Scientific Computing

April 23, 2025

Announcements

→ Homework 6 due Friday, May 2, 11:59pm

→ Final exam is take-home only

assigned Fri, May 2

due Fri, May 9

Today

→ Coding NN and Batching

→ Loss Functions

Office Hours:

Mon + Fri

9:30am - 10:30am

Cudahy 307

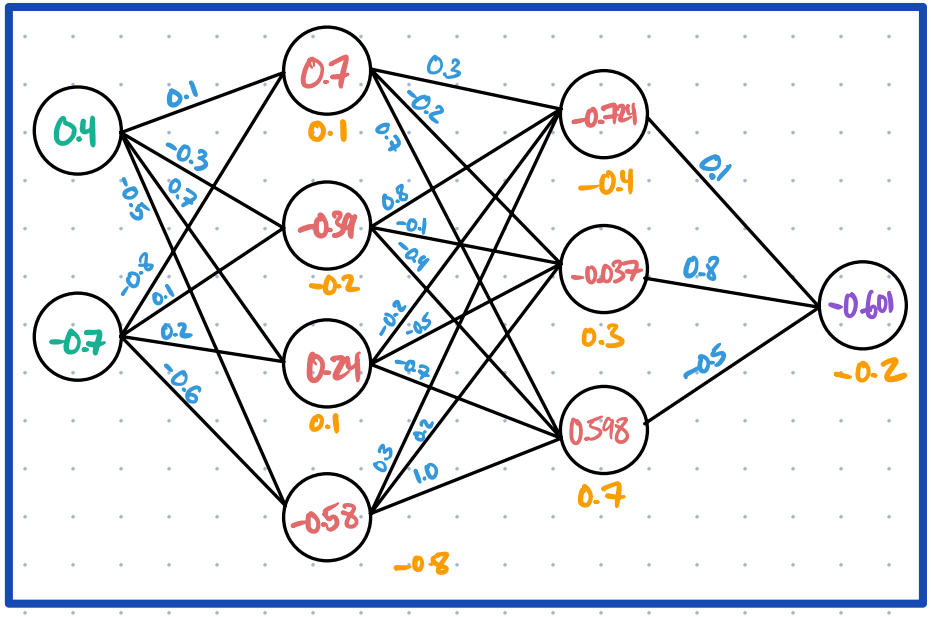
Think of each layer as a vector of values.

$$\begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix} \quad \begin{bmatrix} 0.7 \\ -0.39 \\ 0.24 \\ -0.58 \end{bmatrix} \quad \begin{bmatrix} -0.724 \\ -0.037 \\ 0.598 \end{bmatrix} \quad \begin{bmatrix} -0.601 \end{bmatrix}$$

How is each layer computed from the previous one?
Make matrices for the weights between each layer and vectors for the biases.

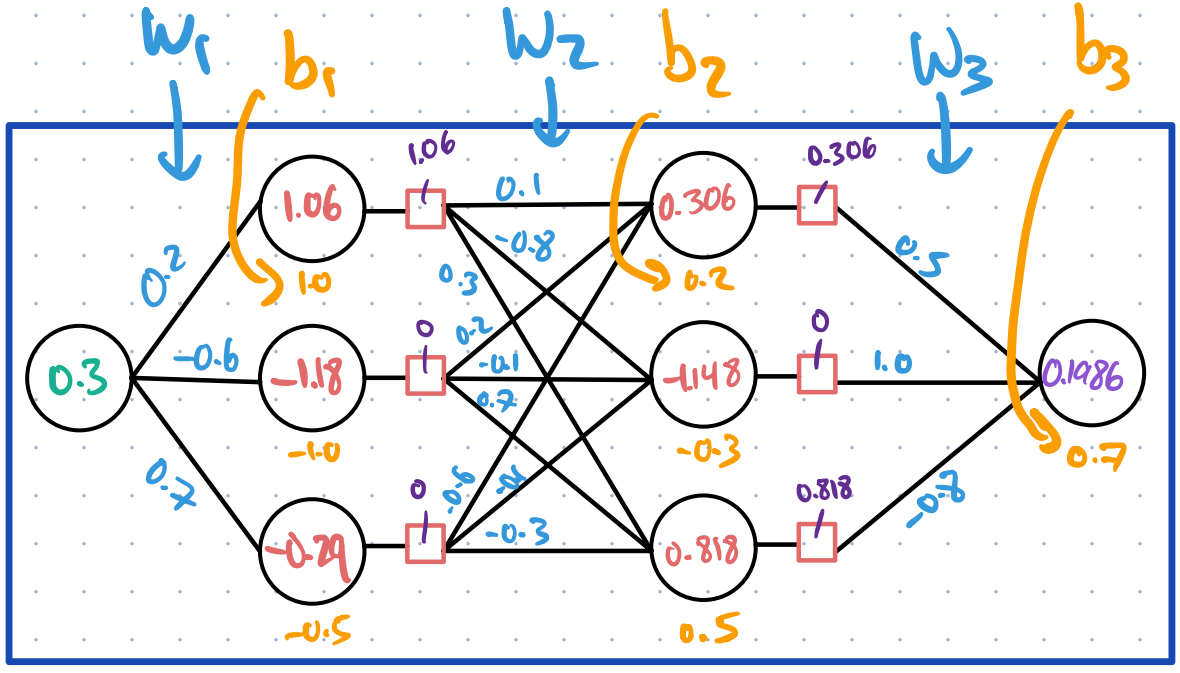
$$M_1 = \begin{bmatrix} 0.1 & -0.8 \\ -0.3 & 0.1 \\ 0.7 & 0.2 \\ -0.5 & -0.6 \end{bmatrix} \qquad b_1 = \begin{bmatrix} 0.1 \\ -0.2 \\ 0.1 \\ -0.8 \end{bmatrix}$$

$$\begin{bmatrix} 0.1 & -0.3 \\ -0.8 & 0.1 \end{bmatrix} \cdots$$



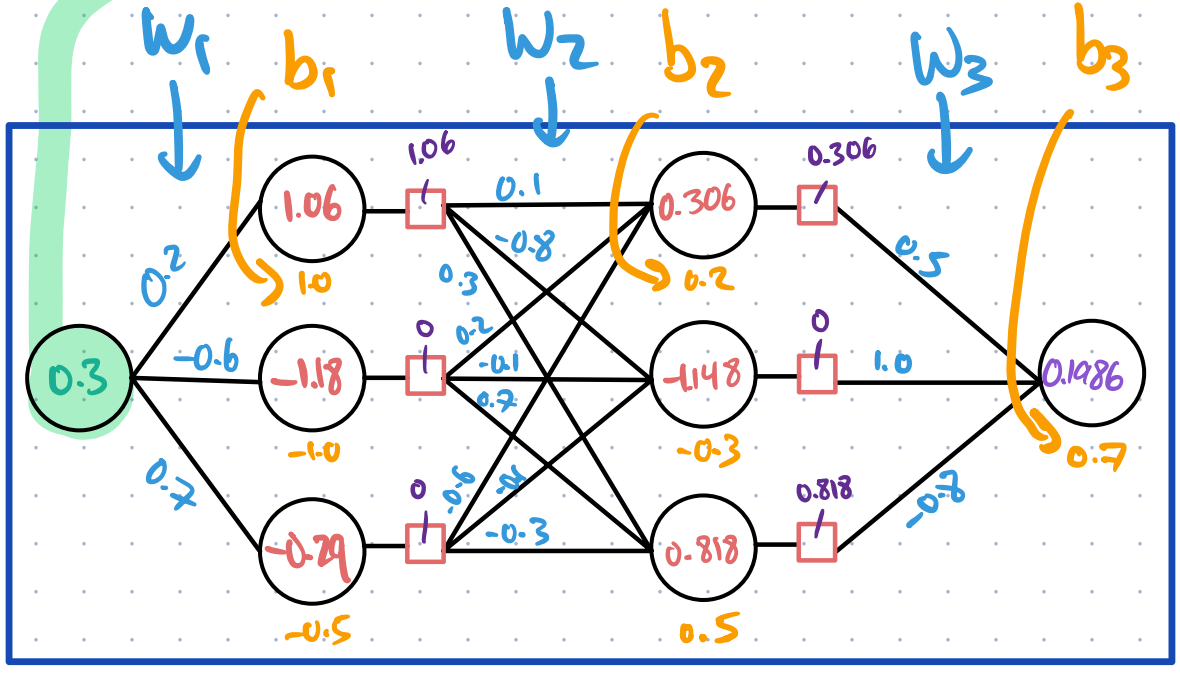
So this neural network, with activation function σ for each layer, is the function:

$$f(x) = \sigma(W_3\sigma(W_2\sigma(W_1[x] + b_1) + b_2) + b_3)$$



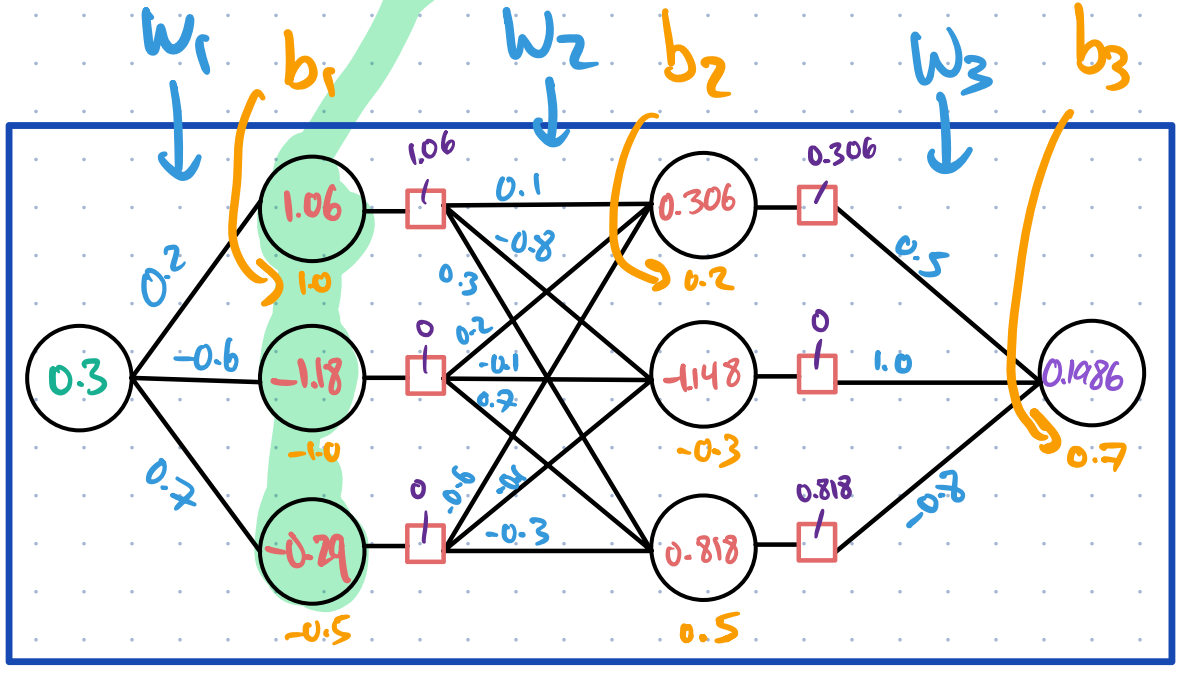
So this neural network, with activation function σ for each layer, is the function:

$$f(x) = \sigma(W_3\sigma(W_2\sigma(W_1[x] + b_1) + b_2) + b_3)$$



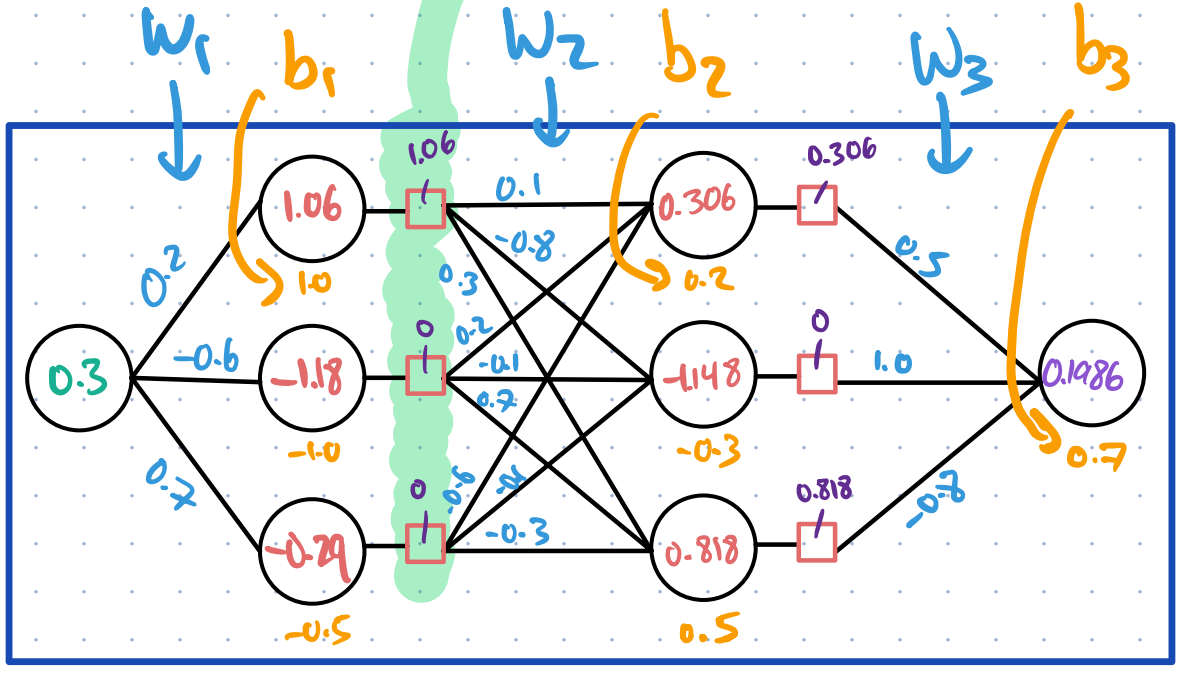
So this neural network, with activation function σ for each layer, is the function:

$$f(x) = \sigma(W_3\sigma(W_2\sigma(W_1[x] + b_1) + b_2) + b_3)$$



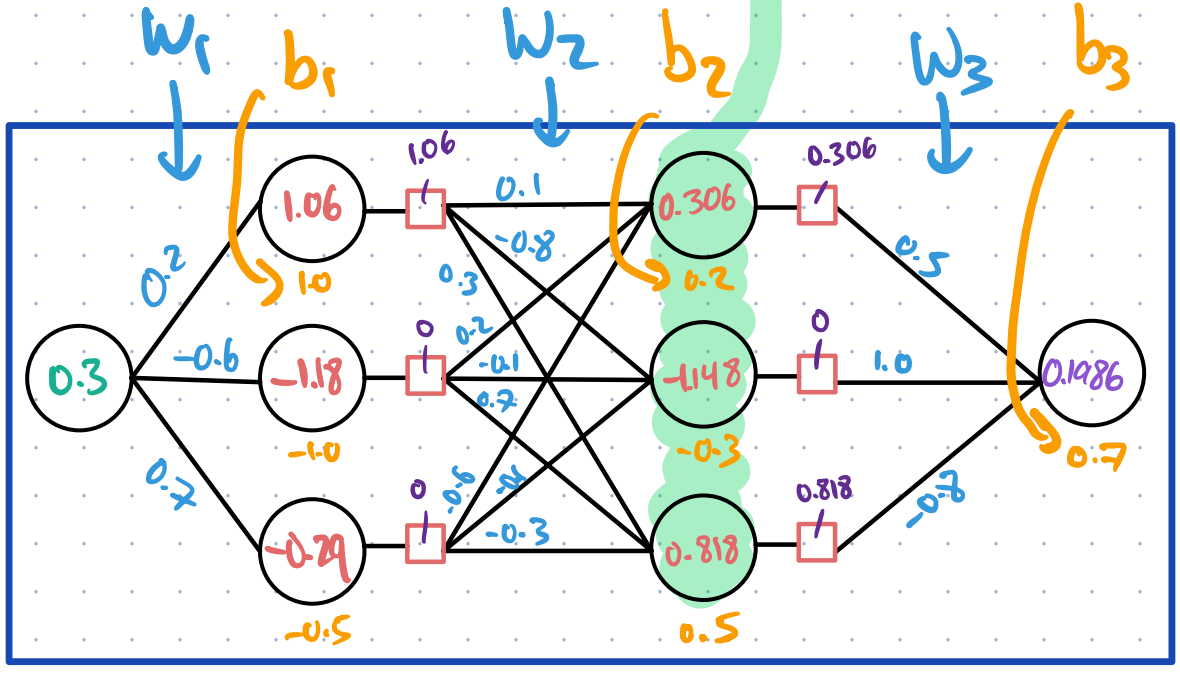
So this neural network, with activation function σ for each layer, is the function:

$$f(x) = \sigma(W_3\sigma(W_2\sigma(W_1[x] + b_1) + b_2) + b_3)$$



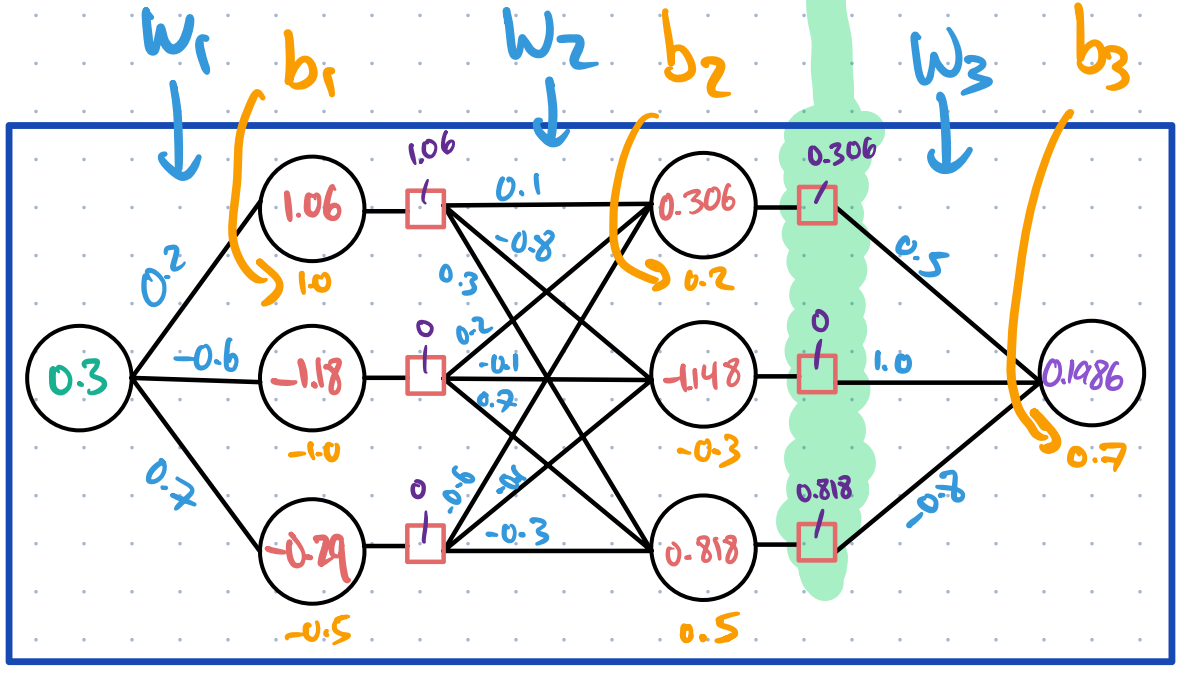
So this neural network, with activation function σ for each layer, is the function:

$$f(x) = \sigma(W_3 \sigma(W_2 \sigma(W_1[x] + b_1) + b_2) + b_3)$$



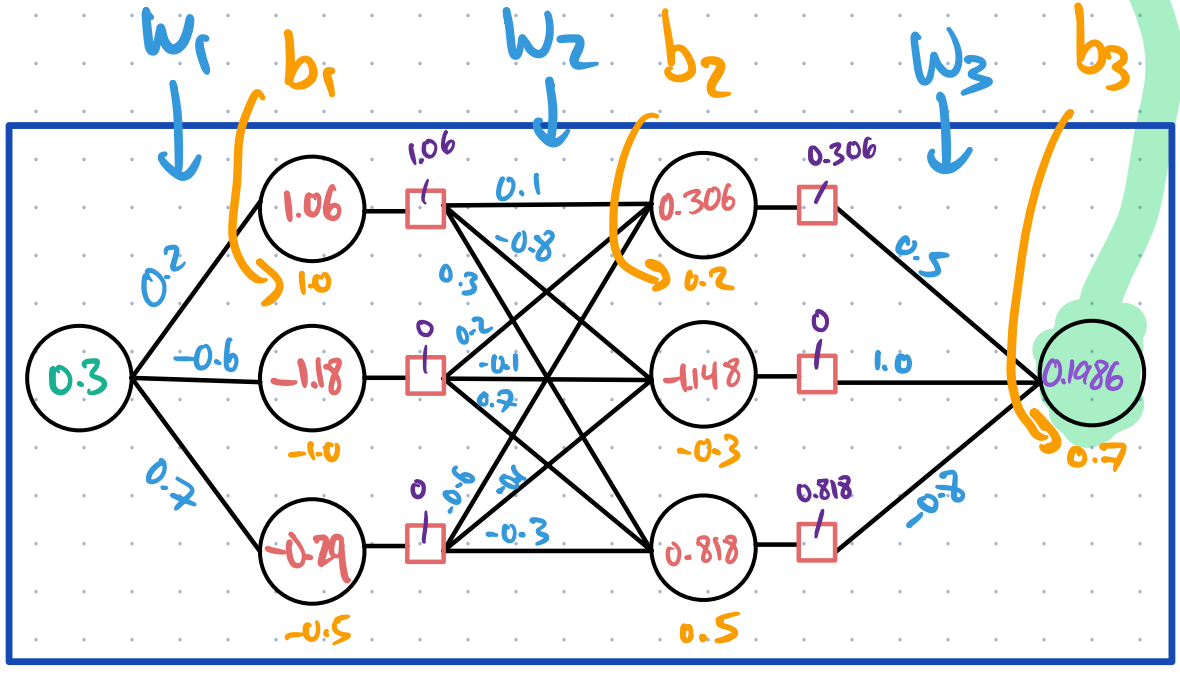
So this neural network, with activation function σ for each layer, is the function:

$$f(x) = \sigma(W_3 \sigma(W_2 \sigma(W_1[x] + b_1) + b_2) + b_3)$$



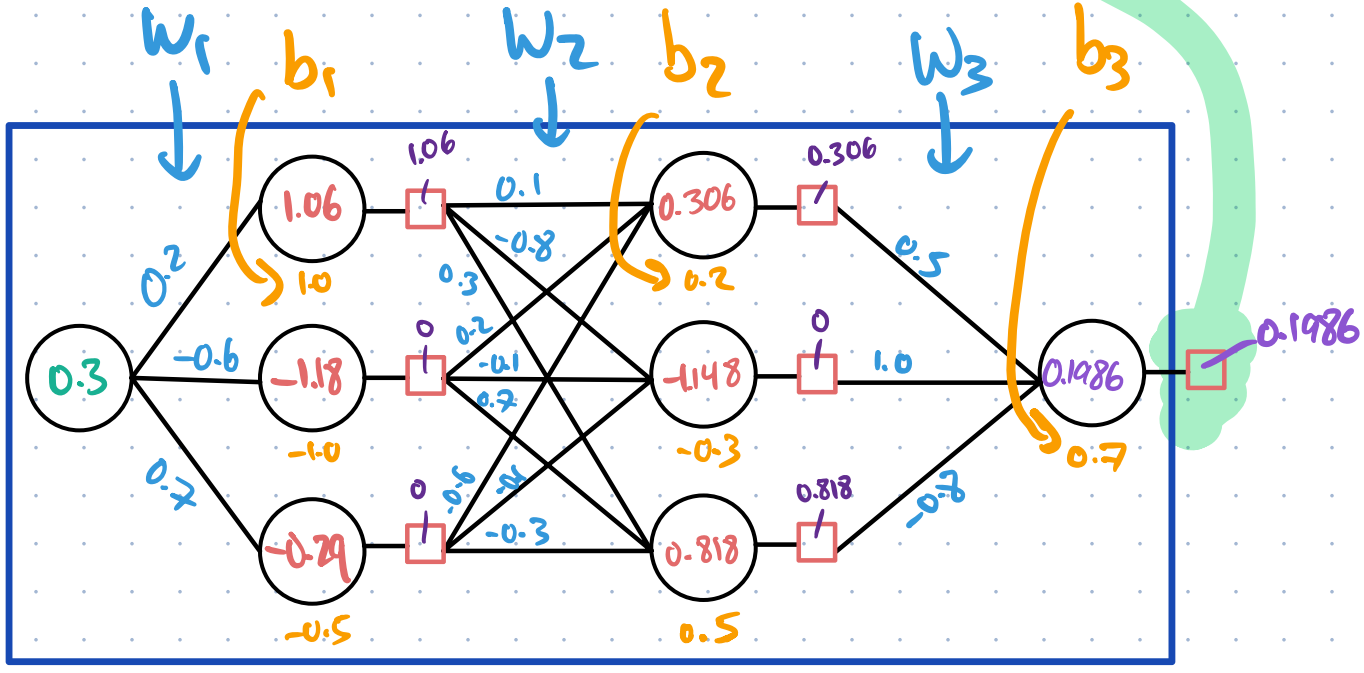
So this neural network, with activation function σ for each layer, is the function:

$$f(x) = \sigma(W_3\sigma(W_2\sigma(W_1[x] + b_1) + b_2) + b_3)$$



So this neural network, with activation function σ for each layer, is the function:

$$f(x) = \sigma(W_3\sigma(W_2\sigma(W_1[x] + b_1) + b_2) + b_3)$$



* Sometimes the output layer has a different activation function, or no AF.

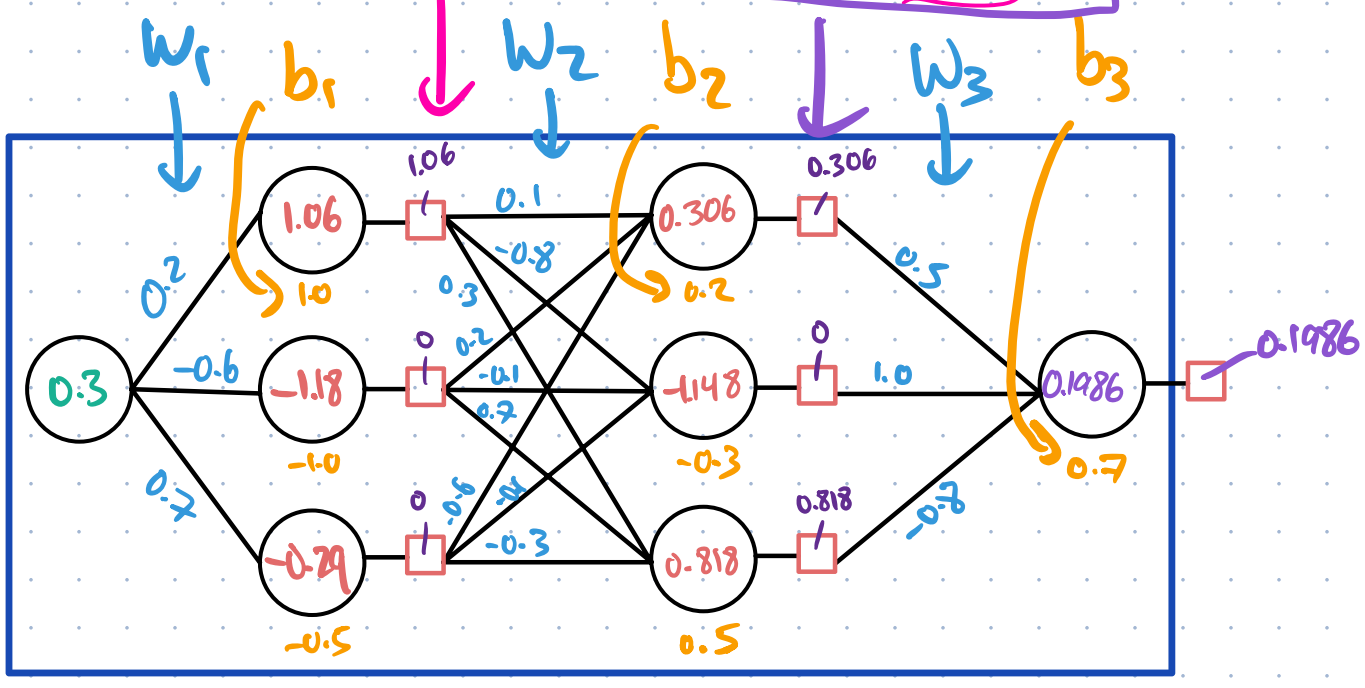
$$W_1 = \begin{bmatrix} 0.2 \\ -0.6 \\ 0.7 \end{bmatrix} \quad W_2 = \begin{bmatrix} 0.1 & -0.8 & 0.3 \\ 0.2 & -0.1 & 0.7 \\ -0.6 & -0.1 & -0.3 \end{bmatrix} \quad W_3 = \begin{bmatrix} 0.5 & 1.0 & -0.8 \end{bmatrix}$$

$$b_1 = \begin{bmatrix} 1.0 \\ -1.0 \\ -0.5 \end{bmatrix}, \quad b_2 = \begin{bmatrix} 0.2 \\ -0.3 \\ 0.5 \end{bmatrix}, \quad b_3 = \begin{bmatrix} 0.7 \end{bmatrix}$$

each row of a W is the weights coming out of one neuron

$$f(0.3) = \max\left(0, \begin{bmatrix} 0.5 & 1.0 & -0.8 \end{bmatrix} \max\left(0, \begin{bmatrix} 0.1 & -0.8 & 0.3 \\ 0.2 & -0.1 & 0.7 \\ -0.6 & -0.1 & -0.3 \end{bmatrix} \max\left(0, \begin{bmatrix} 0.2 \\ -0.6 \\ 0.7 \end{bmatrix} [0.3] + \begin{bmatrix} 1.0 \\ -1.0 \\ -0.5 \end{bmatrix}\right) + \begin{bmatrix} 0.2 \\ -0.3 \\ 0.5 \end{bmatrix}\right) + [0.7]\right)$$

$$= 0.1986$$



Now you understand exactly what a neural network is:

a fancy way to define a function

But not yet:

* How to find a good one (training)
* How NNs accomplish different tasks (what good is a fancy function?)

Next topic: Coding our first NN, and batching the forward pass

Topic 14 - Implementing our first NN and Batching

* Goal: Write code to perform the forward pass of a NN, with activation functions

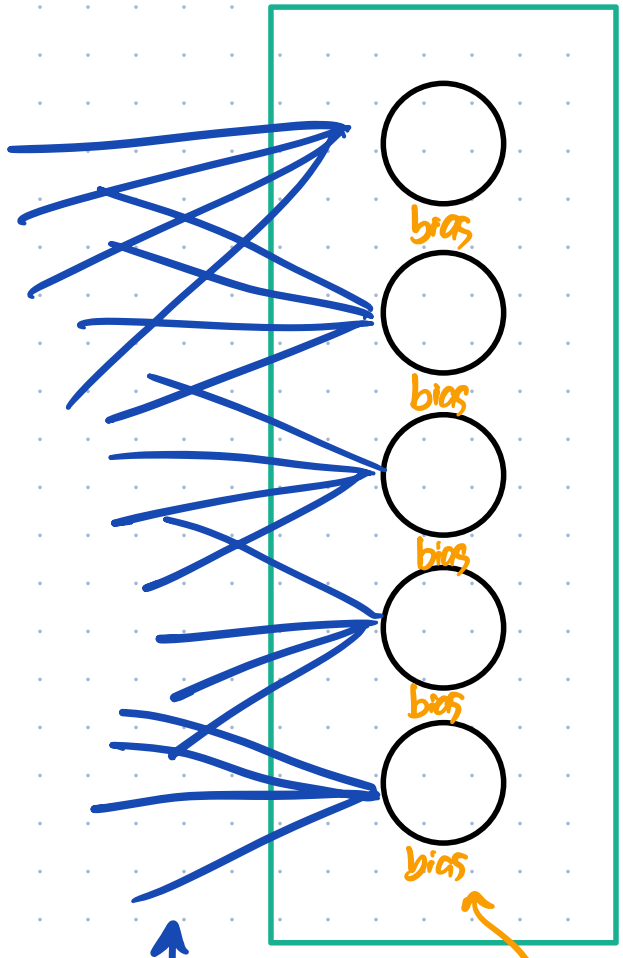
The purpose of writing this code is to understand the topic.

If you needed a fast NN for research, you would probably use a Python library like PyTorch

Structure:

* Object Oriented
* Objects:
    - Layer
        - knows the weights that feed into it from the previous layer
        - knows the biases of its neurons
        - can take input data and compute output data
    - ActivationFunction
        - can take input data and compute output data

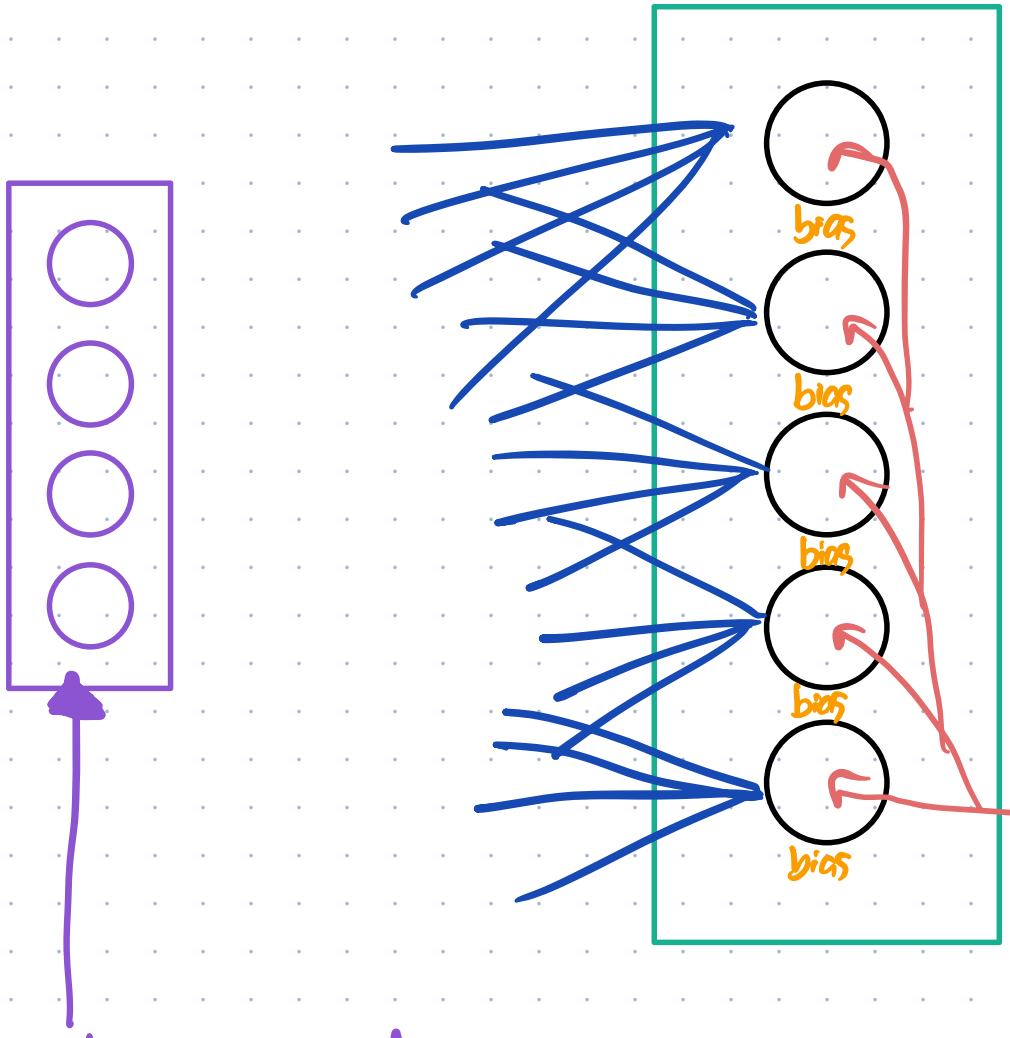
Layer:



knows the biases (vector)

knows the weights on these edges (matrix)

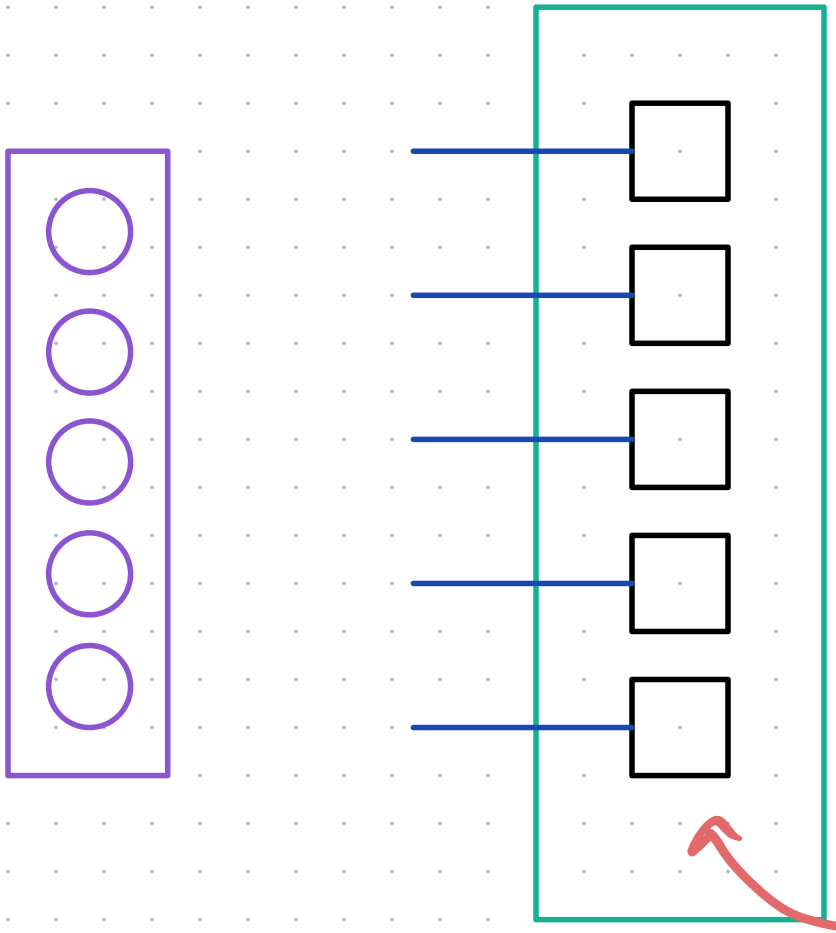
Layer:



input: value of previous layer's neurons
(actually, the activation function of those neurons)

output: value of this layer's neurons (pre-activation)

Activation Function classes (one for each activation function)

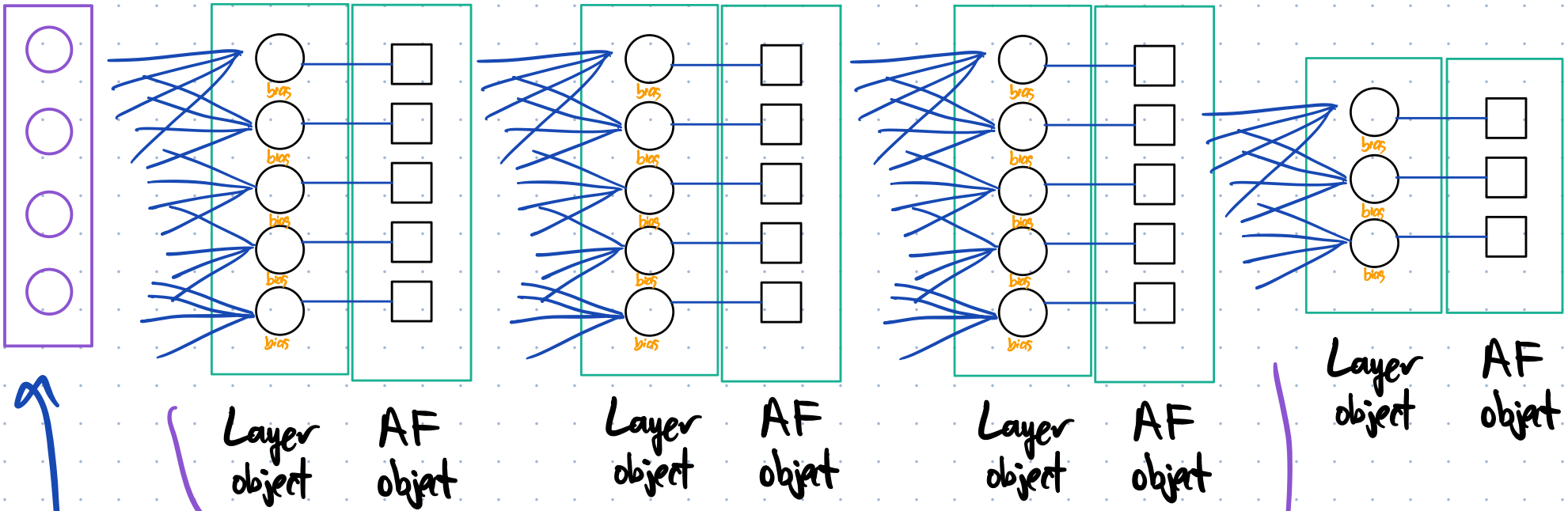


$\max(0, v)$

inputs: values of neurons we're activating

outputs: activated values

Then we can chain instances of these objects together to make a full NN.



Coding time:

First: the "numpy" Python library

* Jupyter notebook demo

Coding time:

First: the "numpy" Python library

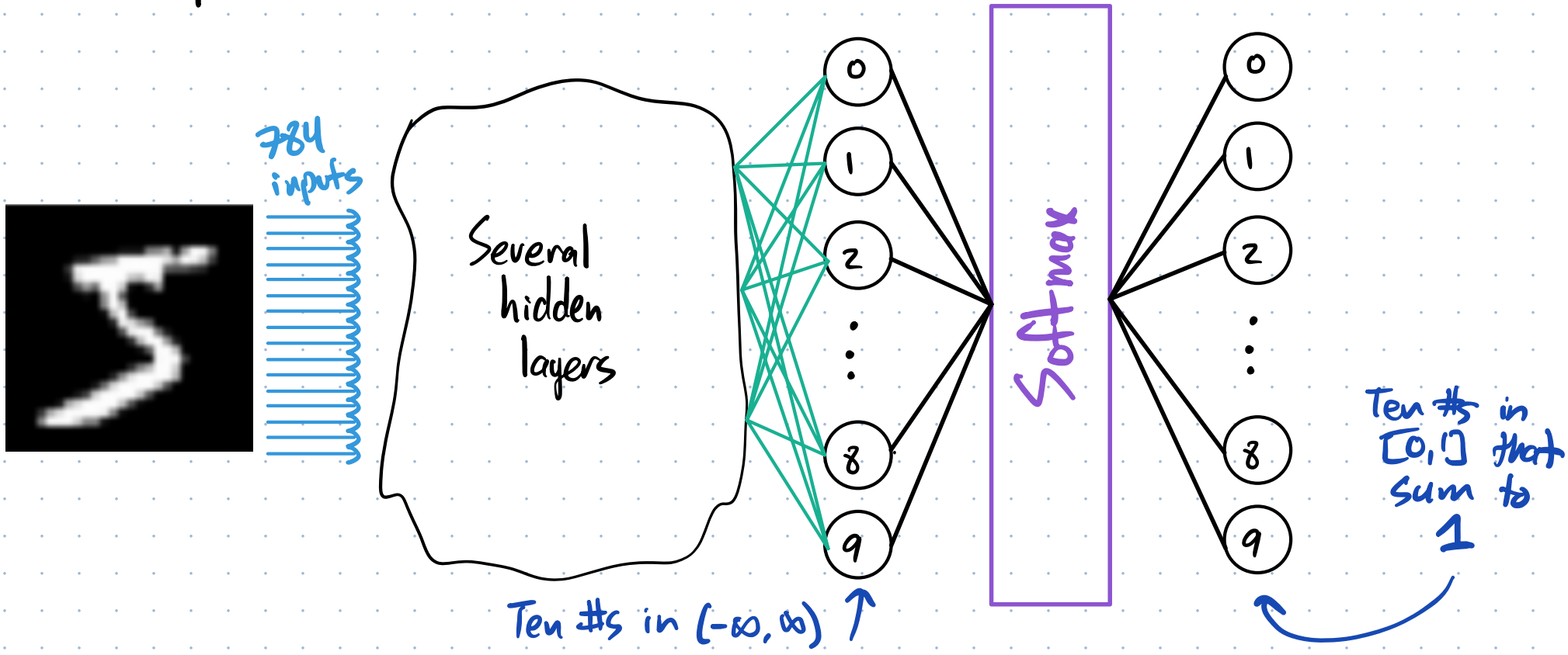
* Jupyter notebook demo

Next: Coding our first layers together from scratch.

(very heavy inspiration from the "Neural Networks from Scratch" book!)

New Activation Function: Softmax

* Turns a vector of #s into a probability distribution
  (a vector of #s in [0,1] that sums to 1)
* Useful for the output layer in a classification problem.



Unlike our other activation functions, Softmax works on the whole vector at once, not one value at a time individually.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_k \end{bmatrix} \xrightarrow{\text{softmax}} \begin{bmatrix} e^{x_1}/S \\ e^{x_2}/S \\ e^{x_3}/S \\ \vdots \\ e^{x_k}/S \end{bmatrix}$$

where $S = \sum_{i=1}^{k} e^{x_i}$

Obvious these add up to 1 because the denominator is their sum.

Obviously > 0 because $e^x > 0$.

Ex:

0	-0.8
1	-0.7
2	0.3
3	0.1
4	0.8
5	0.5
6	0.2
7	-0.3
8	0
9	0.6

softmax →

$\sum_{i=0}^{9} e^{x_i} \approx 12.06$

0	0.037
1	0.041
2	0.111
3	0.091
4	0.184
5	0.136
6	0.101
7	0.061
8	0.082
9	0.150

* 18.4% chance the digit is a 4

Numeric Stability (because e^x gets big!)

Let $m = \max(\vec{x})$

Then do $\dfrac{e^{x_i - m}}{\sum_{j=1}^{k} e^{x_j - m}}$ all components ≤ 0

$$\frac{e^{x_i - m}}{\sum_{j=1}^{k} e^{x_j - m}} = \frac{\frac{e^{x_i}}{e^m}}{\sum_{j=1}^{k} \frac{e^{x_j}}{e^m}} = \frac{\frac{1}{e^m} \cdot e^{x_i}}{\frac{1}{e^m} \cdot \sum_{j=1}^{k} e^{x_j}}$$

* Let's add this as a new Activation Function class