially a means to help me make this class better!

1. What is something that you learned in today's lecture?
2. What is something that you are still confused about from today's lecture?
3. Do you have any other comments/feedback/thoughts/suggestions/concerns?

https://forms.office.com/r/Kv8LtW4iJH