

## Lecture 07:

# Introduction to Artificial Neural Networks (ANN)



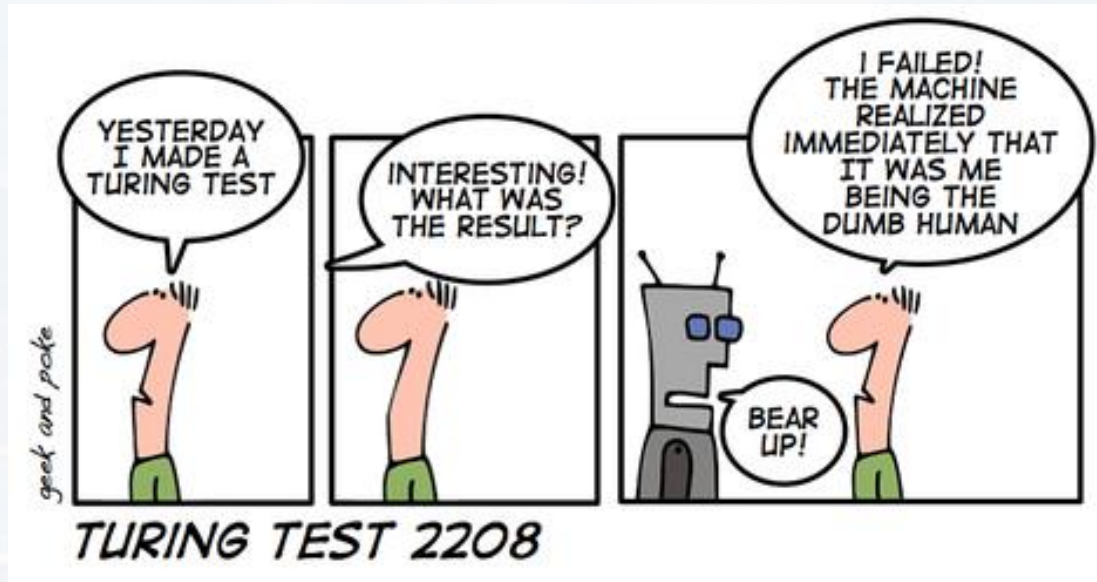
Markus Hohle

University California, Berkeley

**Machine Learning Algorithms**

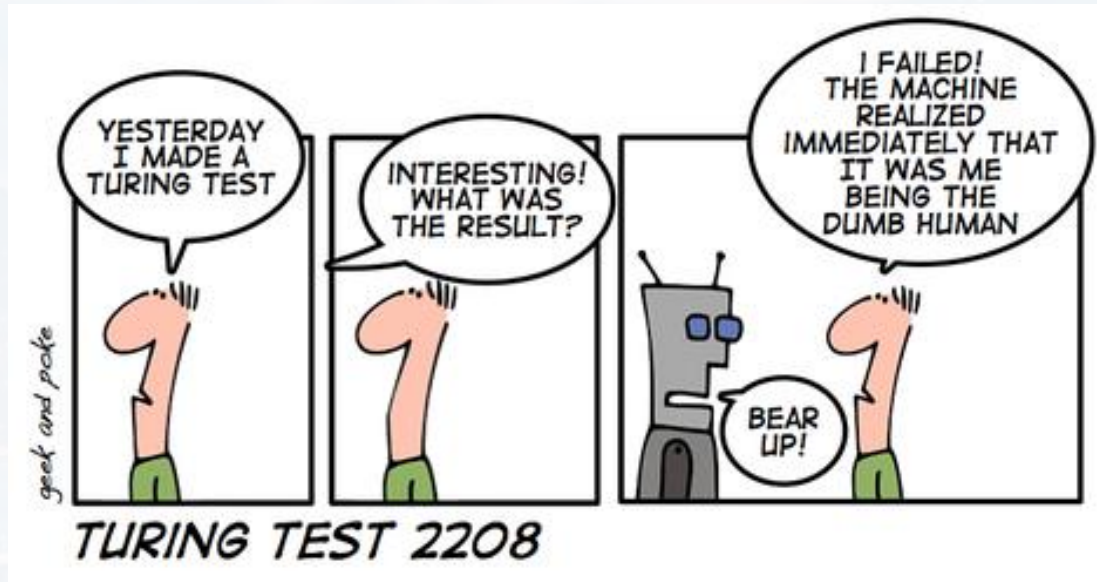
MSSE 277B, 3 Units

Fall 2024



## Outline

- Motivation & Overview
- The Perceptron aka Neuron
- Building a Perceptron



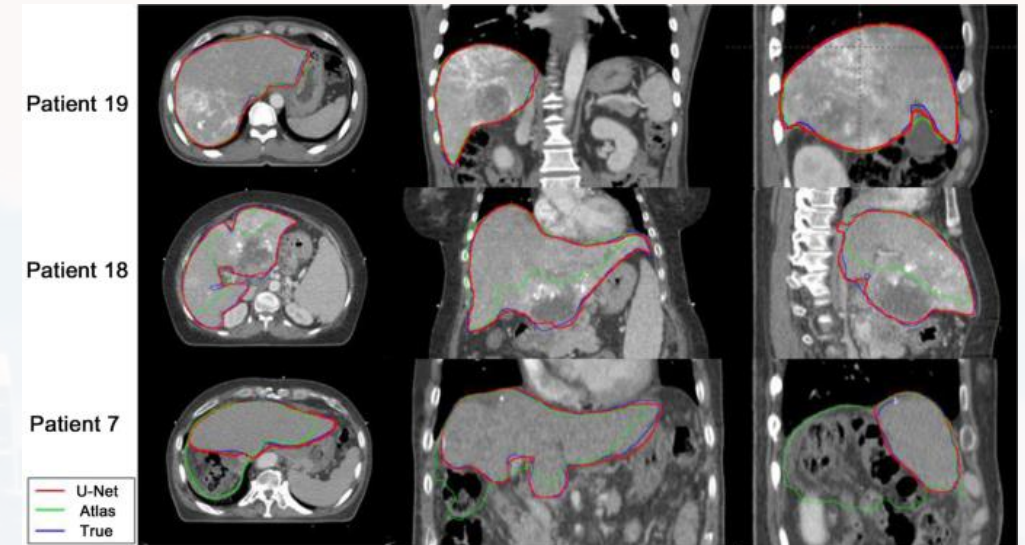
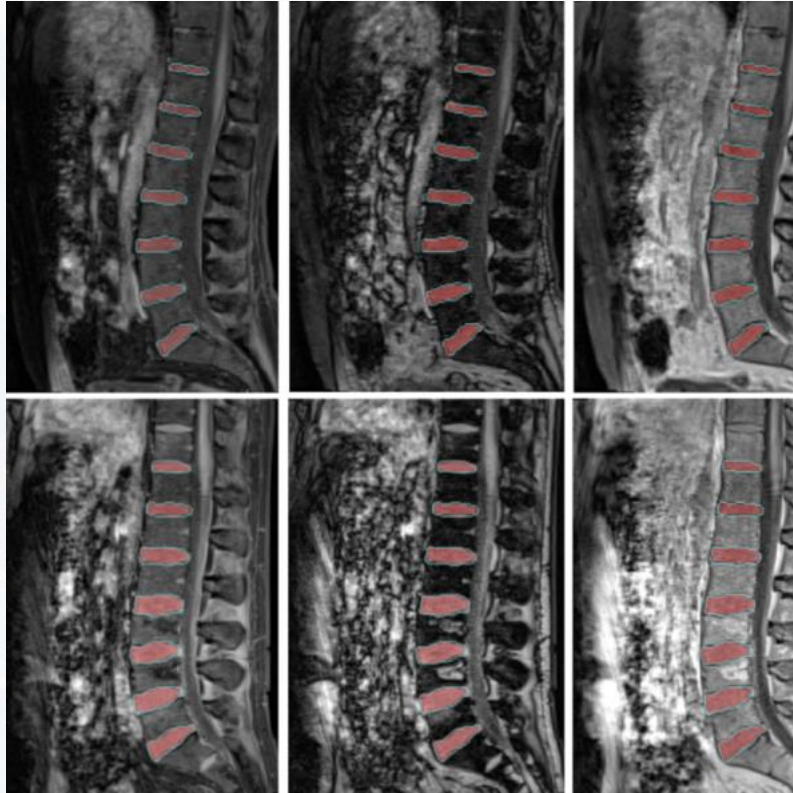
## Outline

- **Motivation & Overview**
- The Perceptron aka Neuron
- Building a Perceptron





## Why Artificial Neural Networks?



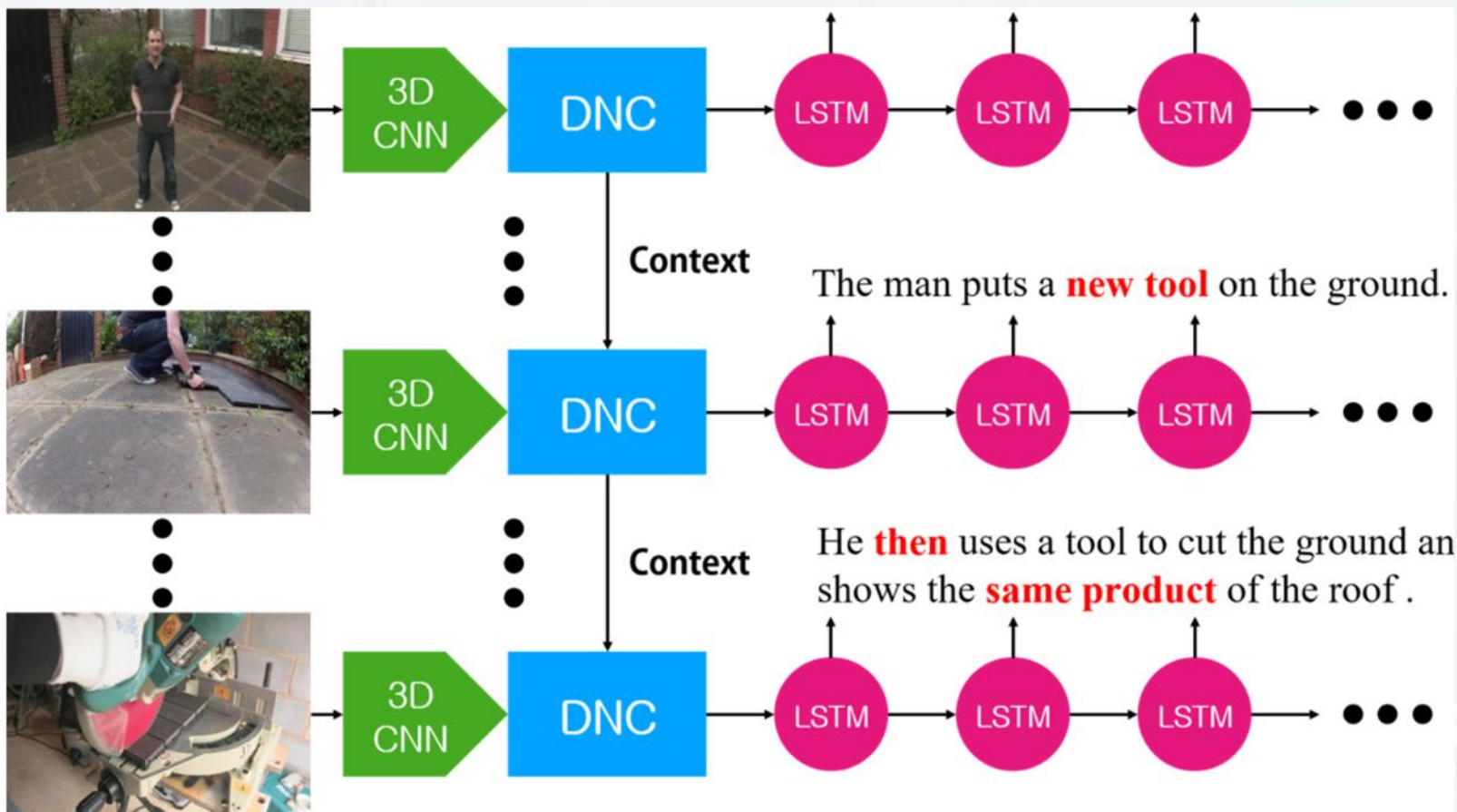


## Why Artificial Neural Networks?

**S:** 我<sup>3</sup>就<sup>4</sup>取<sup>5</sup>钱<sup>6</sup>给<sup>7</sup>了 她们  
i will get money to perf. them

**T:** <sup>2</sup>i <sup>1</sup>will <sup>0</sup>get the money to them

P(the | get, will, i, 就, 取, 钱, 给, 了)







## Why Artificial Neural Networks?

Article

### DeepSOCIAL: Social Distancing Monitoring and Infection Risk Assessment in COVID-19 Pandemic

YOLOv4-based *Deep Neural Network* (DNN) model for automated people detection in the crowd in indoor and outdoor environments using common CCTV security cameras. The proposed DNN model in combination with an adapted inverse perspective mapping (IPM) technique and SORT tracking algorithm leads to a robust people detection and social distancing monitoring. The model has been trained against two most comprehensive datasets by the time of the research—the Microsoft Common Objects in Context (MS COCO) and Google Open Image datasets. The system has been

health authorities have set the 2-m physical distancing as a mandatory safety measure in shopping centres, schools and other covered areas. In this research, we develop a hybrid *Computer Vision* and YOLOv4-based *Deep Neural Network* (DNN) model for automated people detection in the crowd in indoor and outdoor environments using common CCTV security cameras. The proposed DNN model in combination with an adapted inverse perspective mapping (IPM) technique and SORT tracking algorithm leads to a robust people detection and social distancing monitoring. The model has been trained against two most comprehensive datasets by the time of the research—the Microsoft Common Objects in Context (MS COCO) and Google Open Image datasets. The system has been evaluated against the Oxford Town Centre dataset (including 150,000 instances of people detection)

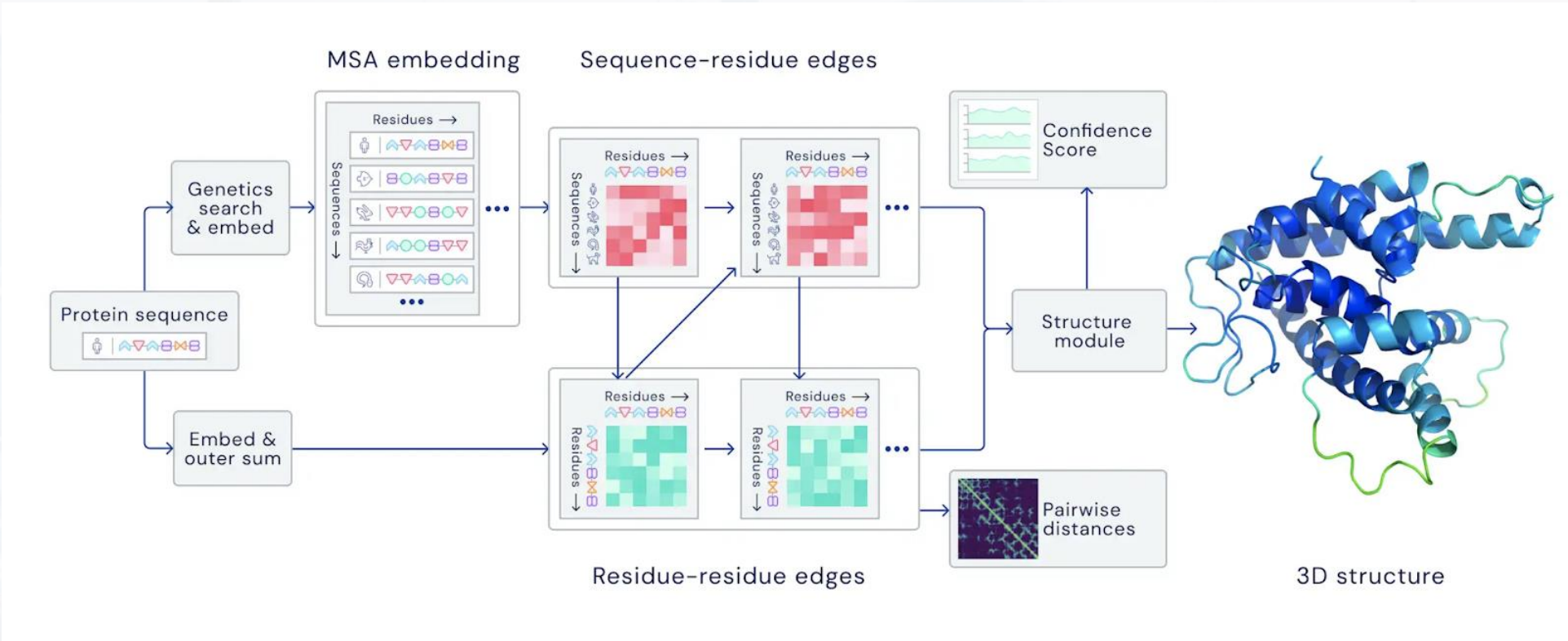


## Why Artificial Neural Networks?



Demis Hassabis  
@demishassabis

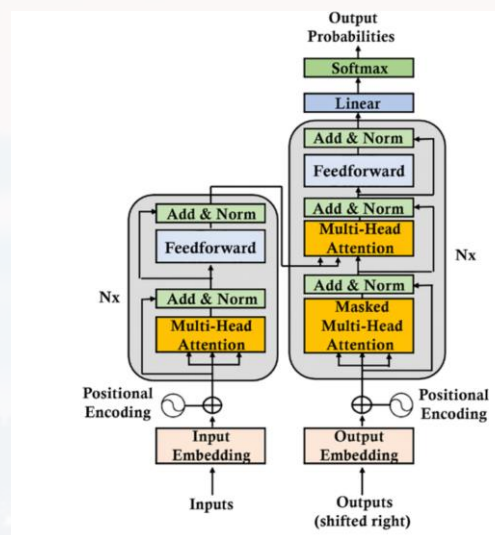
Thrilled to announce our first major breakthrough in applying AI to a grand challenge in science. [#AlphaFold](#) has been validated as a solution to the 'protein folding problem' & we hope it will have a big impact on disease understanding and drug discovery:







## Why Artificial Neural Networks?



language models  
“understanding” context

AI art




AI generated videos







## Why Artificial Neural Networks?




World ▾ US Election Business ▾ Markets ▾ Sustainability ▾ Legal ▾ Breakingviews ▾ Technology


Artificial Intelligence | Ukraine and Russia at War

# Ukraine rushes to create AI-enabled war drones

By Max Hunder

July 18, 2024 3:36 PM GMT+2 · Updated 2 months ago






**navigation and more complex programs including interconnected swarms.**


**TECHNOLOGY**

## In the Age of AI, Can a Terminator Scenario Actually Happen?

Today we are using AI for a lot of reasons. And unknowingly, we are giving it a lot of power over our day-to-day lives. But can AI turn upon us with the same resources that we have provided it? Or is it a fantasy to think that AI can ever go against humans?



**By Paul Baker**  
Director of Partnerships Paul Baker builds strong business relationships between BairesDev and clients through strategy and partnership management.

6 min read  Share



### In the next lectures we will learn about:

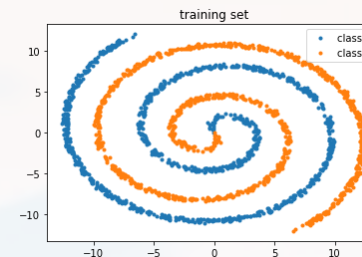
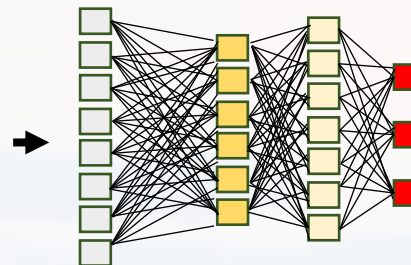
- |                                      |  |
|--------------------------------------|--|
| - classical multi-layer ANN          | regression, classification   |
| - <b>Convolution Neural Networks</b> | image classification, segmentation, sequence analysis                                      |
| - <b>Recurrent Neural Networks</b>   | time series forecasting and classification   |
| - <b>Long-Short-Term-Memory ANNs</b> | time series forecasting and classification,<br>language processing                         |
| - combining the above networks       |  |
| - <b>Graph Neural Networks</b>       | molecular 3D structures, process/workflow planning   |
| - Transformers                       | time series analysis, <b>Large Language Models</b> ,<br><b>Natural Language Processing</b> |

- bonus:**
- running ANNs on GPUs using Cuda in PyTorch → 20 – 50 times faster!
  - Parallel Processing in Python



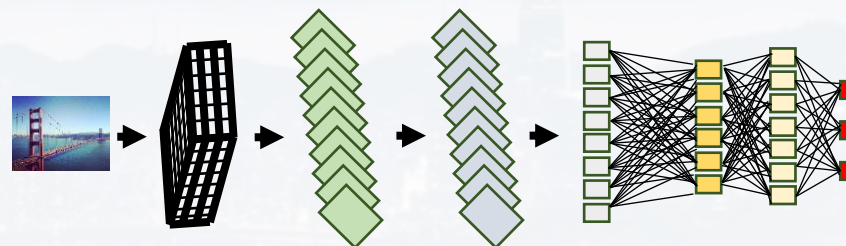


fully connected  
(ANN)



classification/  
regression

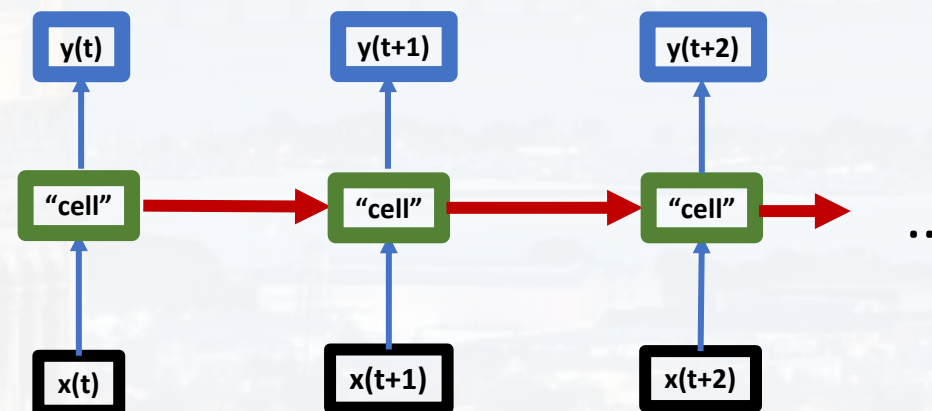
convolution  
(CNN)



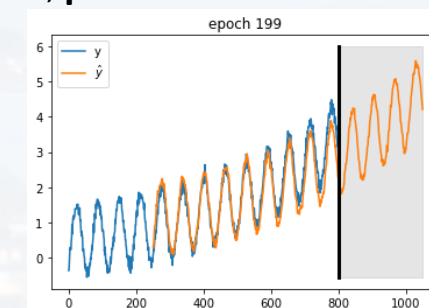
classification/  
regression



recurrent  
(RNN, LSTM)

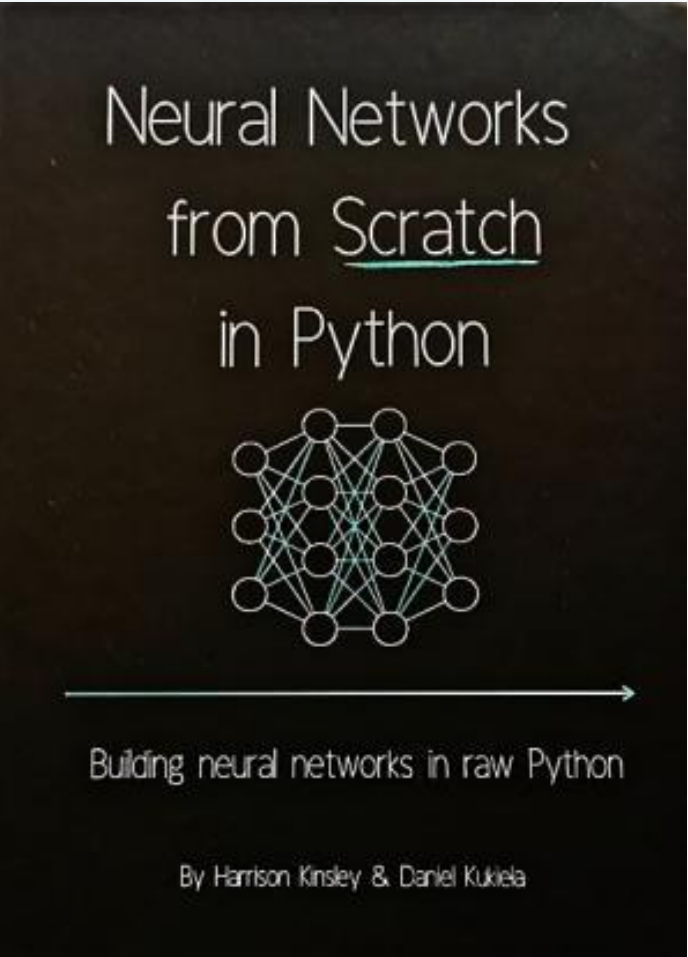



classification/  
Regression, **prediction**





## Tutorials and Books





## Dr Fridolin

@DrFridolin · 111 subscribers · 37 videos

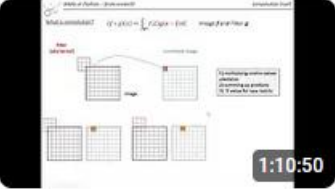
Dr Fridolin shows you how to program AI step by step, line by line without using external libraries. ...more

[github.com/DrBigMau](https://github.com/DrBigMau)

Subscribe

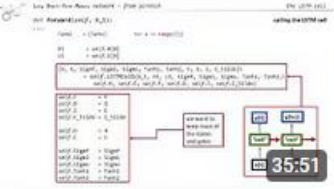
Home Videos Playlists

### Videos




**CNN 1a Intro and Convolution**

92 views · 6 months ago




**LSTM 2c LSTM Cell - the Code**

35:51



**LSTM 1 Intro**

397 views · 6 months ago



**LSTM 4 Full Backpropagation**

813 views · 6 months ago





## Tutorials and Books

examples and **application** (CNN/RNN/LSTM and more):



**Jason Brownlee**  
Machine Learning Mastery

all about **transformers, LLM/NLP** and way more



## Andrej Karpathy

@AndrejKarpathy · 451K subscribers · 14 videos

FAQ ...more

[karpathy.ai](https://karpathy.ai) and 2 more links

Subscribe



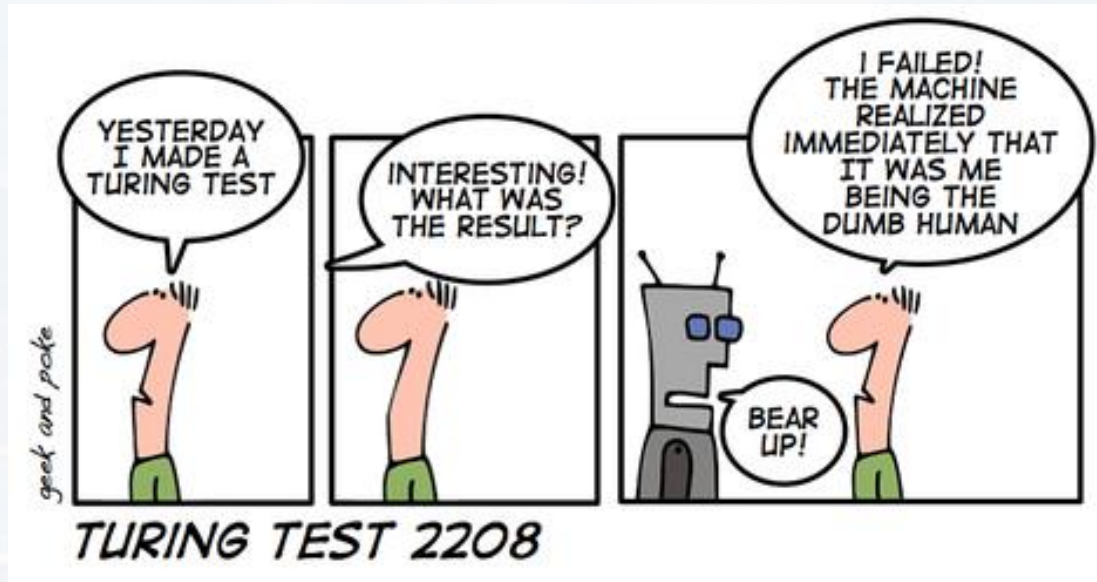
## Misra Turp

@misraturp · 38.2K subscribers · 163 videos

Here is where we learn! This is a space to take it slow

[misraturp.com/roadmap](https://misraturp.com/roadmap) and 3 more links

Subscribe



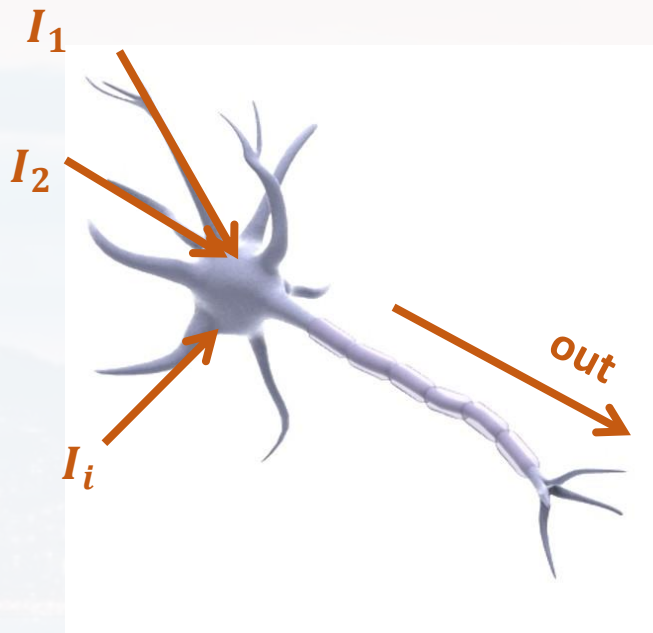
## Outline

- Motivation & Overview
- **The Perceptron aka Neuron**
- Building a Perceptron





the perceptron:



**what we know...**

- inputs enter the neuron
- something happens inside the neuron
- neuron generates an output

**...how we could model it**

- inputs must be weighted (important vs unimportant input)
- **learning process: changing weights**
- output = sum of weighted inputs

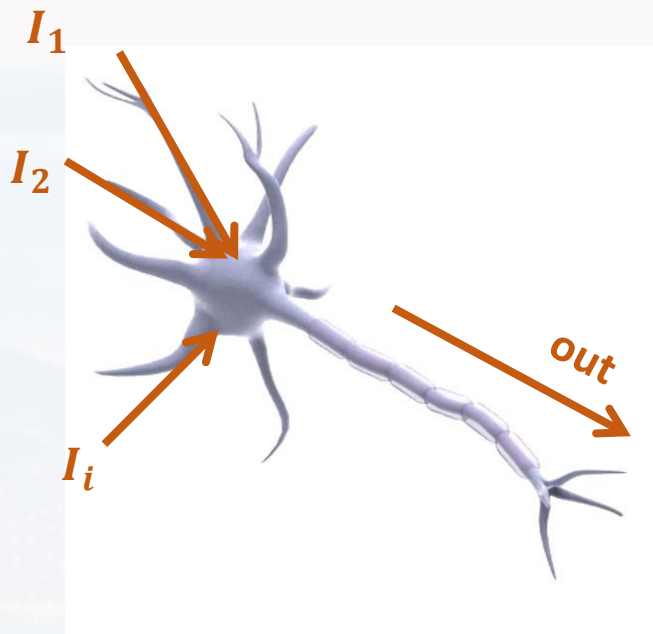
$I_i$       input  $i$   
 $w_i$       corresponding weight  
 $b$       bias (base potential)

$$net = \sum_i I_i \cdot w_i + b$$

$\underbrace{\hspace{1.5cm}}$   
**dot product**



the perceptron:



$$net = \sum_i I_i \cdot w_i + b$$

$I_i$   
 $w_i$   
 $b$

input  $i$   
corresponding weight  
bias (base potential)

*dot product*

recall: linear models

$$y_k = \beta_0 + \sum_{n=1}^N \beta_n x_n + \epsilon$$

$$\begin{pmatrix} y_1 \\ \vdots \\ y_k \\ \vdots \\ y_K \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} & \dots & x_{1N} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ 1 & x_{k1} & & & x_{kn} & & \\ \vdots & \vdots & & & \vdots & & \vdots \\ 1 & \dots & & & \dots & & \\ 1 & x_{K1} & x_{K2} & \dots & x_{Kn} & \dots & x_{KN} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \\ \vdots \\ \beta_N \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_k \\ \vdots \\ \epsilon_K \end{pmatrix}$$





the perceptron:



$$net = \sum_i I_i \cdot w_i + b$$

$I_i$   
 $w_i$   
 $b$

input  $i$   
corresponding weight  
bias (base potential)

dot product

simple example:

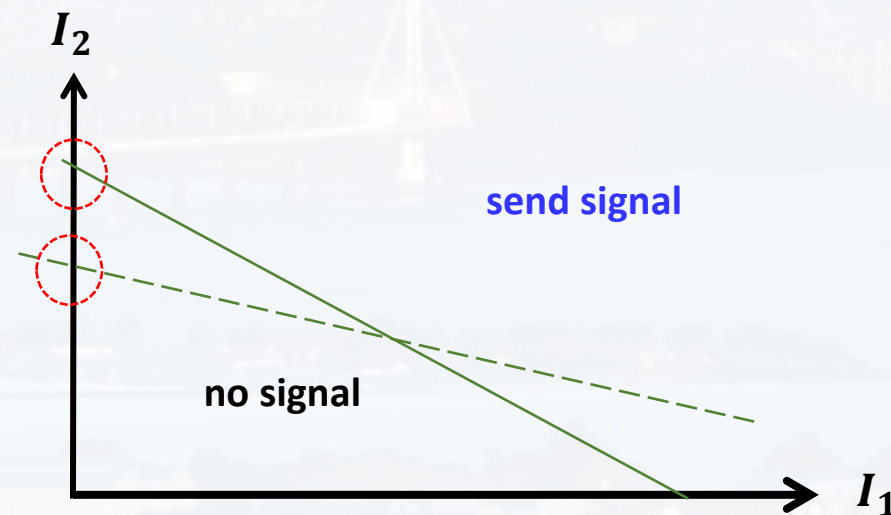
**one** neuron with a **switch**, threshold **T**  
and **two** input channels

fire if:  $b + I_1 w_1 + I_2 w_2 > T$

$$I_2 = -\frac{w_1}{w_2} I_1 + \frac{T - b}{w_2}$$

slope

offset





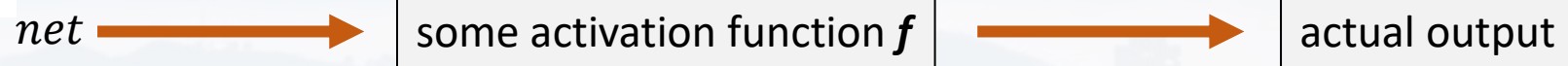
the perceptron:



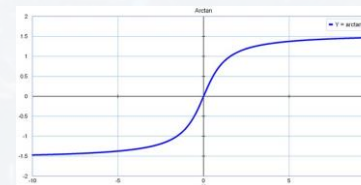
$$net = \sum_i I_i \cdot w_i + b$$

$I_i$   
 $w_i$   
 $b$

input  $i$   
corresponding weight  
bias (base potential)



-  $y = \arctan(net)$



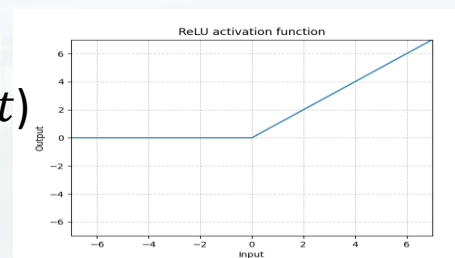
$(-\infty; +\infty) \rightarrow (-\pi/2; +\pi/2)$

-  $y = \text{sigm}(net)$



$(-\infty; +\infty) \rightarrow (0; 1)$

-  $y = \text{ReLU}(net)$   
 $= \max(0, net)$

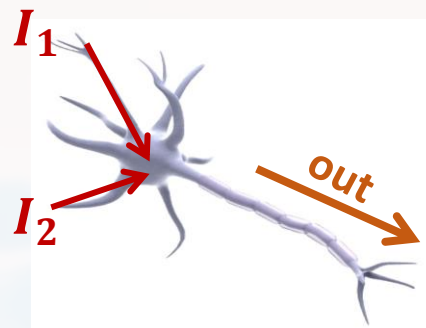


$(-\infty; +\infty) \rightarrow (0; +\infty)$





the perceptron:



$$net = \sum_i I_i \cdot w_i + b$$

$I_i$   
 $w_i$   
 $b$   
 $f$

input  $i$   
corresponding weight  
bias (base potential)  
activation function

$$y = f(net)$$

learning:

$$net = \sum_i I_i \cdot w_i + b \xrightarrow{\text{activation function } f} y = f(net) \xrightarrow{\text{objective function } E} E = \frac{1}{2} (t - y)^2$$

target output  $t$

finding best  $w_i$  by **minimizing**  $E$

$$\Delta w_i = w_i(\text{new}) - w_i(\text{old}) = -\alpha \frac{\partial E}{\partial w_i}$$

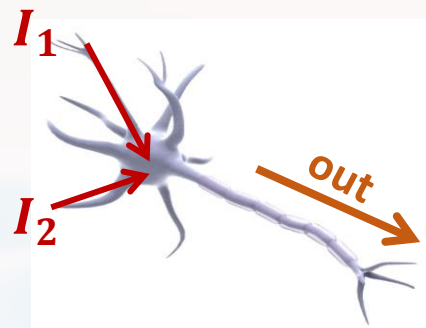
gradient descent with learning rate  $\alpha$

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial net} \frac{\partial net}{\partial w_i}$$

chain rule



the perceptron:



$$net = \sum_i I_i \cdot w_i + b$$

$I_i$   
 $w_i$   
 $b$   
 $f$

input  $i$   
corresponding weight  
bias (base potential)  
activation function

$$y = f(net)$$

learning:

$$net = \sum_i I_i \cdot w_i + b \xrightarrow{\text{activation function } f} y = f(net) \xrightarrow{\text{objective function } E} E = \frac{1}{2} (t - y)^2$$

target output  $t$

finding best  $w_i$  by **minimizing**  $E$

$$\Delta w_i = w_i(\text{new}) - w_i(\text{old}) = -\alpha \frac{\partial E}{\partial w_i}$$

gradient descent with learning rate  $\alpha$

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial net} \frac{\partial net}{\partial w_i} = -(t - y) f'(net) I_i$$





learning:

$$net = \sum_i I_i \cdot w_i + b \xrightarrow{\text{activation function } f} y = f(net) \xrightarrow{\text{objective function } E} E = \frac{1}{2} (t - y)^2$$

target output  $t$

finding best  $w_i$  by **minimizing**  $E$

$$\Delta w_i = w_i(\text{new}) - w_i(\text{old}) = -\alpha \frac{\partial E}{\partial w_i} \quad \text{gradient descent with learning rate } \alpha$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial net} \frac{\partial net}{\partial w_i} = \boxed{-(t - y)} \boxed{f'(net)} \boxed{I_i}$$

inner derivative

outer derivatives

$f'(net)$

depends on the activation function

The required change of  $E$  propagates back to the changes of  $w_i \rightarrow$  **backpropagation**



learning:

$$net = \sum_i I_i \cdot w_i + b \xrightarrow{\text{activation function } f} y = f(net) \xrightarrow{\text{objective function } E} E = \frac{1}{2} (t - y)^2$$

target output  $t$

finding best  $w_i$  by **minimizing**  $E$

$$\Delta w_i = w_i(\text{new}) - w_i(\text{old}) = -\alpha \frac{\partial E}{\partial w_i} \quad \text{gradient descent with learning rate } \alpha$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial net} \frac{\partial net}{\partial w_i} = \boxed{-(t - y)} \boxed{f'(net)} \boxed{I_i}$$

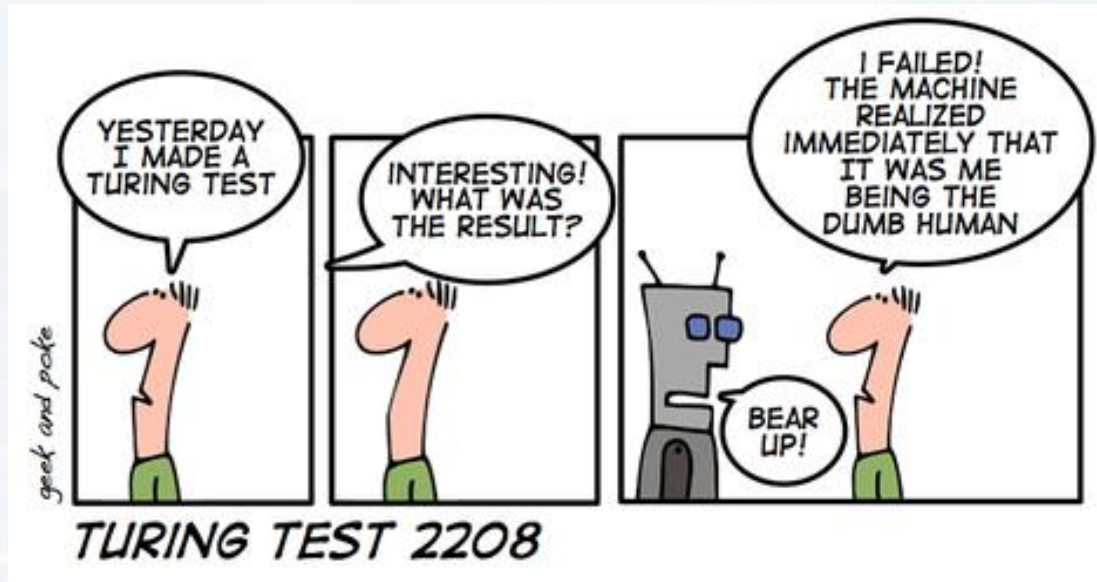
inner derivative      outer derivatives

The required change of  $E$  propagates back to the changes of  $w_i \rightarrow$  **backpropagation**

$$\Delta w_i = w_i(\text{new}) - w_i(\text{old}) = \alpha (t - y) f'(net) I_i$$

$$\Delta b = b(\text{new}) - b(\text{old}) = \alpha (t - y) f'(net) \cdot 1$$



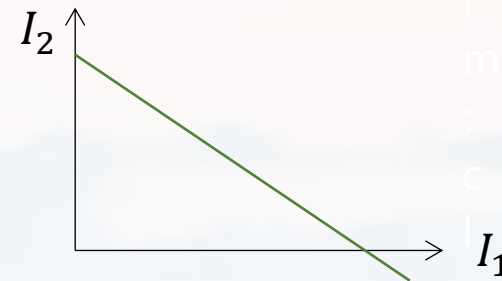


## Outline

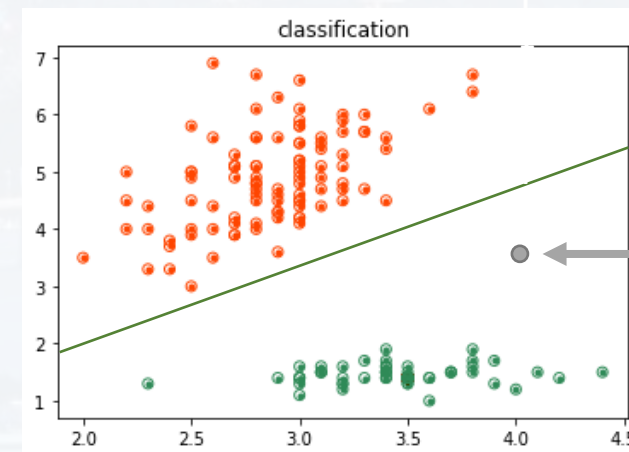
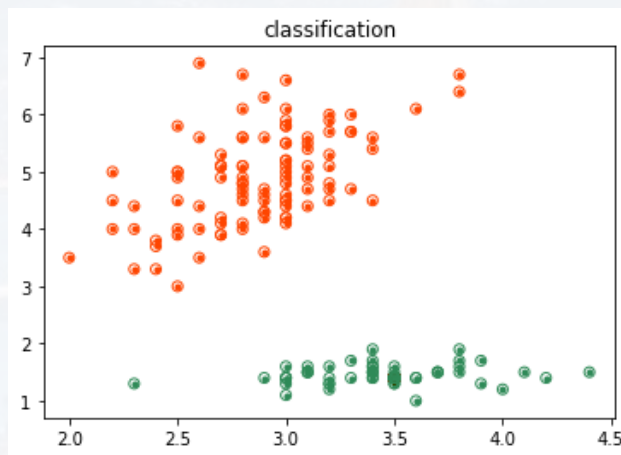
- Motivation & Overview
- The Perceptron aka Neuron
- **Building a Perceptron**



**one** neuron  
**two** features  
**two** classes  
**N** data points



goal



classifying a new  
data point

see the Jupyter Notebook `Perceptron.ipynb` for details



main part of the code:

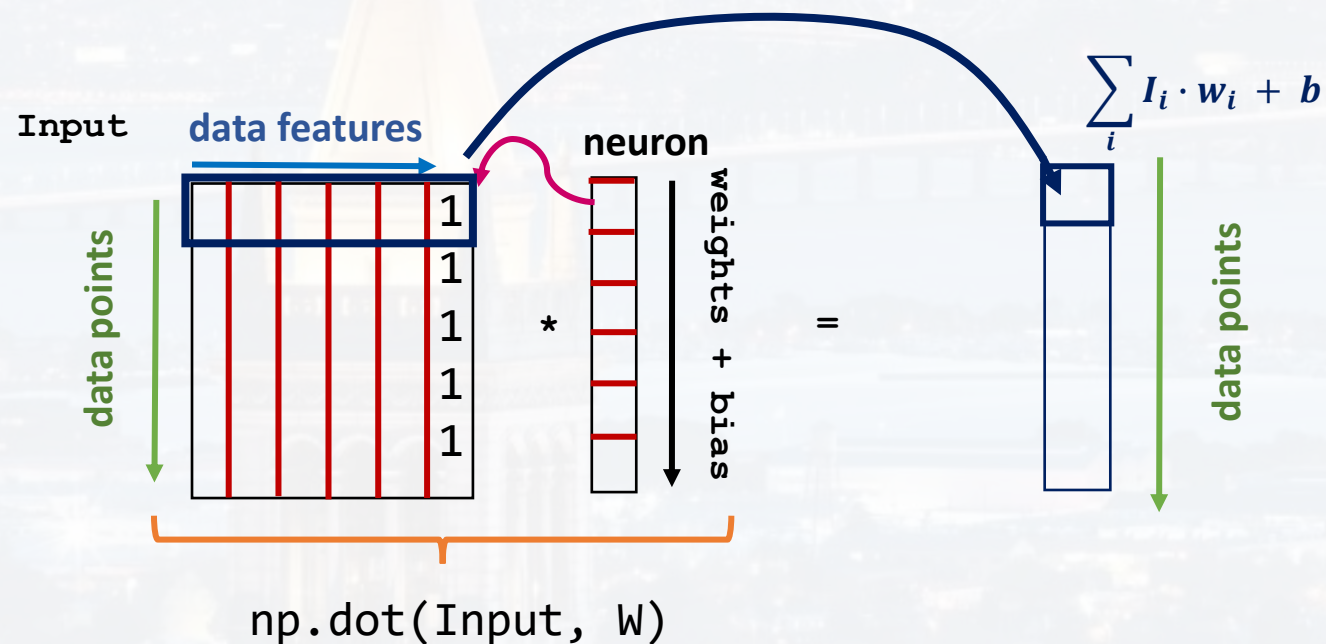
data features

data points

[	5.1	3.5	1.4	0.2	]
[	4.9	3.	1.4	0.2	]
[	4.7	3.2	1.3	0.2	]
[	4.6	3.1	1.5	0.2	]
[	5.	3.6	1.4	0.2	]
[	5.4	3.9	1.7	0.4	]
[	4.6	3.4	1.4	0.3	]
[	5.	3.4	1.5	0.2	]

$$net = \sum_i I_i \cdot w_i + b$$

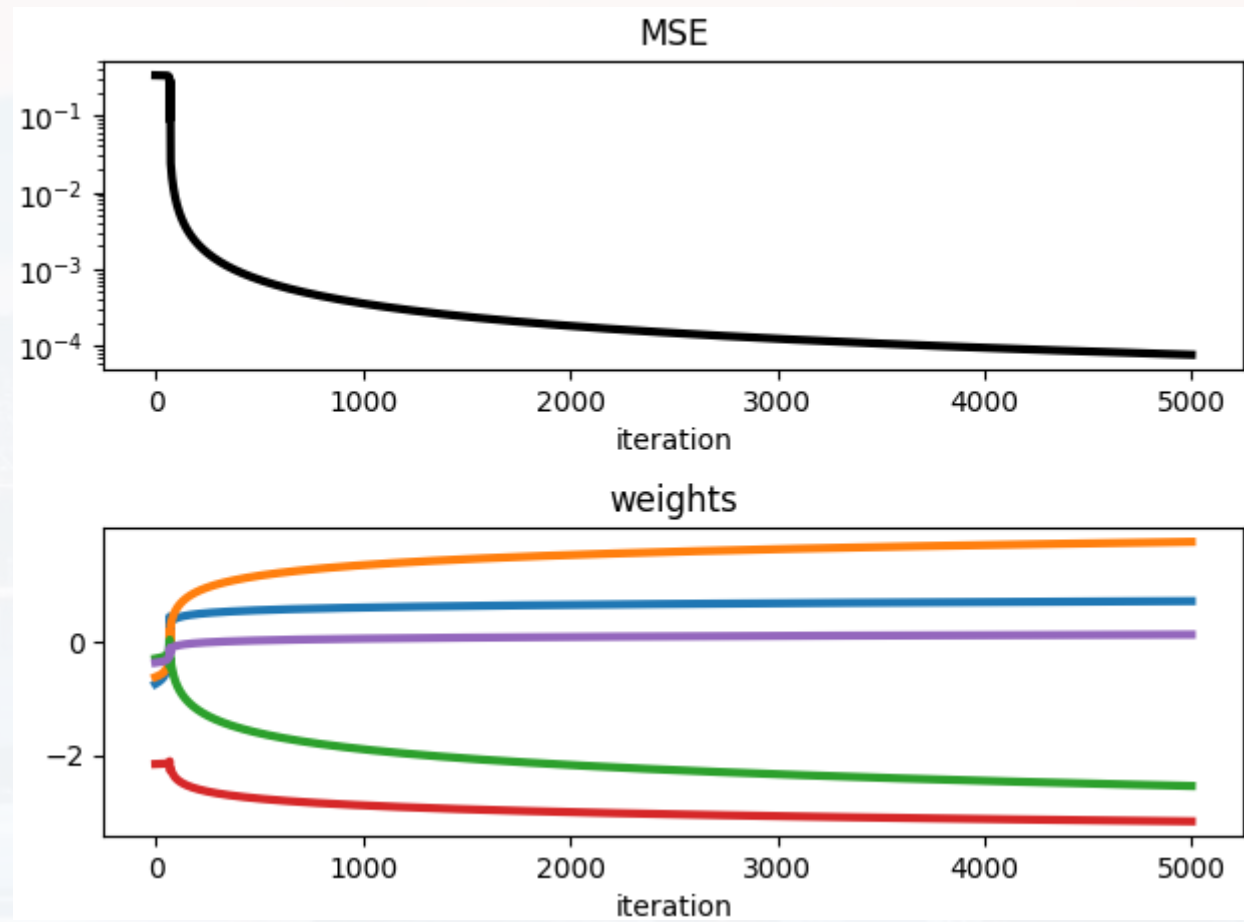
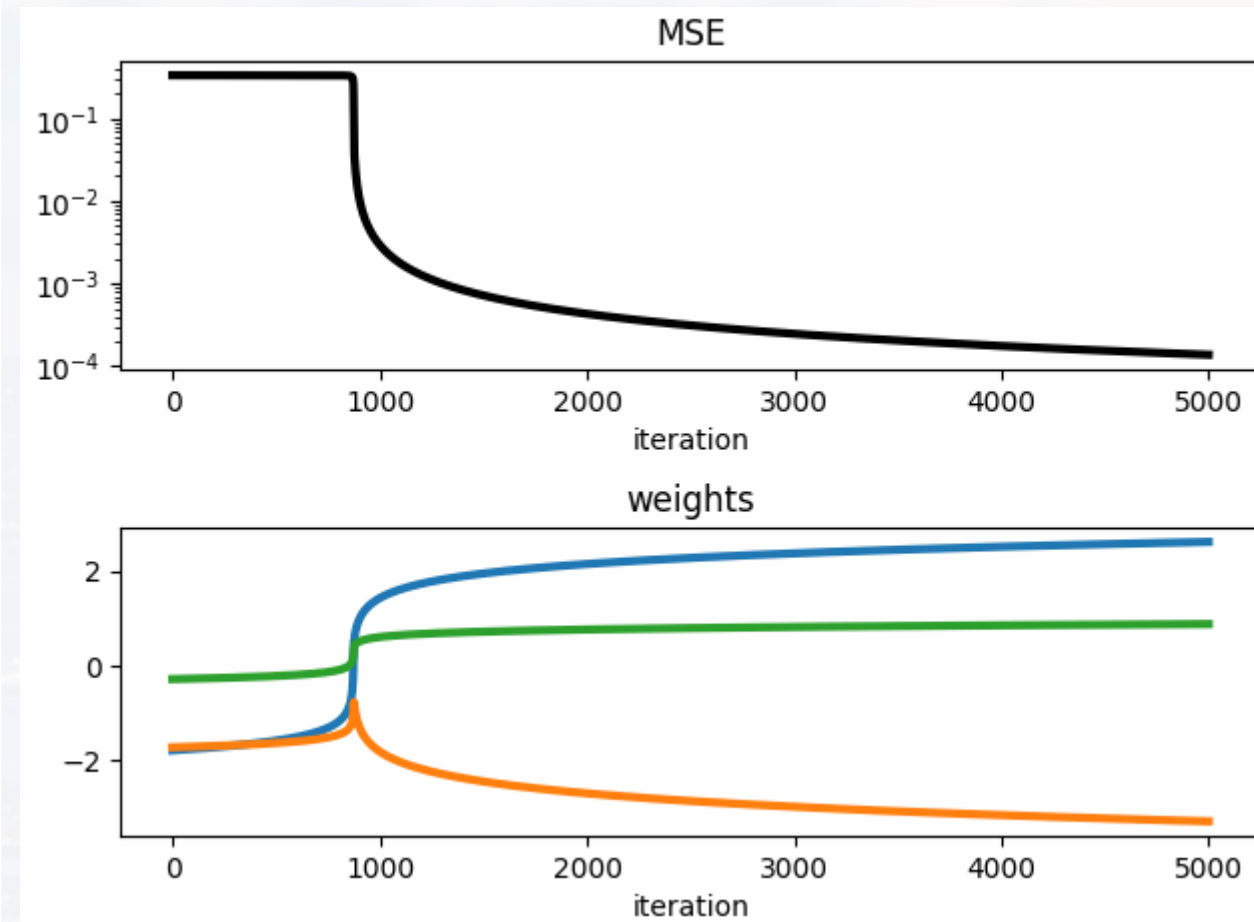
*dot product*





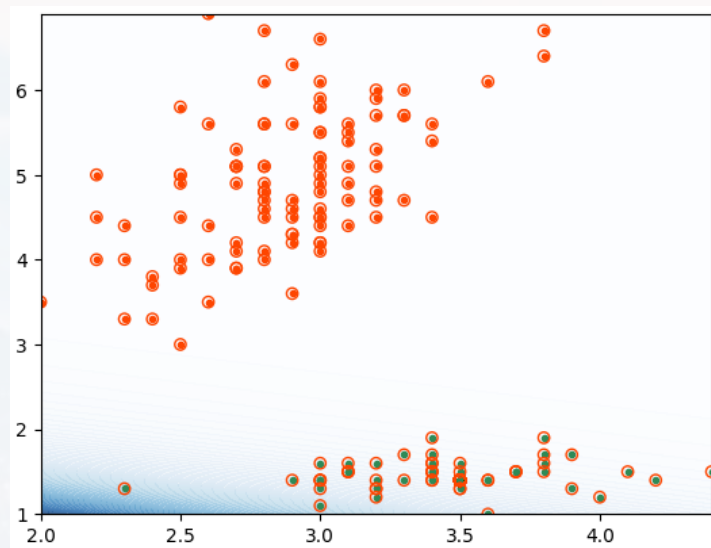


## The training process

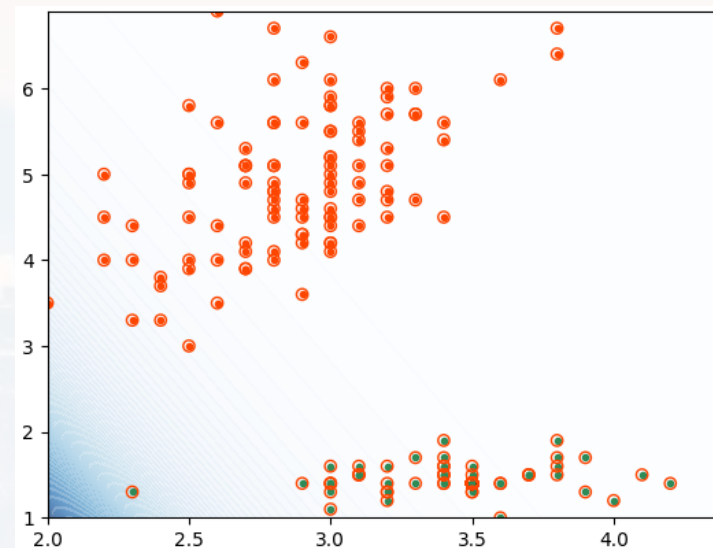




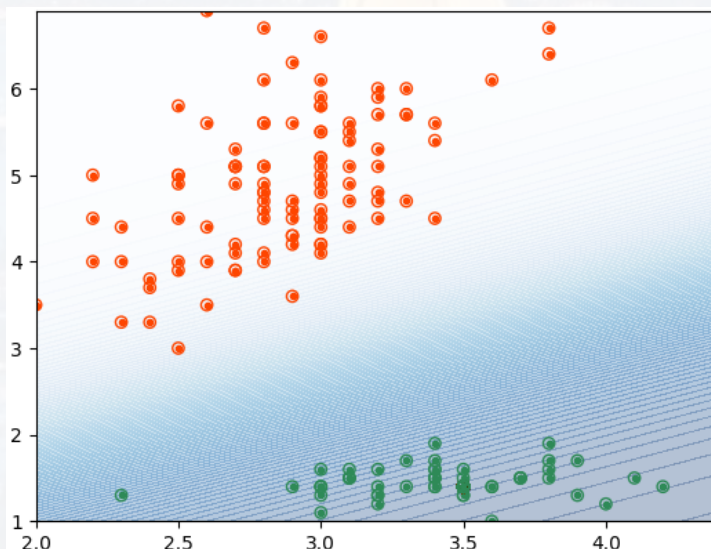
## The training process



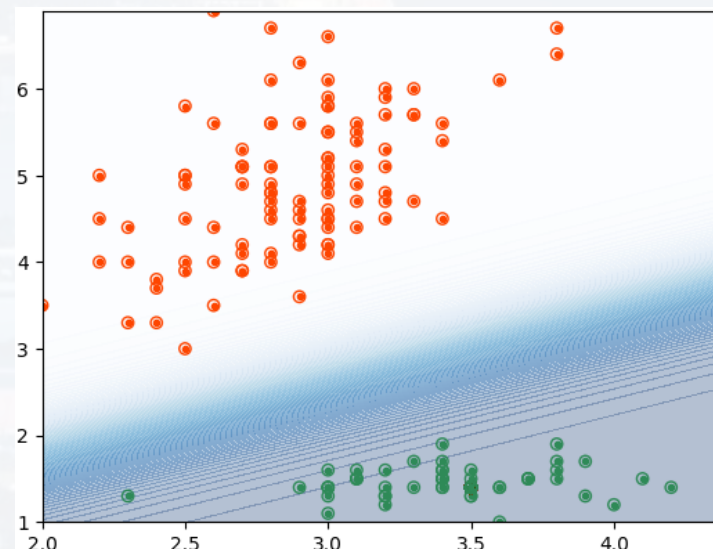
$t = 3$



$t = 5$



$t = 100$



$t = 10,000$

Thank you for your attention!

