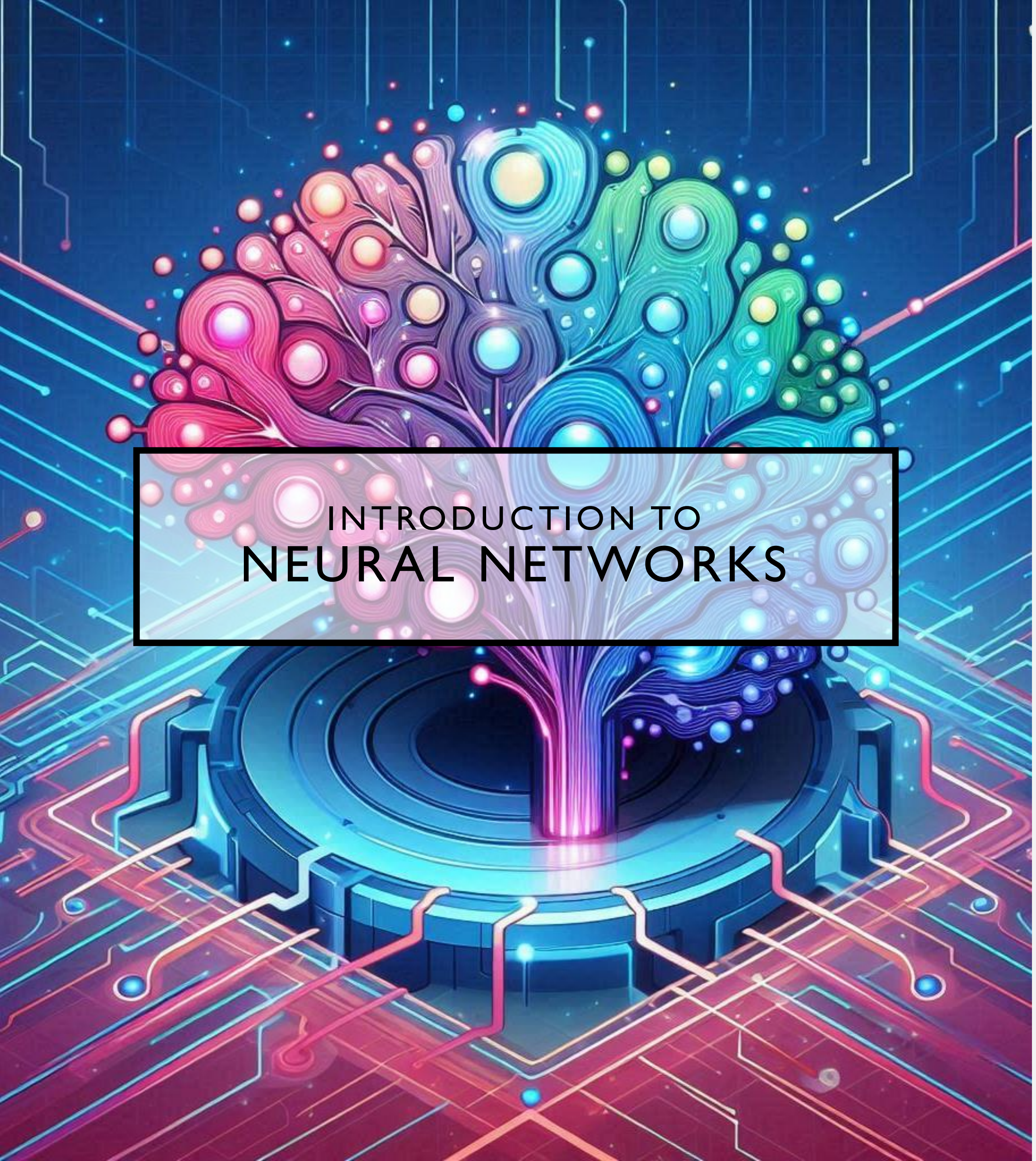
INTRODUCTION TO
NEURAL NETWORKS

- Lecture 11
- MAL1, 2024



INTRODUCTION TO NEURAL NETWORKS

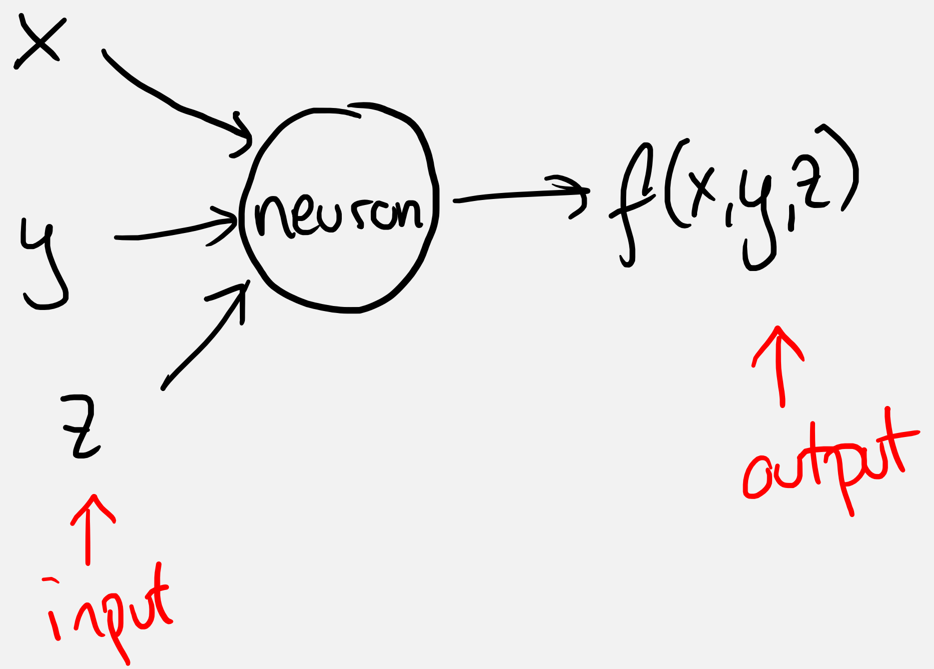
- What is a neural network?
- How do we structure it?
- How do we train it?
- How do we implement it?

NEURONS

neural networks try to mimic this idea

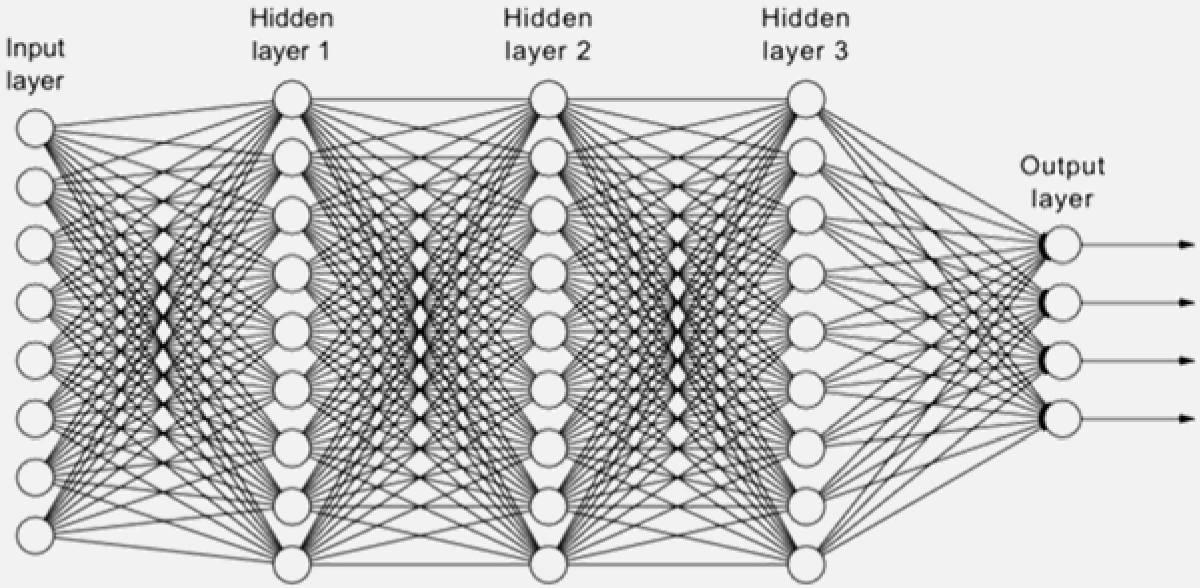
ARTIFICIAL NEURONS

ARE REALLY JUST FUNCTIONS



ARTIFICIAL NEURAL NETWORKS

attempts to mimic the brain

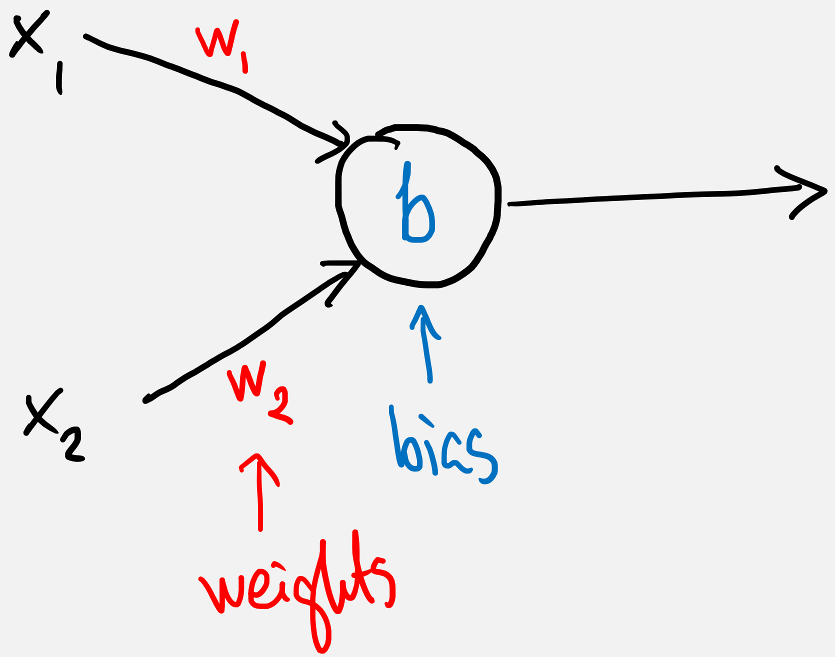


HOWEVER …

… a neural network has absolutely nothing to do with a brain.

Rather, it allows for arbitrarily complex decision boundaries/regression functions

NOTATION: WEIGHTS AND BIASES

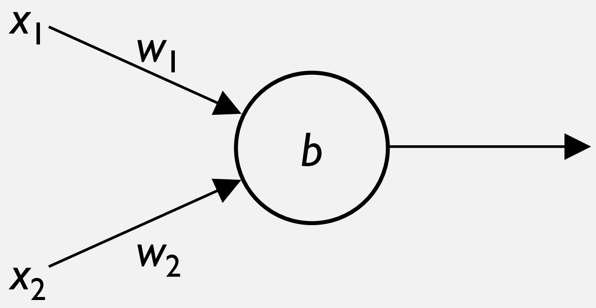


corresponds to a linear combination

$$w_1 \cdot x_1 + w_2 \cdot x_2 + b$$

$$= \bar{w} \cdot \bar{x} + b$$

PERCEPTRONS

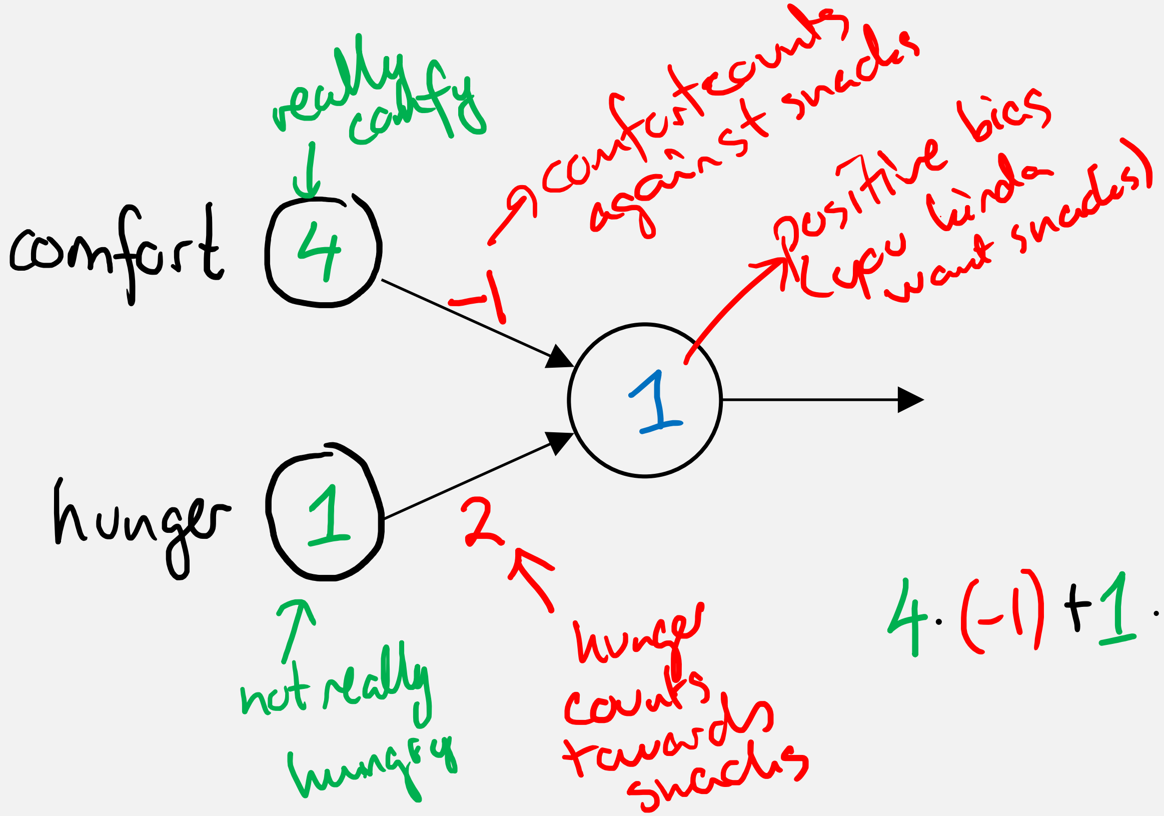


$$\text{output} = \begin{cases} 0 & \text{if } \bar{w} \cdot \bar{x} + b \leq 0 \\ 1 & \text{if } \bar{w} \cdot \bar{x} + b > 0 \end{cases}$$

THE SNACK EXAMPLE

You just sat down in the couch to watch your favorite tv show!

Is it really worth it to get up again to get some snacks?



0: stay in couch

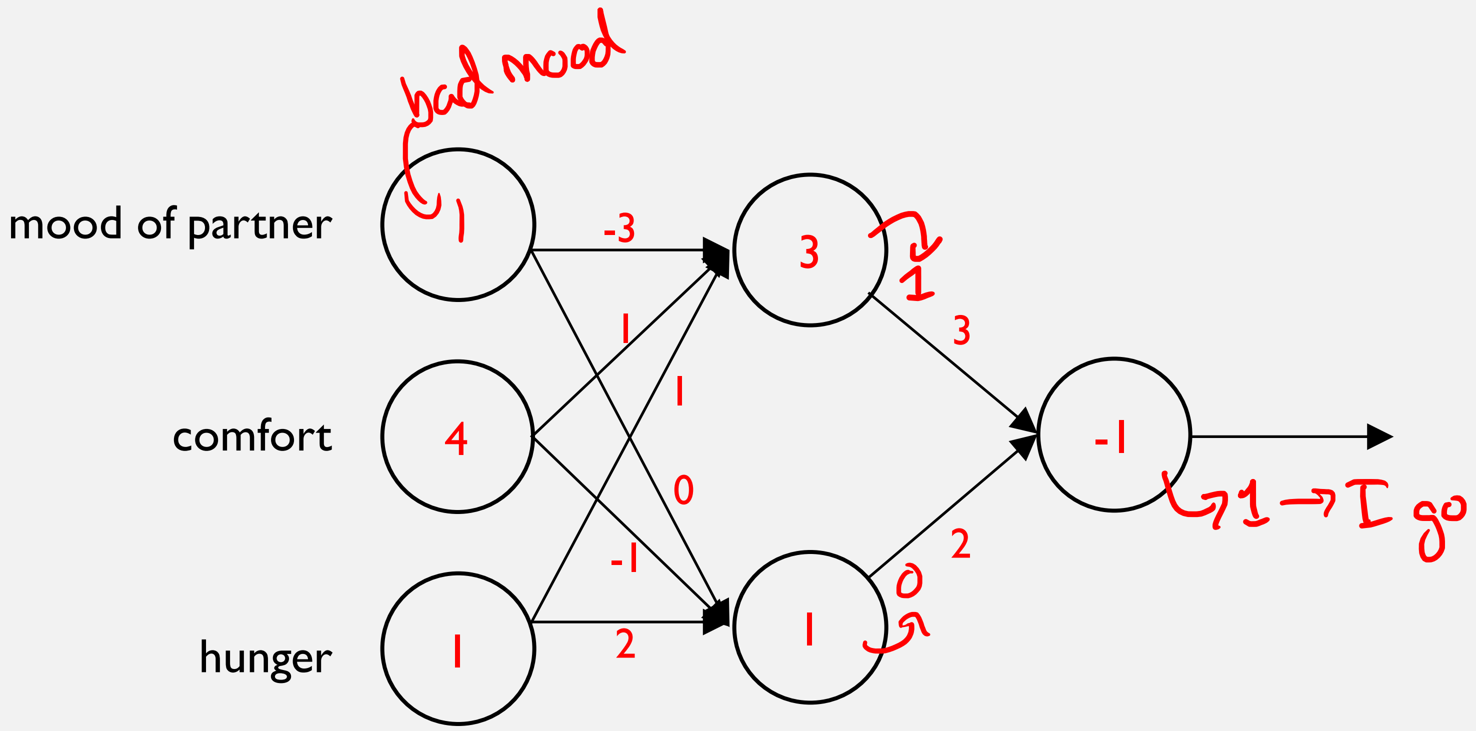
1: get snacks

$4 \cdot (-1) + 1 \cdot 2 + 1 = -1 \Rightarrow$ output 0

THE SNACK EXAMPLE II

You decide that you absolutely want snacks.

Luckily, your partner is not sitting in the couch!

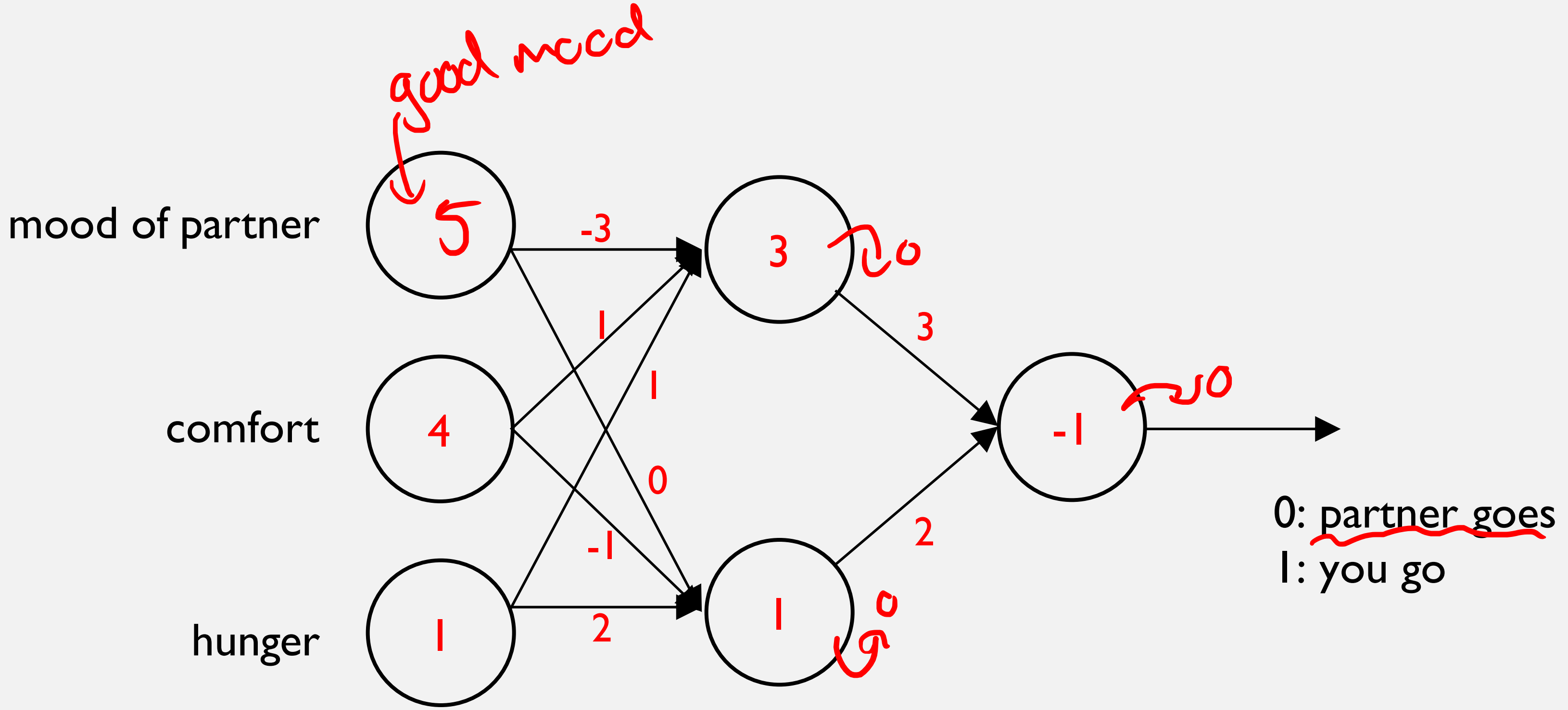


0: partner goes
1: I go

THE SNACK EXAMPLE II

You decide that you absolutely want snacks.

Luckily, your partner is not sitting in the couch!



TRAINING A NEURAL NETWORK

means finding the best set of weights & biases for a given network architechture

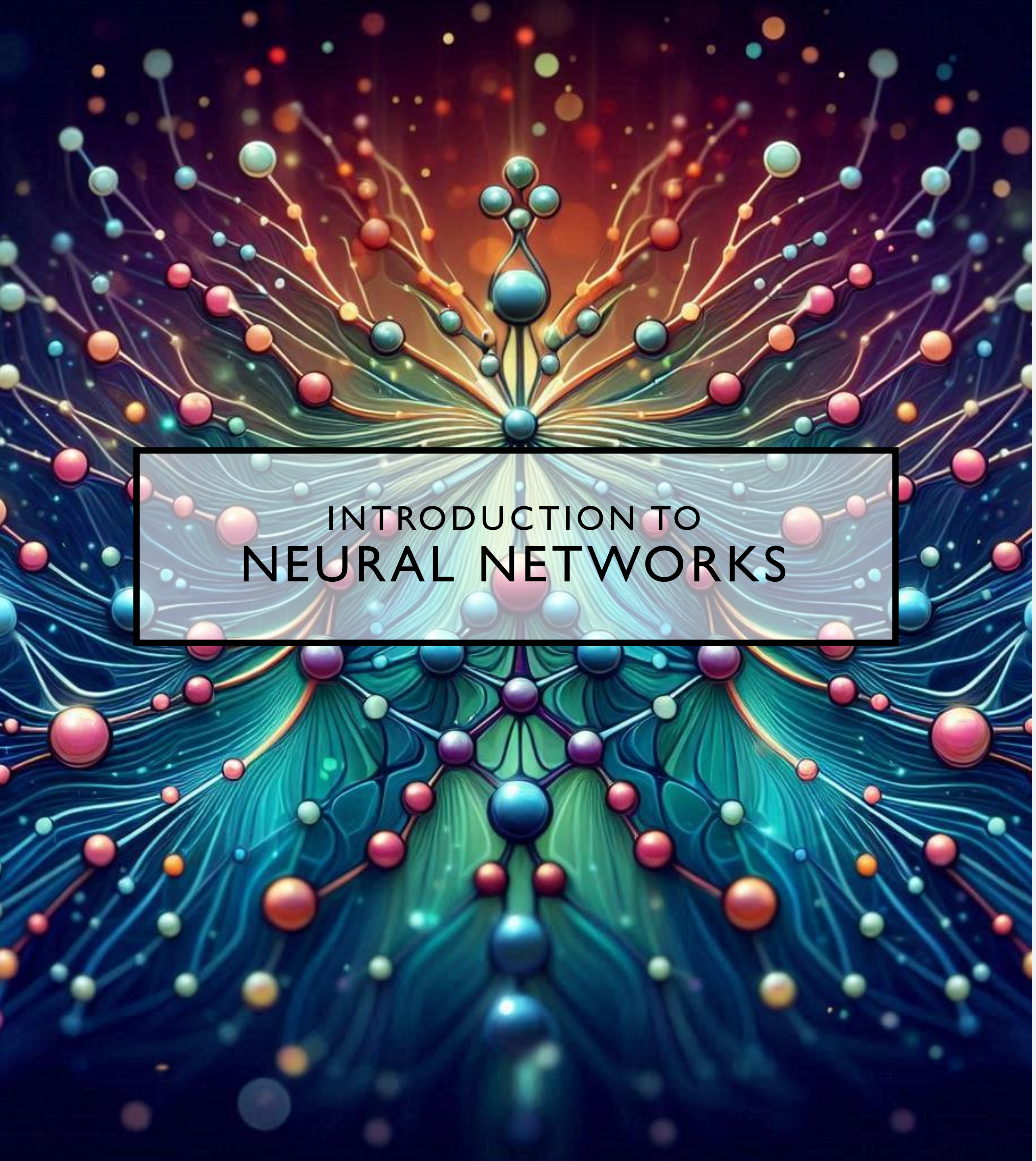
THIS LEAVES TWO QUESTIONS

1. How to structure the network

2. How to optimize weights & biases

THE TENSORFLOW PLAYGROUND

playground.tensorflow.org

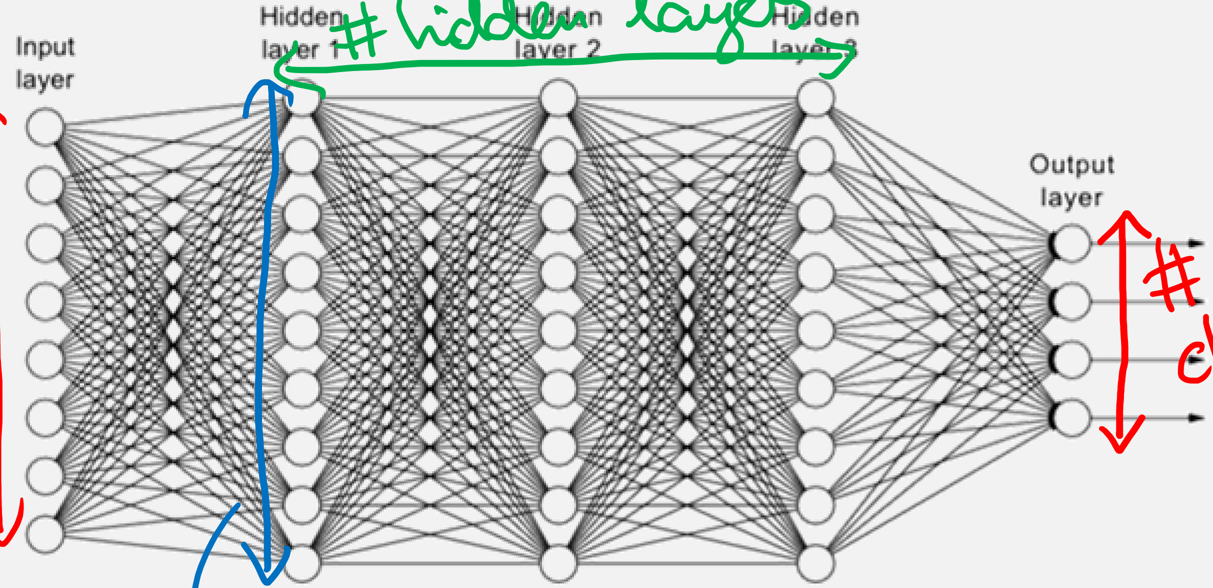


- What is a neural network?
- How do we structure it?
- How do we train it?
- How do we implement it?

FULLY-CONNECTED

HOW TO STRUCTURE THE NETWORK

hidden layers



# of features

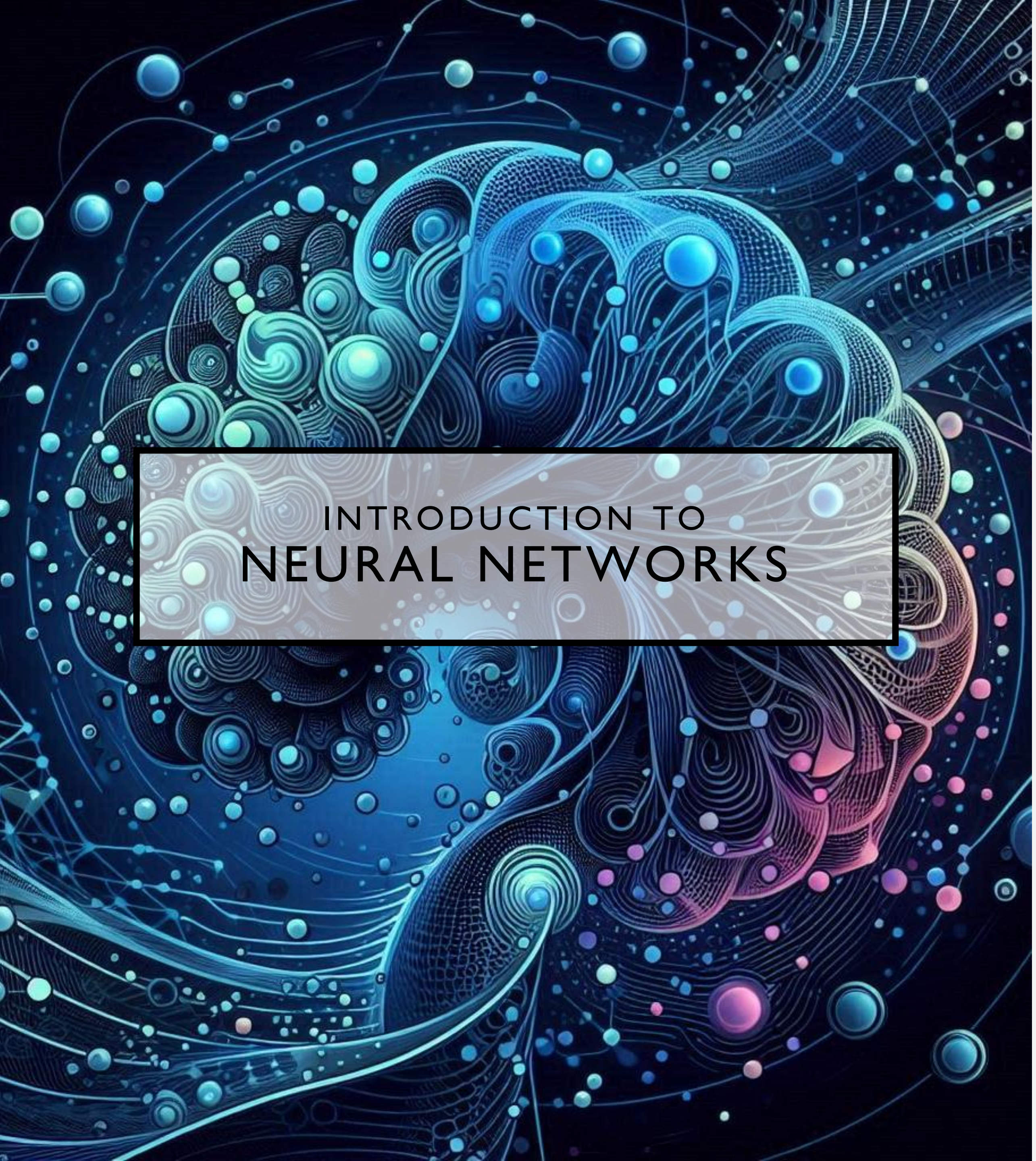
# of classes

size of hidden layer

usually one hidden layer is enough

input layer > hidden layer > output layer

Rules of thumb:



- What is a neural network?
- How do we structure it?
- How do we train it?
- How do we implement it?

HOW TO OPTIMIZE WEIGHTS AND BIASES

Minimization problem

Searching for the lowest point in a landscape

Longitude & latitude: weights & biases

Altitude: something like # of misclassifications (1 - accuracy)

"loss function"

GRADIENT DESCENT

1. Find the direction in which the descent is steepest

$$\text{gradient} \rightarrow \nabla \mathcal{L}(\beta) = \left[\frac{\partial \mathcal{L}}{\partial \beta_0}(\beta), \frac{\partial \mathcal{L}}{\partial \beta_1}(\beta), \ldots\ldots \right]$$

loss fct   coordinates

2. Take a step in that direction

$$\beta \leftarrow \beta - \eta \nabla \mathcal{L}(\beta)$$

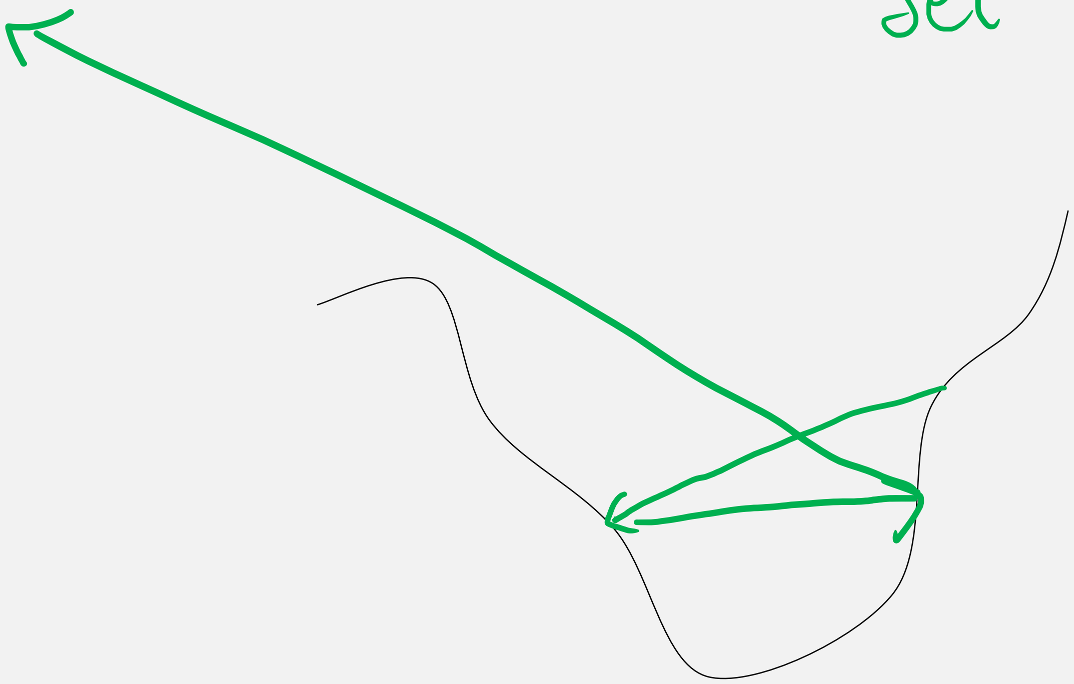
learning rate

3. Repeat until you reach the bottom

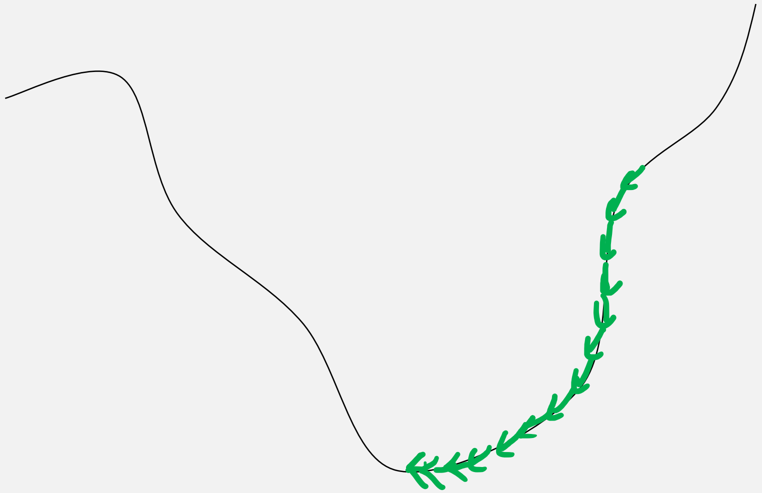
e.g. when $\nabla \mathcal{L}(\beta) = [0, 0, 0, 0, \ldots 0]$

THE LEARNING RATE

set by trial-and-error



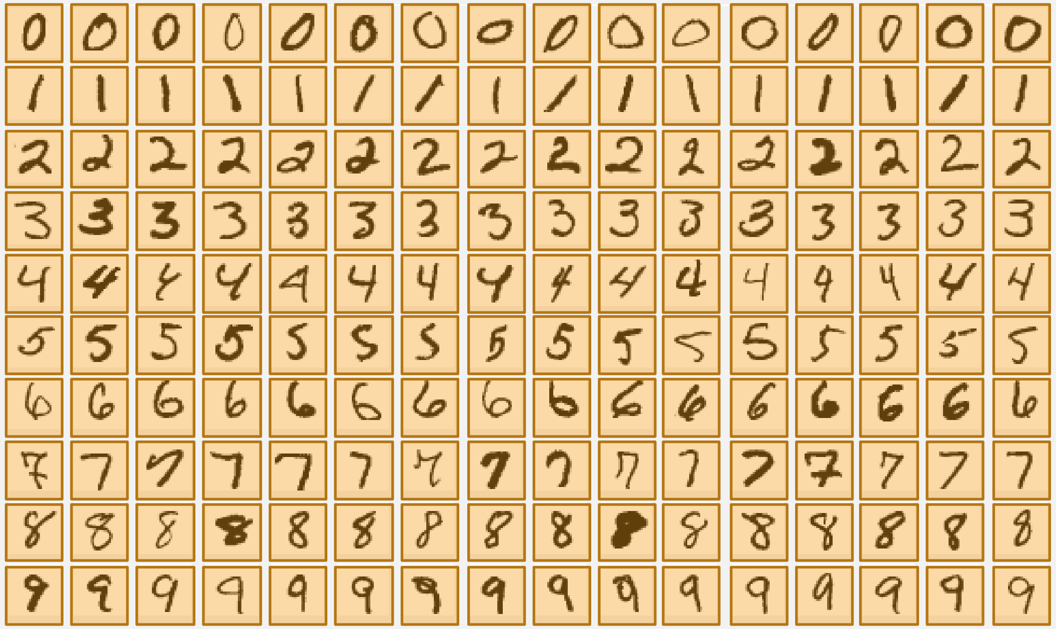
η too big



η too small

STOCHASTIC GRADIENT DESCENT

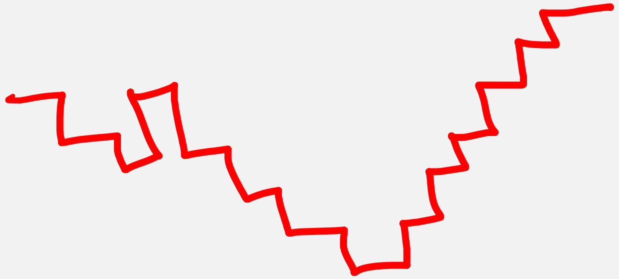
Don't update weights and biases based on **all** your data every time. Instead,

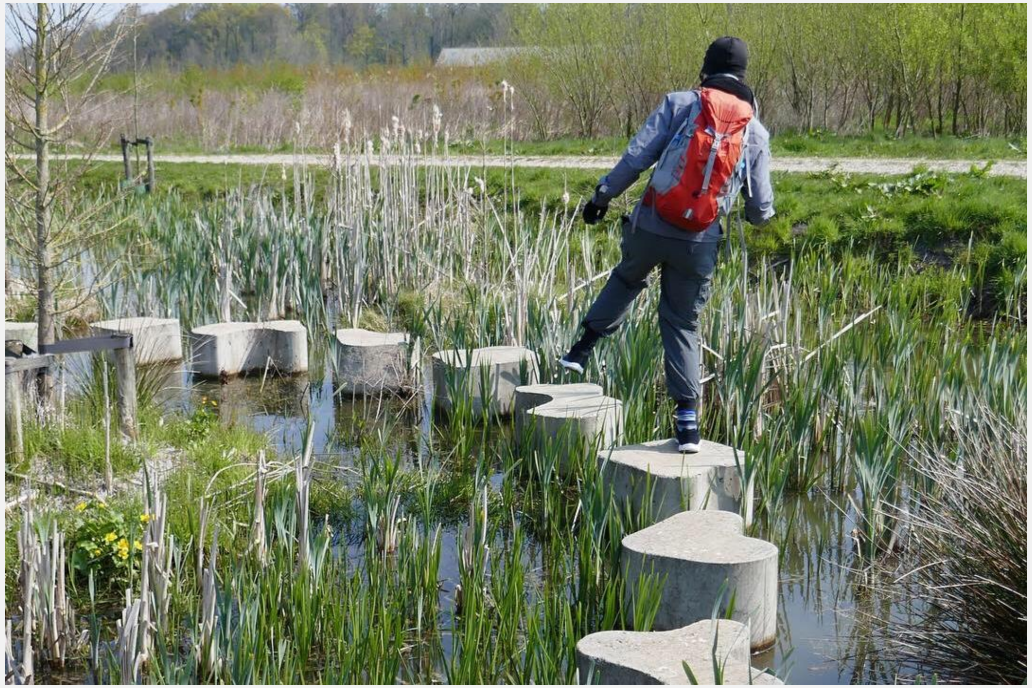


When you have gone through all your samples, you finish a **training epoch.**

BUT WHAT IF …

landscapes mostly flat
&
small step ⇒ sudden change





SUDDEN CHANGES

1. Perceptrons suddenly change output from 0 to 1

   ⇒ don't use perceptrons

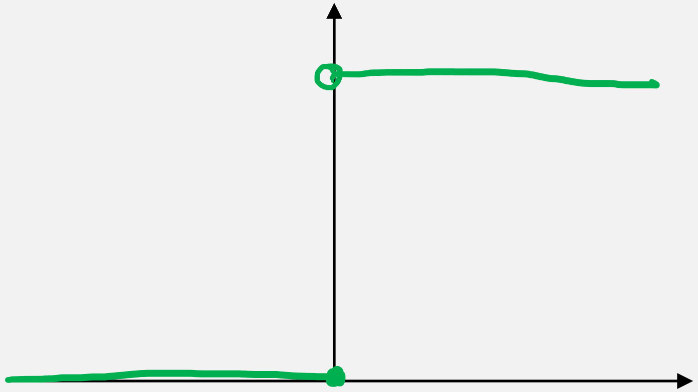
2. The number of misclassifications (1-accuracy) suddenly changes when a perceptron changes its mind

   ⇒ don't use accuracy

FIXING THE PERCEPTRON PROBLEM

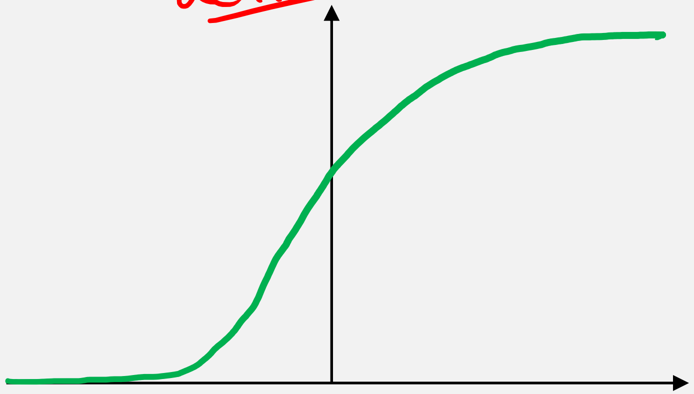
$$\text{output} = \begin{cases} 0 & \text{if} \quad \boldsymbol{wx} + b \leq 0 \\ 1 & \text{if} \quad \boldsymbol{wx} + b > 0 \end{cases}$$

without activation function



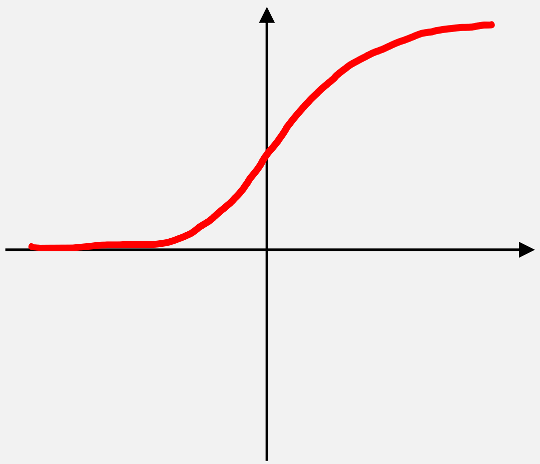
$$\text{output} = \frac{1}{1 + e^{-(\boldsymbol{wx}+b)}}$$

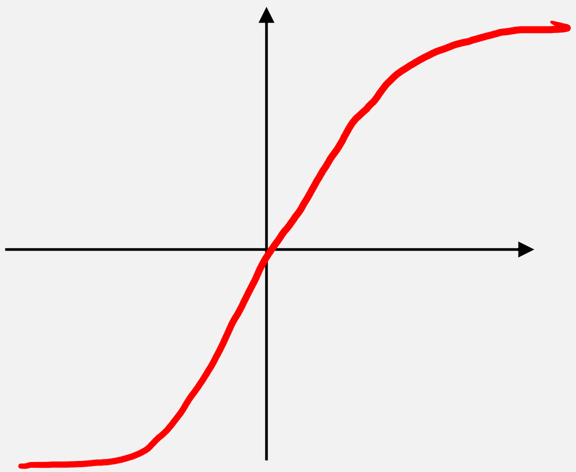
with the sigmoid activation function

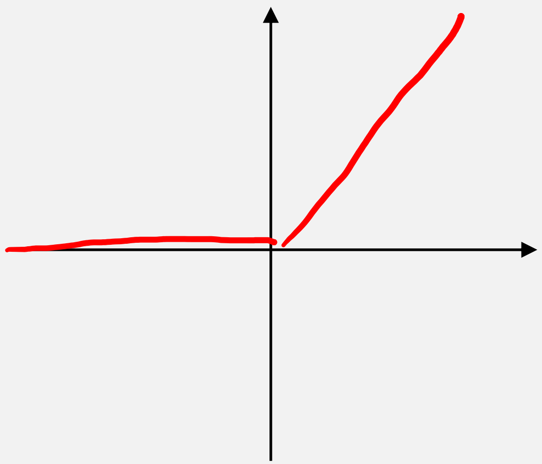


DIFFERENT ACTIVATION FUNCTIONS

sigmoid	tanh	ReLU
		"rectified linear unit"
$\frac{1}{1+e^{-z}}$	$\tanh(z)$	$\max(0, z)$







ACTIVATION IN THE OUTPUT LAYER

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

output layer

class A	5		0.92
class B	2.5	$\xrightarrow{\text{softmax}}$	0.07
class C	0.5		0.01

probability of belonging to particular class

sums to 1

FIXING THE ACCURACY PROBLEM

Define a loss function $\mathcal{L}$

belongs to class 2

$$y(x_1) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

true class

$$\hat{y}(x_1, w, b) = \begin{bmatrix} 0.1 \\ 0.8 \\ 0.07 \\ 0.03 \end{bmatrix}$$

prediction

$$\mathcal{L} = \frac{1}{2n} \sum_i \| y(x_i) - \hat{y}(x_i, w, b) \|^2 \quad (\text{MSE})$$

↳ # training data

THE LOSS FUNCTION

The quadratic loss function captures the general idea

$$L(\boldsymbol{w},\boldsymbol{b}) = \frac{1}{2n}\sum_x \left|\left|y(x) - \hat{y}(x,\boldsymbol{w},\boldsymbol{b})\right|\right|^2$$

but usually we use the cross-entropy loss function

dot product

$$\mathcal{L}(w,b) = -\frac{1}{n}\sum_x y(x)\cdot \ln(\hat{y}(x,w,b))$$

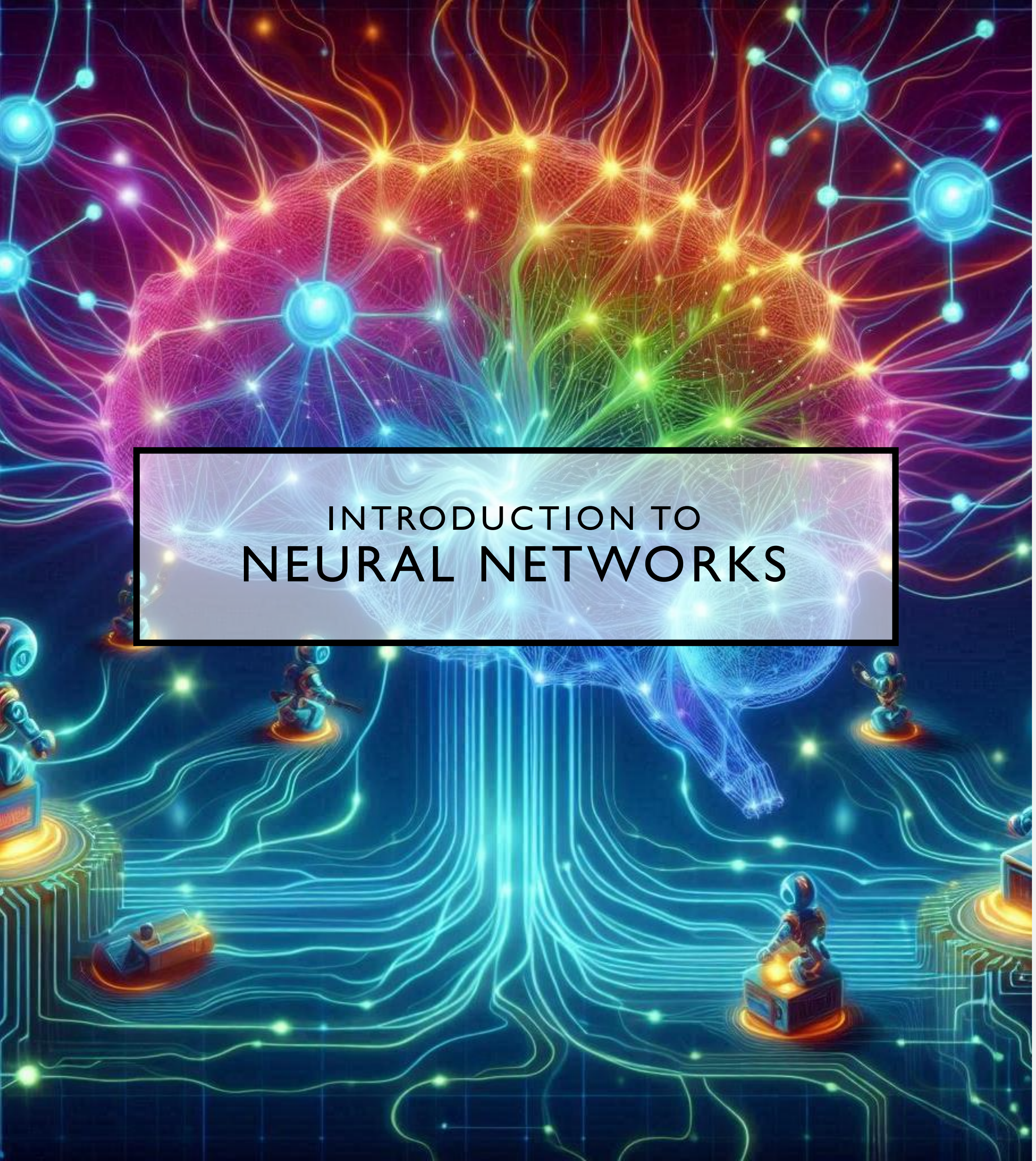
$$y(x)\cdot \ln(\hat{y}(x_1,w,b)) = 0\times \ln 0{,}1 + 1\times \ln 0{,}8 + 0\times \dots + 0 \dots$$

$$= \ln 0{,}8 = -0{,}223$$

Example from before

$$y(x_1) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{y}(x_1,\boldsymbol{w},\boldsymbol{b}) = \begin{bmatrix} 0.1 \\ 0.8 \\ 0.07 \\ 0.03 \end{bmatrix}$$



- What is a neural network?
- How do we structure it?
- How do we train it?
- How do we implement it?

LET'S TRY TO MAKE ONE

Jupyter Notebook **Neural networks - Digits**