



INTRODUCTION TO NEURAL NETWORKS

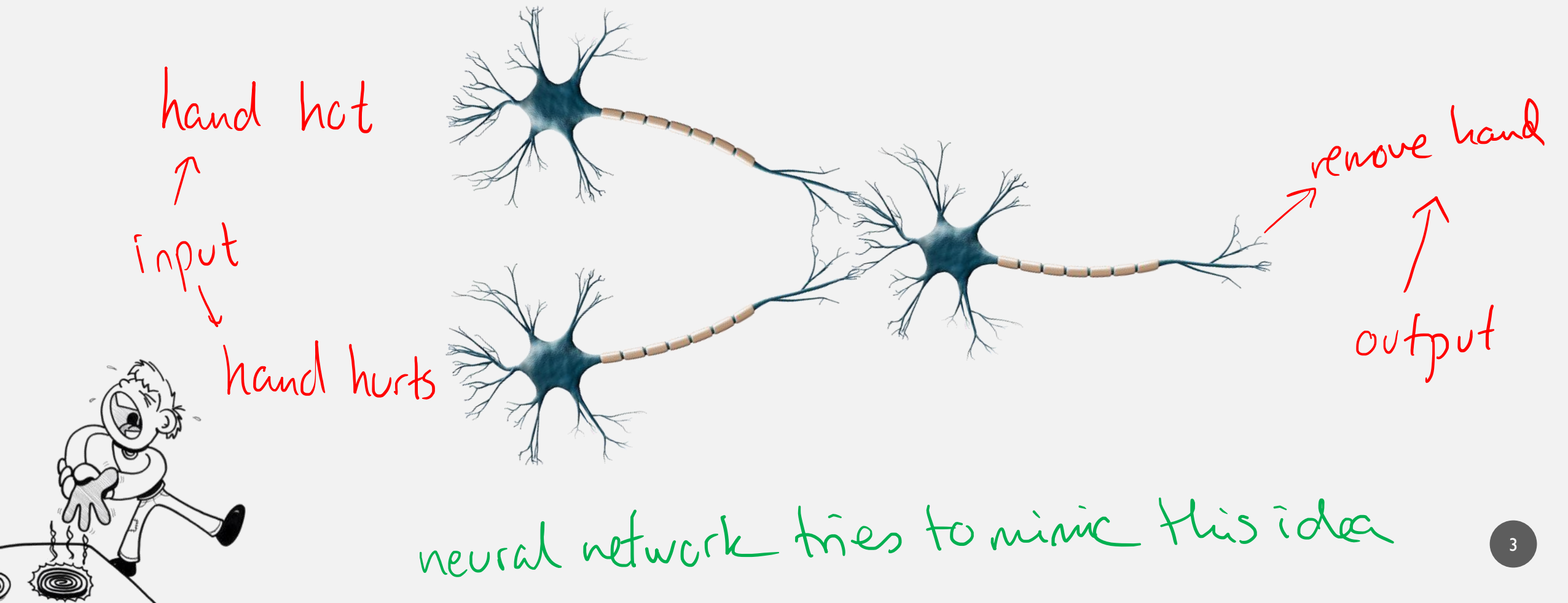
- Lecture II
- MALI, 2024



INTRODUCTION TO NEURAL NETWORKS

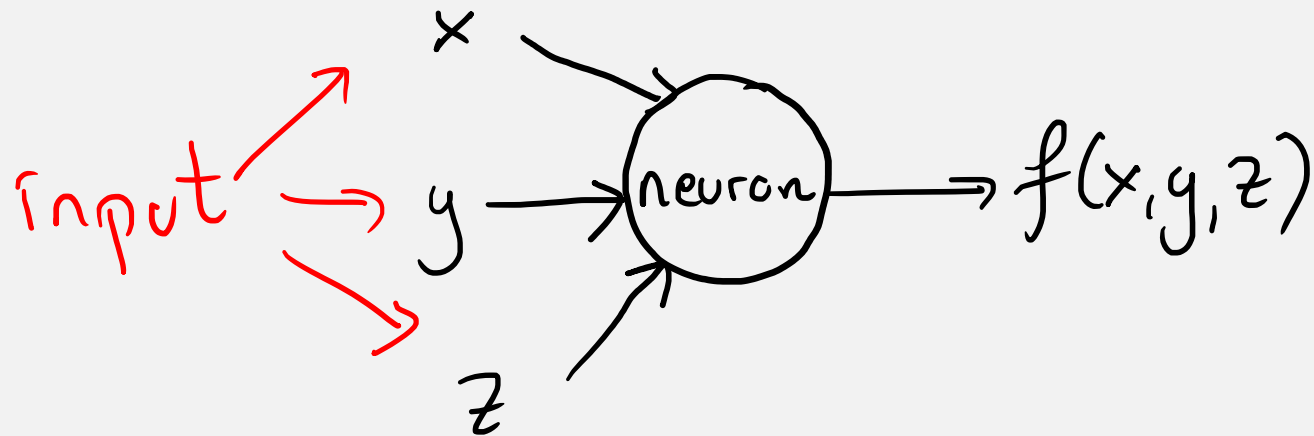
- What is a neural network?
- How do we structure it?
- How do we train it?
- How do we implement it?

NEURONS

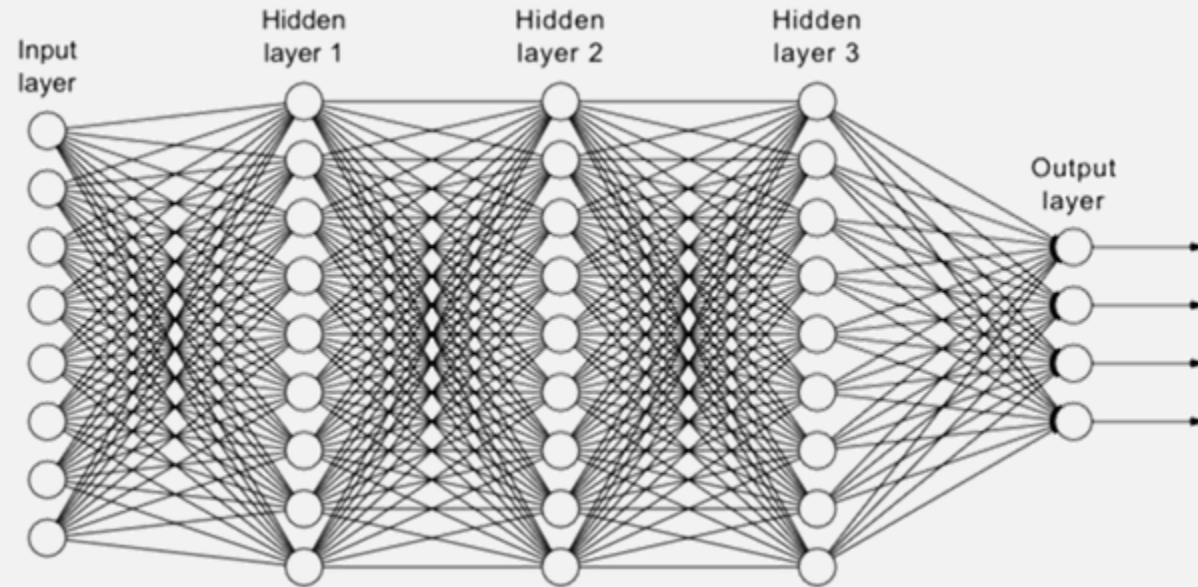


ARTIFICIAL NEURONS

are really just functions



ARTIFICIAL NEURAL NETWORKS



But how do we train a neural network to make good decisions?

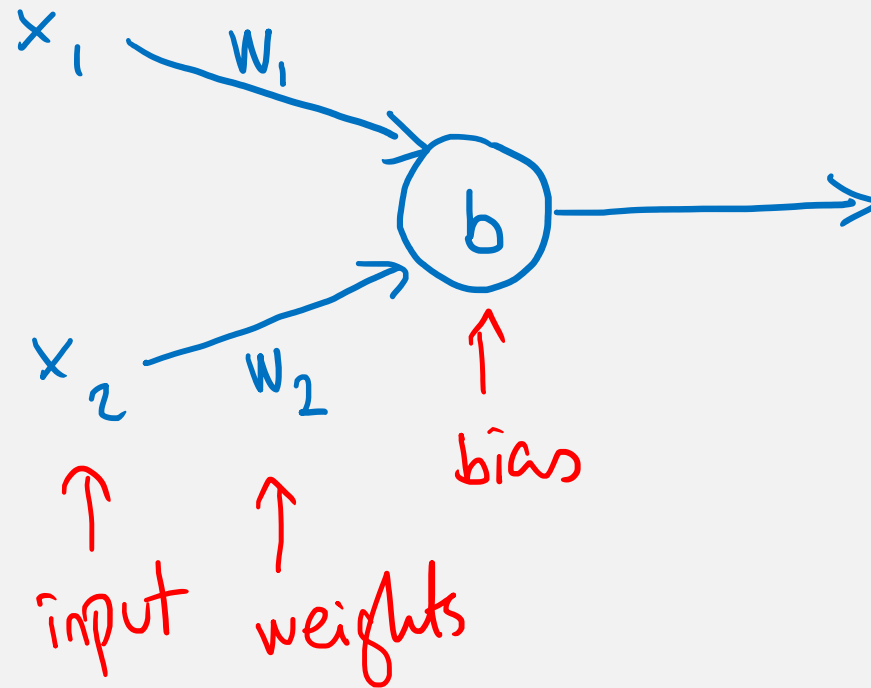
HOWEVER ...

... 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

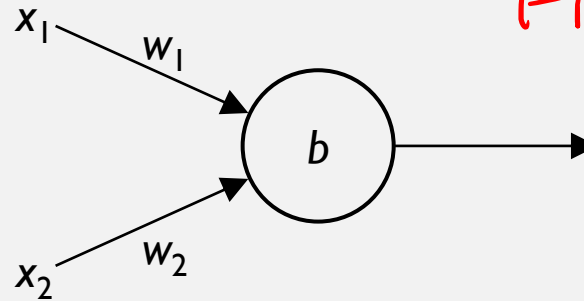


a neuron \Rightarrow
a linear combination

$$w_1x_1 + w_2x_2 + b = \vec{w} \cdot \vec{x} + b$$

PERCEPTRONS

↳ MAKES THE TRANSFORMATION
FROM INPUT → OUTPUT
NONLINEAR

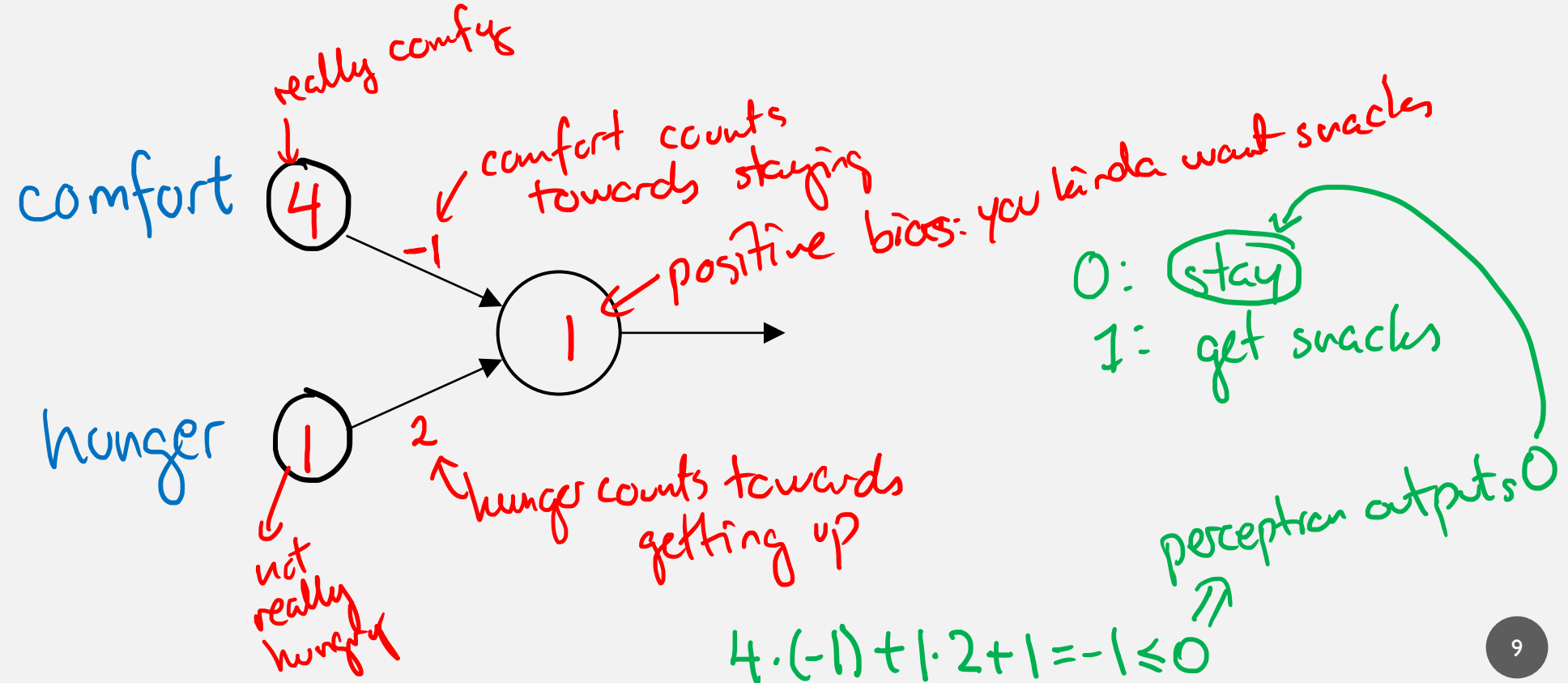


$$\text{output} = \begin{cases} 0 & \text{if } \vec{w} \cdot \vec{x} + b \leq 0 \\ 1 & \text{if } \vec{w} \cdot \vec{x} + b > 0 \end{cases}$$

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?



THE SNACK EXAMPLE II

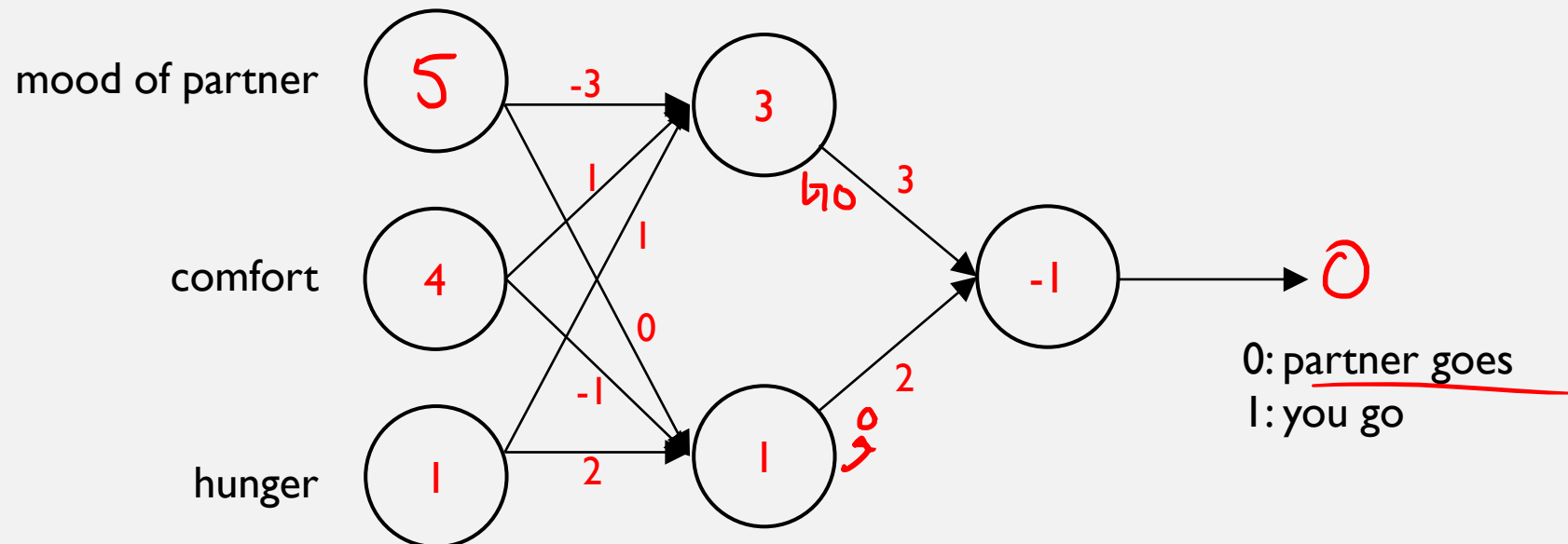
Luckily, your partner is not sitting in the couch!



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 and biases
for a given network architecture

THIS LEAVES TWO QUESTIONS

1. How to structure the network
2. How to optimize weights & biases

THE TENSORFLOW PLAYGROUND



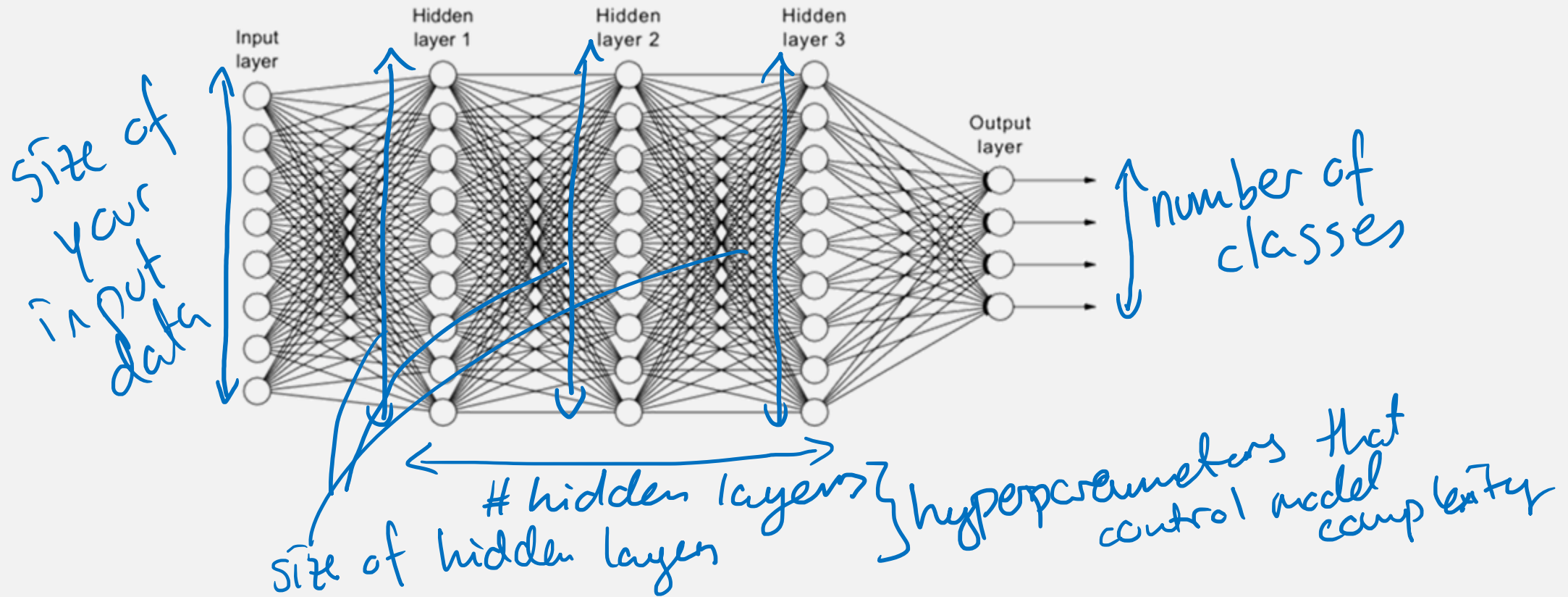
playground.tensorflow.org

An abstract, colorful visualization of a neural network. It features a central blue node from which numerous lines radiate outwards, connecting to other nodes of various colors (red, orange, yellow, green, blue, purple). The background is dark with a bokeh effect of light spots. A semi-transparent rectangular box is centered over the image, containing the title text.

INTRODUCTION TO NEURAL NETWORKS

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

HOW TO STRUCTURE THE NETWORK



neuron in input size > hidden size > output size
Rules of thumb: usually, one hidden layer is enough

The background of the slide is a complex, abstract illustration. It features a central silhouette of a human brain, which is filled with intricate, swirling patterns in shades of blue, teal, and purple. These patterns resemble neural pathways or data flow. Surrounding the brain are numerous small, glowing spheres in various colors (blue, green, yellow, pink) and thin, curved lines that suggest a network or a dynamic system. The overall aesthetic is futuristic and technological.

INTRODUCTION TO NEURAL NETWORKS

- 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: 1-accuracy or something similar



GRADIENT DESCENT

1. Find the direction in which the descent is steepest

$$\nabla L(\beta) = \left[\frac{\partial L}{\partial \beta_0}(\beta), \frac{\partial L}{\partial \beta_1}(\beta), \dots \right]$$

gradient "altitude" \rightarrow points \rightarrow weights & biases

2. Take a step in that direction

$$\beta \leftarrow \beta - \eta \nabla L(\beta)$$

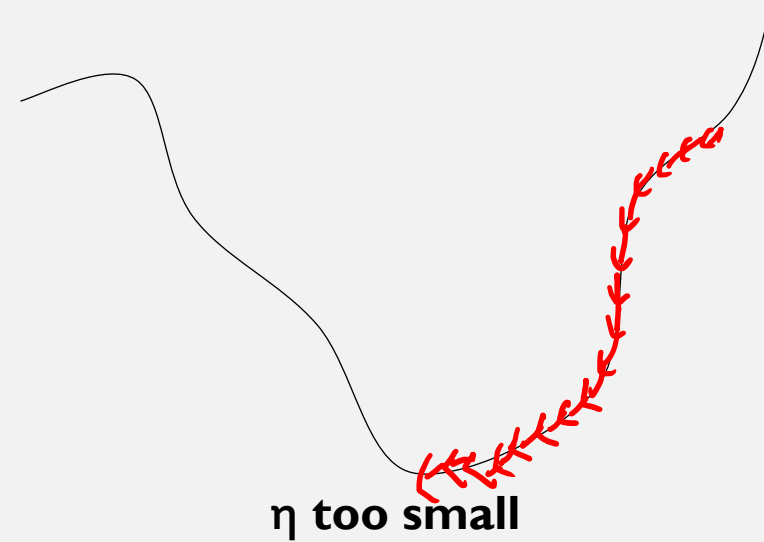
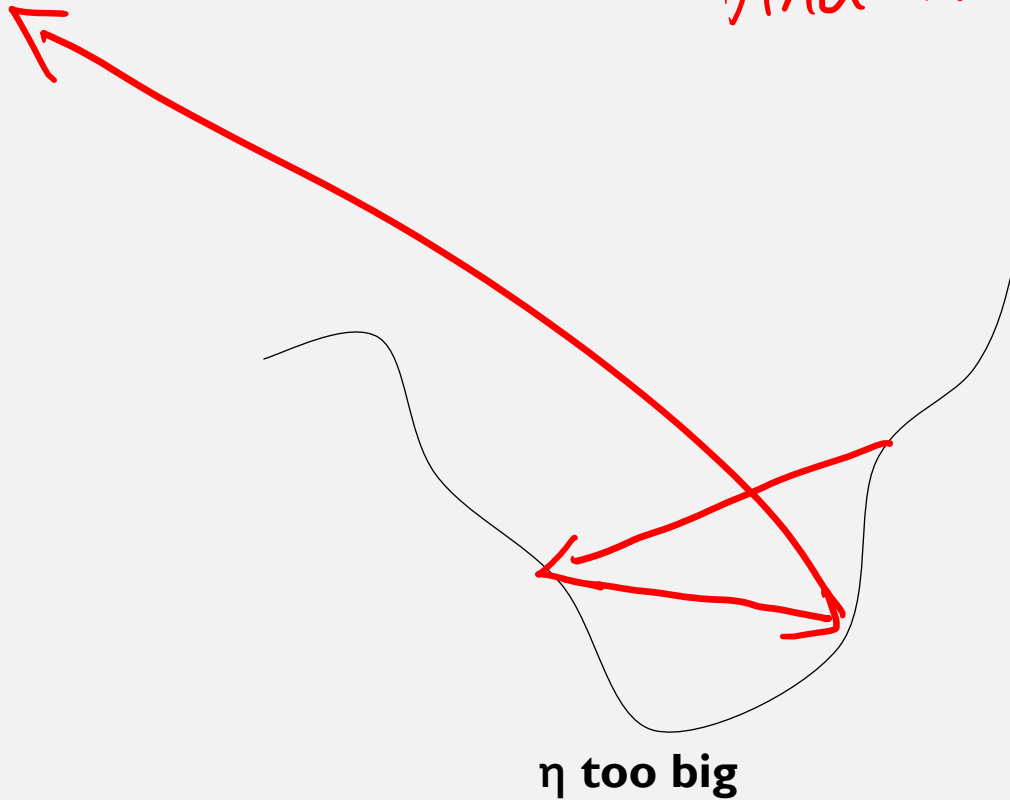
\rightarrow learning rate (step size)

3. Repeat until you reach the bottom

$$\nabla L(\beta) = [0, 0, 0, \dots, 0]$$

THE LEARNING RATE

find it by trial-and-error



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 are
mostly flat but
a small step leads
to a sudden
change



SUDDEN CHANGES

1. Perceptrons suddenly change from 0 to 1

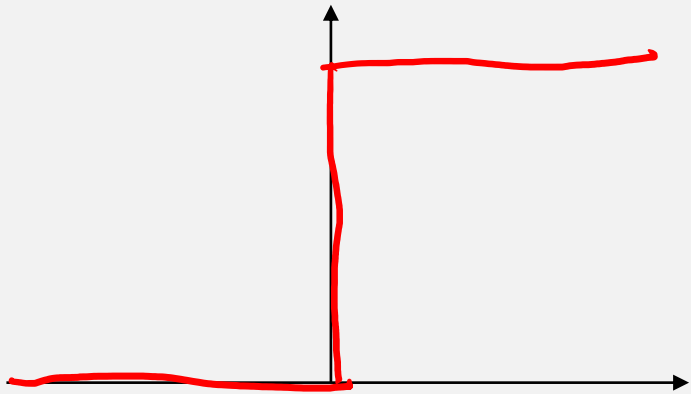
⇒ don't use perceptrons

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

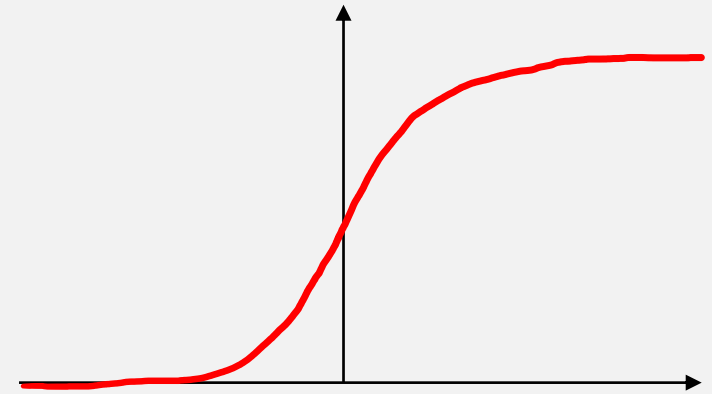
⇒ don't use accuracy

FIXING THE PERCEPTRON PROBLEM

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



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

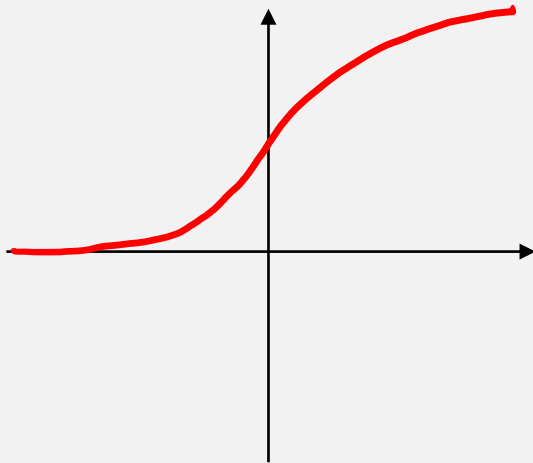


changes smoothly

DIFFERENT ACTIVATION FUNCTIONS

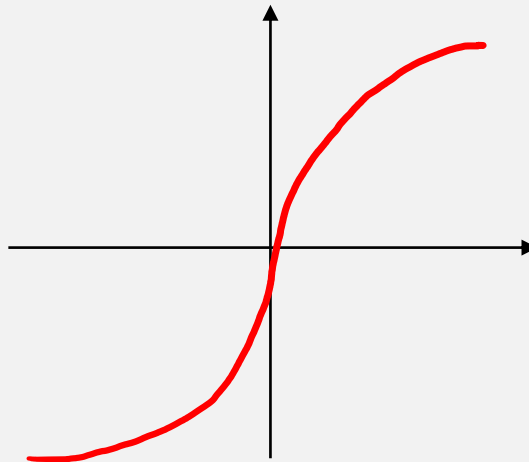
sigmoid

$$\frac{1}{1+e^{-z}}$$



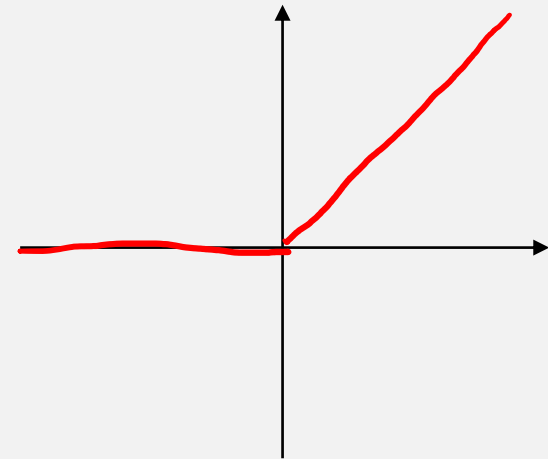
tanh

$$\tanh(z)$$



ReLU

"Rectified
linear unit"
 $\max(0, z)$



ACTIVATION IN THE OUTPUT LAYER

$$z = \vec{w} \cdot \vec{x} + b$$

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

output layer

class A	5
class B	2.5
class C	0.5

\Rightarrow
softmax

0.92
0.07
0.01

} sums to 1

\rightarrow probability of belonging to each class

FIXING THE ACCURACY PROBLEM

Define a loss function L

true label
(class 2) $\rightarrow y_{x_i} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

output of
network $\rightarrow a(x_i, w, b) = \begin{bmatrix} 0.1 \\ 0.8 \\ 0.07 \\ 0.03 \end{bmatrix}$

$$L(w, b) = \frac{1}{2n} \sum_i \|y(x_i) - a(x_i, w, b)\|^2$$

#training data points

THE LOSS FUNCTION

The quadratic loss function captures the idea

$$L(\mathbf{w}, \mathbf{b}) = \frac{1}{2n} \sum_x ||y(x) - a(x, \mathbf{w}, \mathbf{b})||^2$$

but usually we use the cross-entropy loss function

$$L(\mathbf{w}, \mathbf{b}) = -\frac{1}{n} \sum_j y(x_j) \overset{\text{dot product}}{\bullet} \ln(a(x_j, \mathbf{w}, \mathbf{b}))$$

$$\begin{aligned} y(x_1) \bullet \ln(a(x_1, \mathbf{w}, \mathbf{b})) &= 0 \times \ln 0.1 + 1 \times \ln 0.8 \\ &\quad + 0 \times \sim + 0 \times \sim \\ &= 1 \times \ln 0.8 = -0.223 \end{aligned}$$

proportional to "how wrong we are"

Example from before

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

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



INTRODUCTION TO NEURAL NETWORKS

- 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**

