Lecture 07:

Introduction to Artificial Neural Networks (ANN)

Markus Hohle

University California, Berkeley

Machine Learning Algorithms

MSSE 277B, 3 Units

Course Map

classic ML tools & algorithms

ANNs/AI/Deep Learning
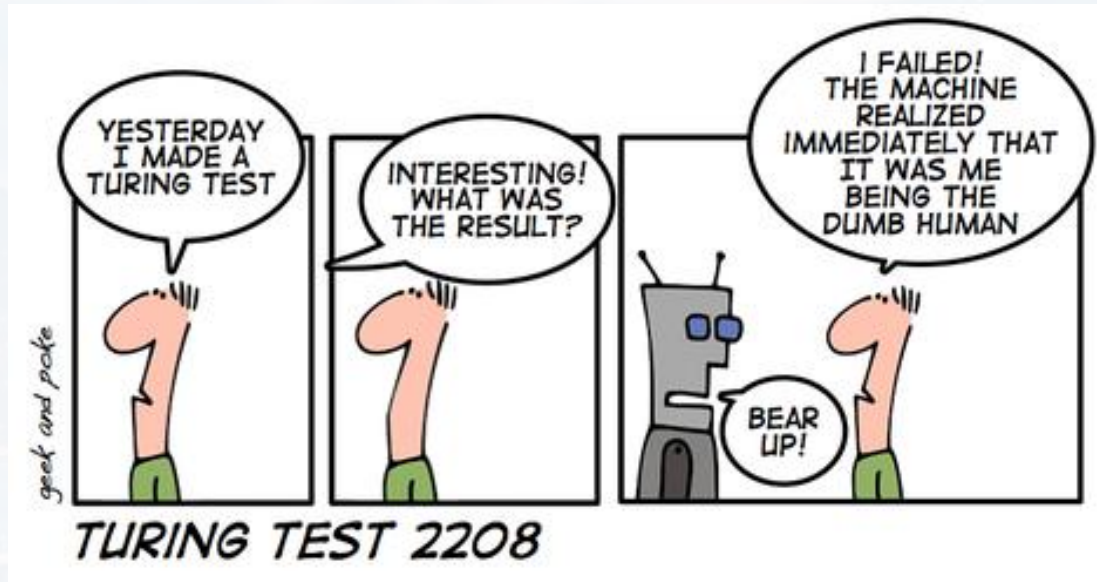
Outline

- **Motivation & Overview**

- **The Perceptron aka Neuron**

- **Building a Perceptron**
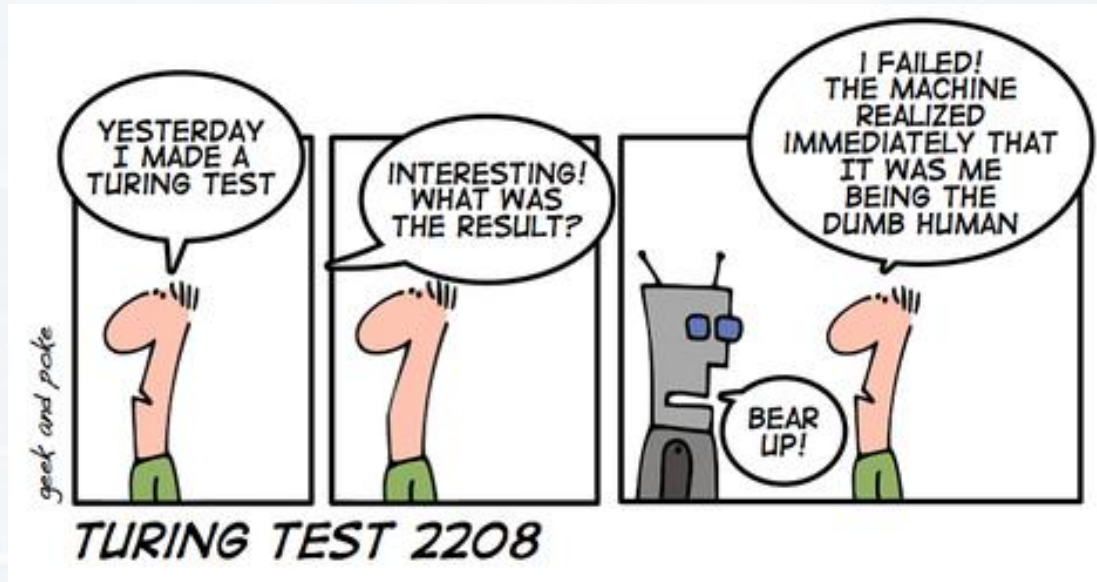
Outline

- **Motivation & Overview**

- The Perceptron aka Neuron

- Building a Perceptron

Why Artificial Neural Networks?

source: Mathworks
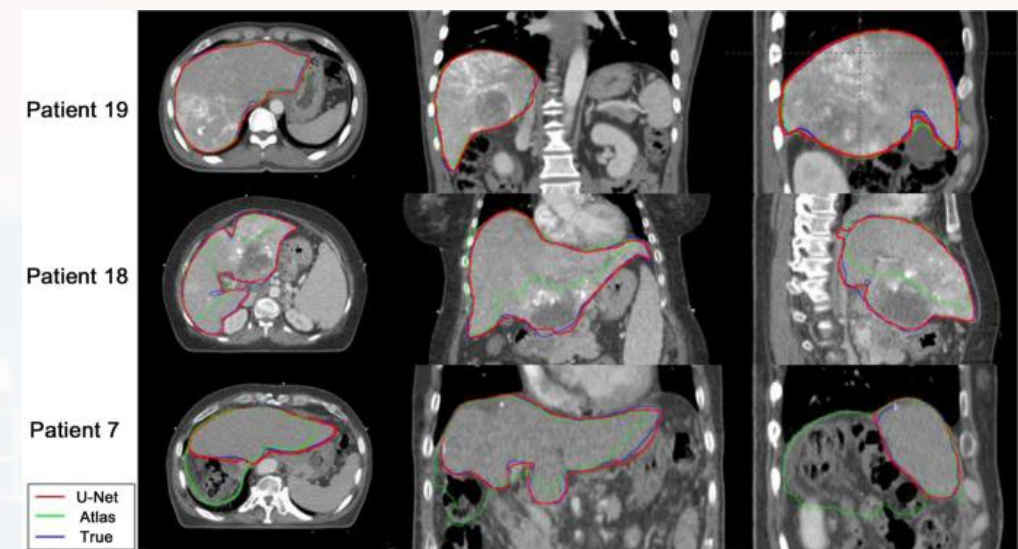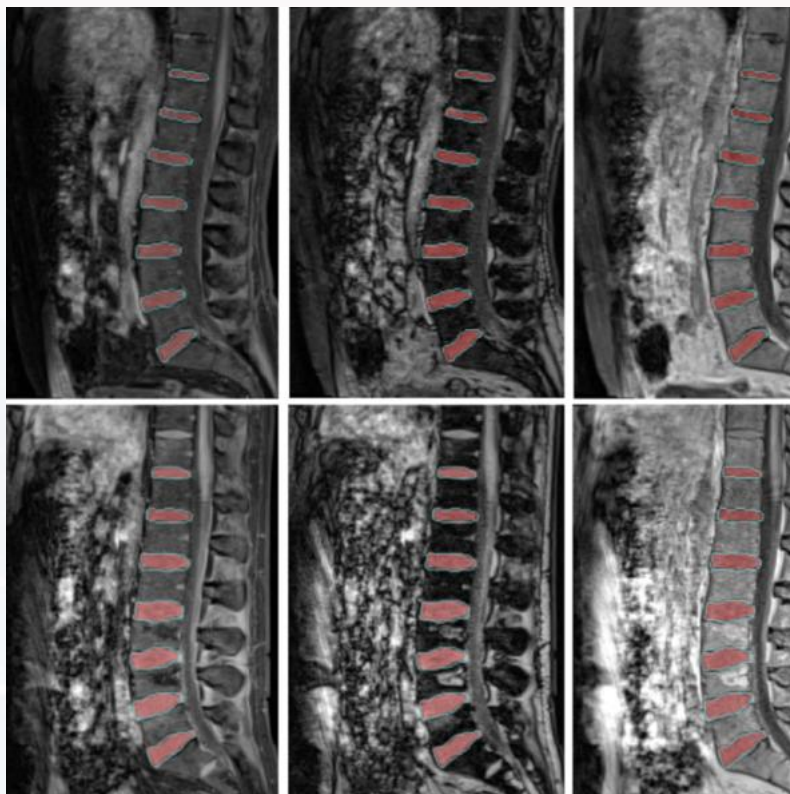
Why Artificial Neural Networks?

S: 我 就 取 钱 给 了 她们
   i   *will*   *get*   *money*   *to*   perf.   *them*

T: i will get the money to them

P(the | get, will, i, 就, 取, 钱, 给, 了)

The man puts a **new tool** on the ground.

He **then** uses a tool to cut the ground an shows the **same product** of the roof .

Why Artificial Neural Networks?

Article

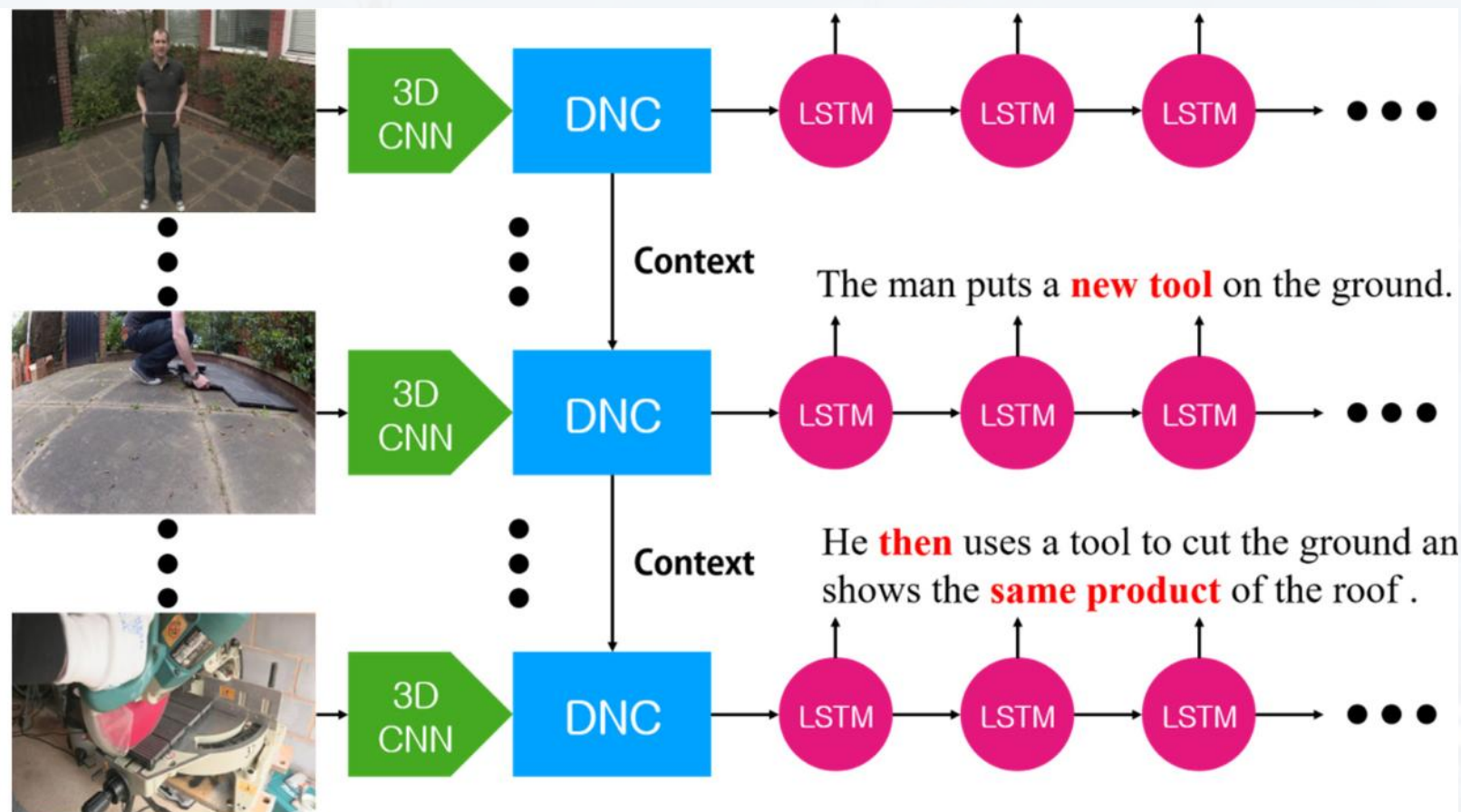## DeepSOCIAL: Social Distancing Monitoring and Infection Risk Assessment in COVID-19 Pandemic

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)
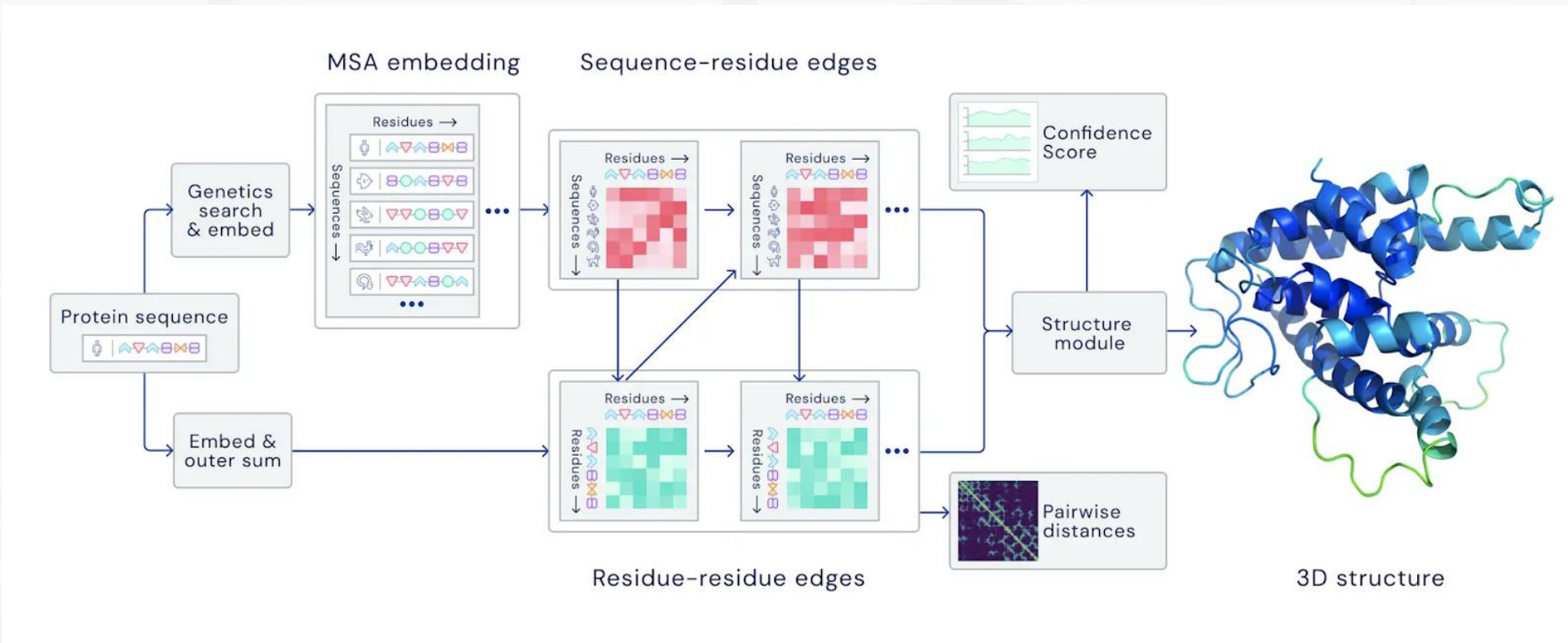
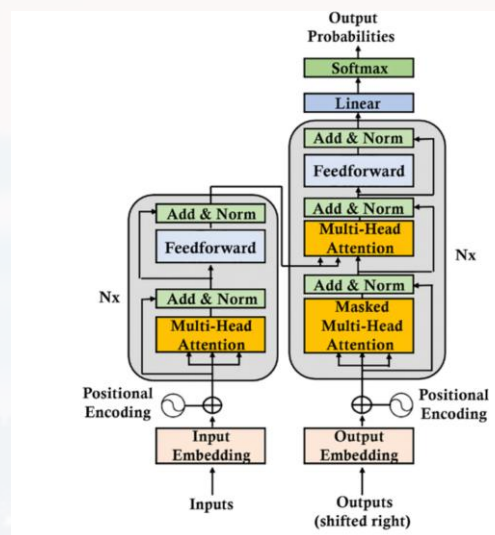**Why Artificial Neural Networks?**

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?

In the next lectures we will learn about:

- classical multi-layer ANN   regression, classification

- **C**onvolution **N**eural **N**etworks   image classification, segmentation, sequence analysis

- **R**ecurrent **N**eural **N**etworks   time series forecasting and classification
- **L**ong-**S**hort-**T**erm-**M**emory ANNs   time series forecasting and classification,
                                                         language processing

- combining the above networks

- **G**raph **N**eural **N**etworks   molecular 3D structures, process/workflow planning

- Transformers   time series analysis, **L**arge **L**anguage **M**odels,
                                             Natural **L**anguage **P**rocessing

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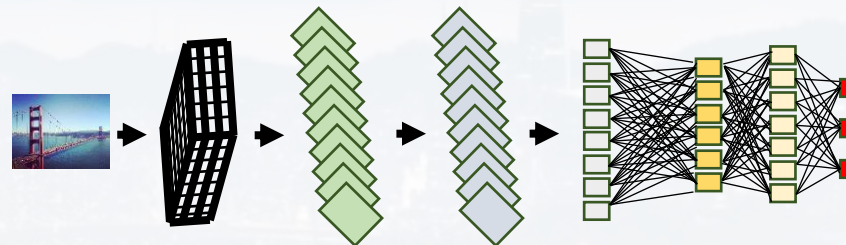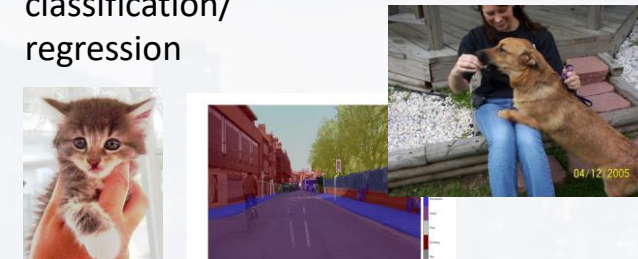   **- Parallel Processing in Python**

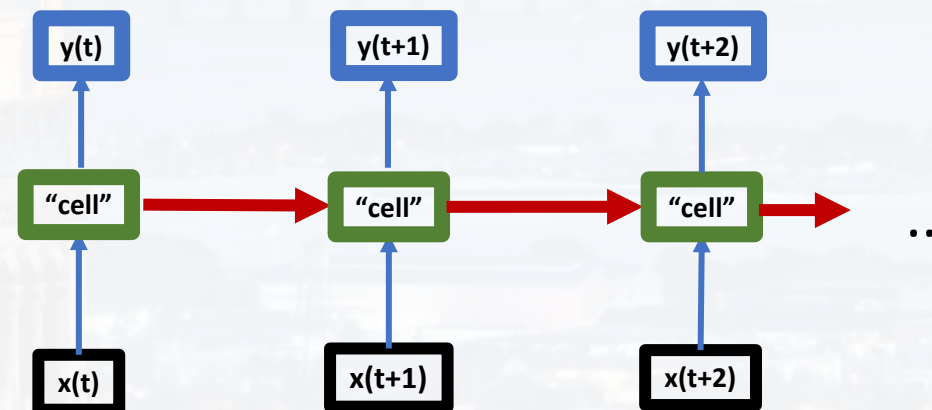fully connected
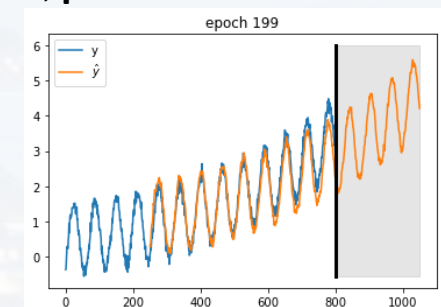(ANN)
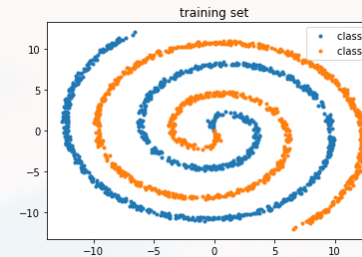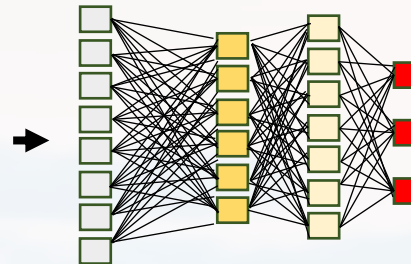
classification/
regression

convolution
(CNN)

classification/
regression

recurrent
(RNN, LSTM)

classification/
Regression, **prediction**

Tutorials and Books
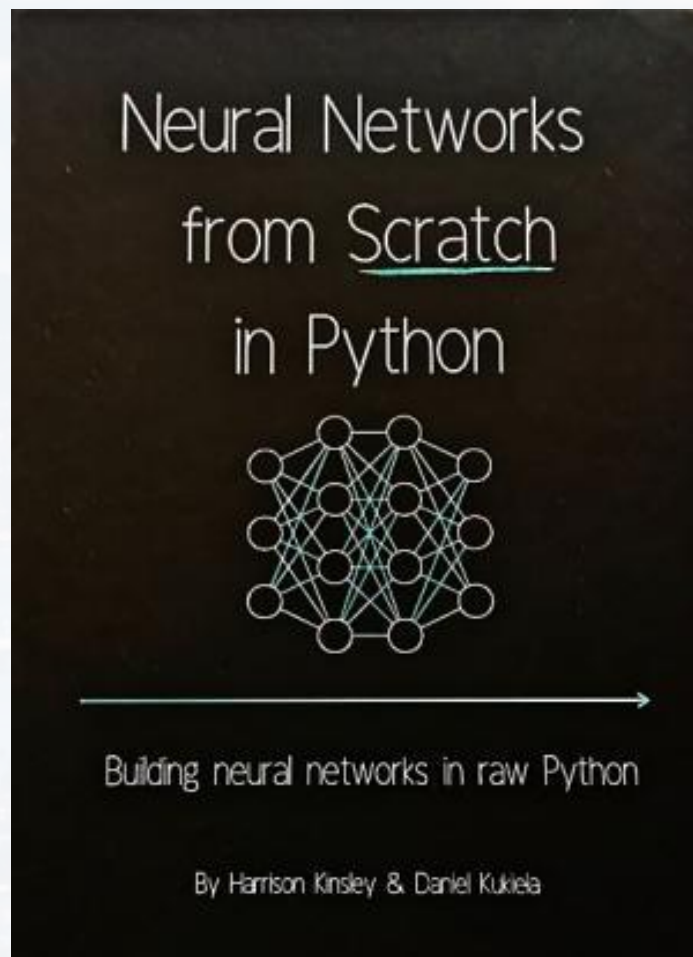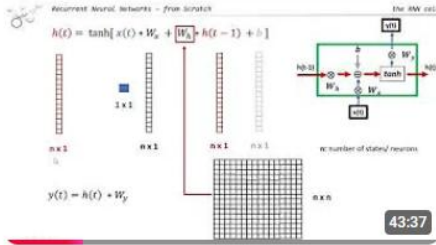
building ANNs, RNNs, LSTMs, CNNs **from scratch**
(only `numpy` and `matplotlib`)

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 and 2 more links

Subscribe

Mısra Turp

@misraturp · 38.2K subscribers · 163 videos

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

misraturp.com/roadmap and 3 more links

Subscribe

TURING TEST 2208

Outline

- Motivation & Overview

- **The Perceptron aka Neuron**

- Building a Perceptron

the perceptron:

I_1

I_2

I_i

out
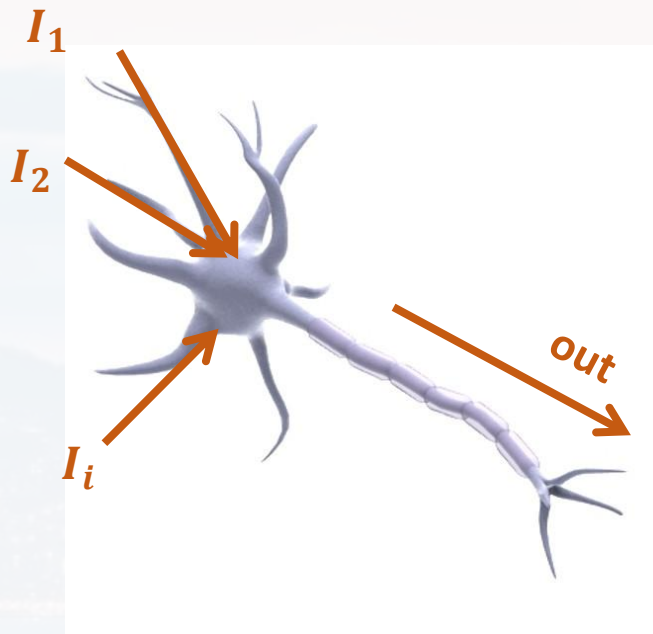
what we know...

- 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)

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

dot product

the perceptron:
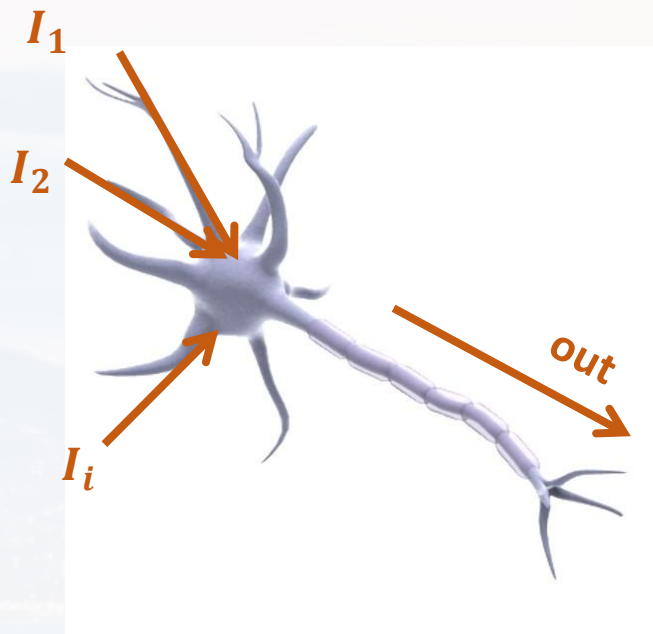
$$net = \sum_i I_i \cdot w_i + b$$

I_i
w_i
b

input *i*
corresponding weight
bias (base potential)

dot product

recall: linear models

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

$$\begin{pmatrix} y_1 \\ ... \\ y_k \\ ... \\ ... \\ y_K \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & ... & x_{1n} & ... & x_{1N} \\ ... & ... & & ... & & & \\ 1 & x_{k1} & & & x_{kn} & & \\ 1 & ... & & & ... & & \\ 1 & x_{K1} & x_{K2} & ... & x_{Kn} & ... & x_{KN} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ ... \\ \beta_n \\ ... \\ \beta_N \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ ... \\ \varepsilon_k \\ ... \\ \varepsilon_K \end{pmatrix}$$

the perceptron:

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

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

dot product
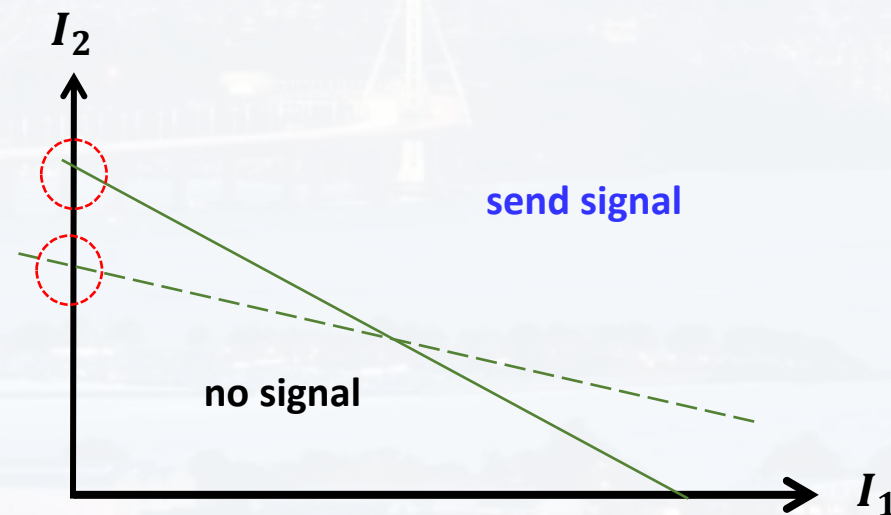
simple example:

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

send signal

no signal

the perceptron:

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

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

out

net → some activation function f → actual output

- y = arctan(net) $(-\infty; +\infty) \rightarrow (-\pi/2; +\pi/2)$

- y = sigm(net) $(-\infty; +\infty) \rightarrow (0; 1)$

- y = ReLU(net) $(-\infty; +\infty) \rightarrow (0; +\infty)$
= *max(0, net)*

the perceptron:

I_1

I_2

out

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

I_i	input i
w_i	corresponding weight
b	bias (base potential)
f	activation function

$$y = f(net)$$

learning:

activation function f objective function E

$$net = \sum_i I_i \cdot w_i + b \longrightarrow y = f(net) \longrightarrow E = \frac{1}{2}(t - y)^2$$

target output t

finding best w_i by **minimizing** E

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

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial net} \frac{\partial net}{\partial w_i}$$ chain rule
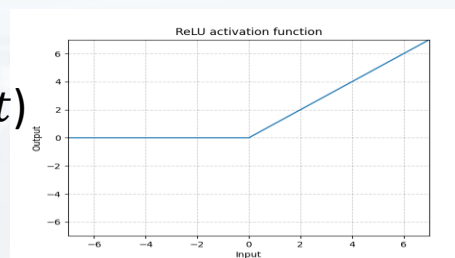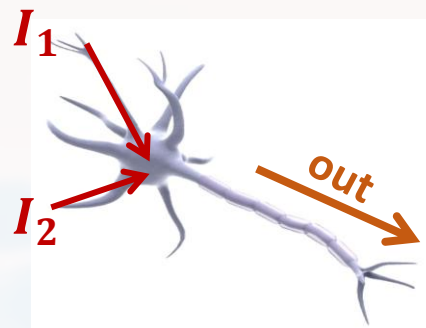
the perceptron:

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

I_i	input i
w_i	corresponding weight
b	bias (base potential)
f	activation function

$$y = f(net)$$

learning:

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

target output t

finding best w_i by **minimizing** E

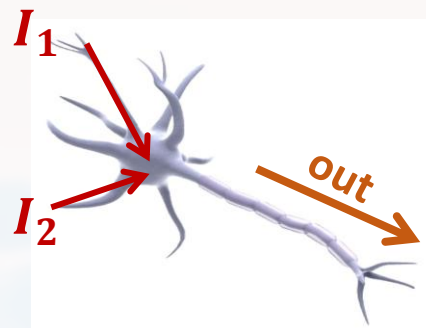$$\Delta w_i = w_i(new) - w_i(old) = -\alpha \frac{\partial E}{\partial w_i} \qquad \text{gradient descent with learning rate } \alpha$$

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

learning:

activation function f

objective function E

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

$$y = f(net)$$

$$E = \frac{1}{2}(t - y)^2$$

target output t

finding best w_i by **minimizing** E

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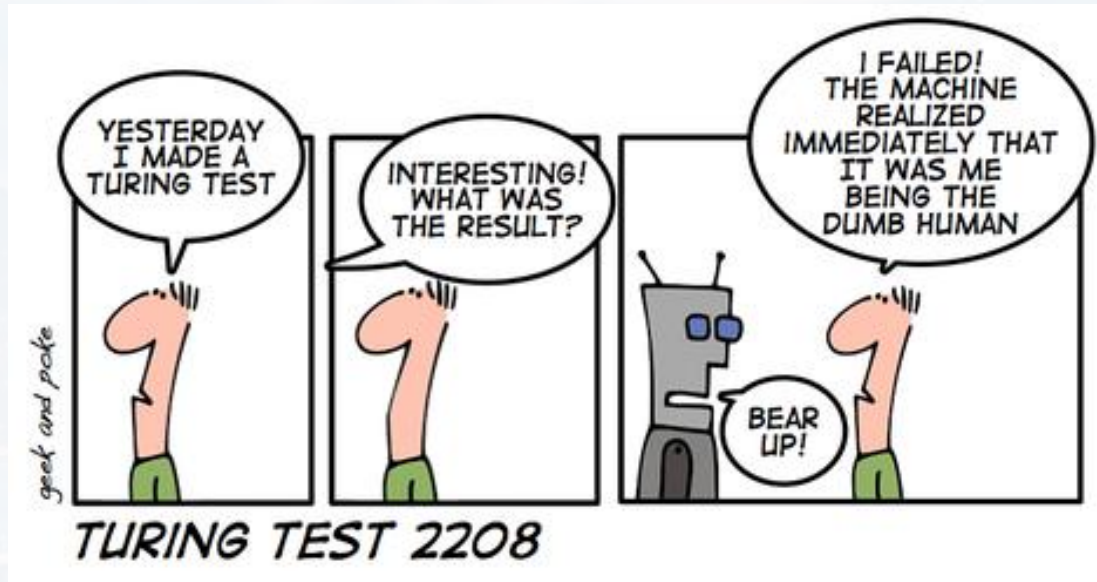
gradient descent with learning rate α

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

inner derivative

outer derivatives

$$f'(net)$$

depends on the activation function

The required change of E propagates back to the changes of w_i → **backpropagation**

learning:

activation function f

objective function E

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

$$y = f(net)$$

$$E = \frac{1}{2}(t - y)^2$$

target output t

finding best w_i by **minimizing** E

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

gradient descent with learning rate α

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

inner derivative   outer derivatives

The required change of E propagates back to the changes of w_i → **backpropagation**

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

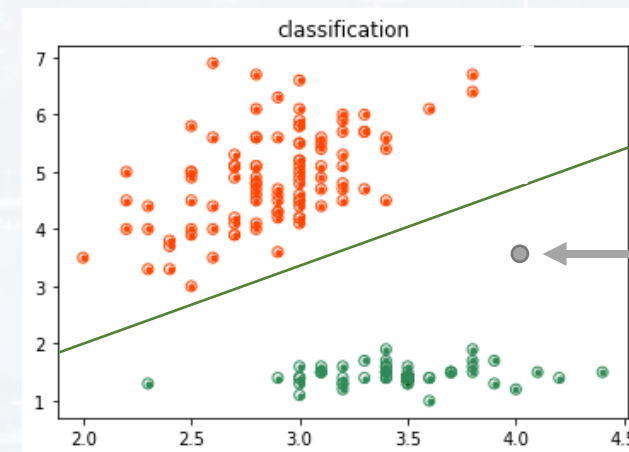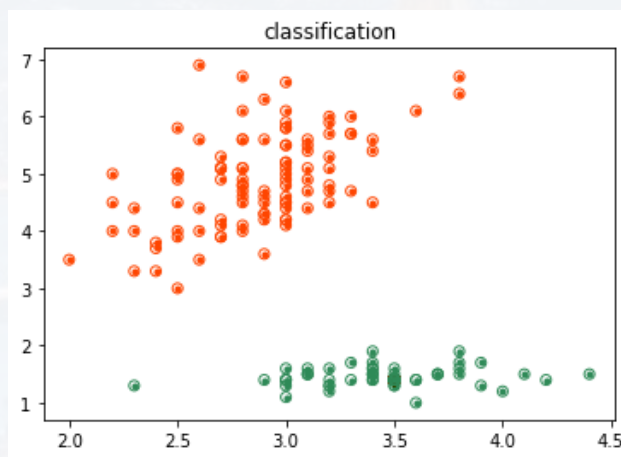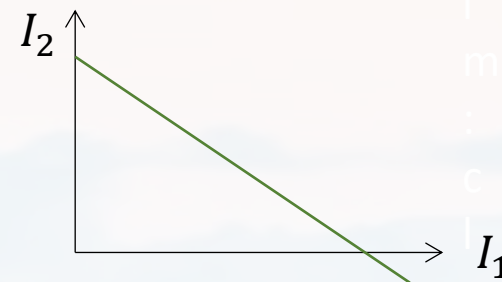$$\Delta b = b(new) - b(old) = \alpha\ (t - y)\ f'(net) \cdot 1$$

Outline

- Motivation & Overview

- The Perceptron aka Neuron

- Building a Perceptron

one neuron

two features

two classes

N data points

I_2

I_1

goal

classification

classifying a new
data point

see the Jupyter Notebook `Perceptron.ipynb` for details

main part of the code:

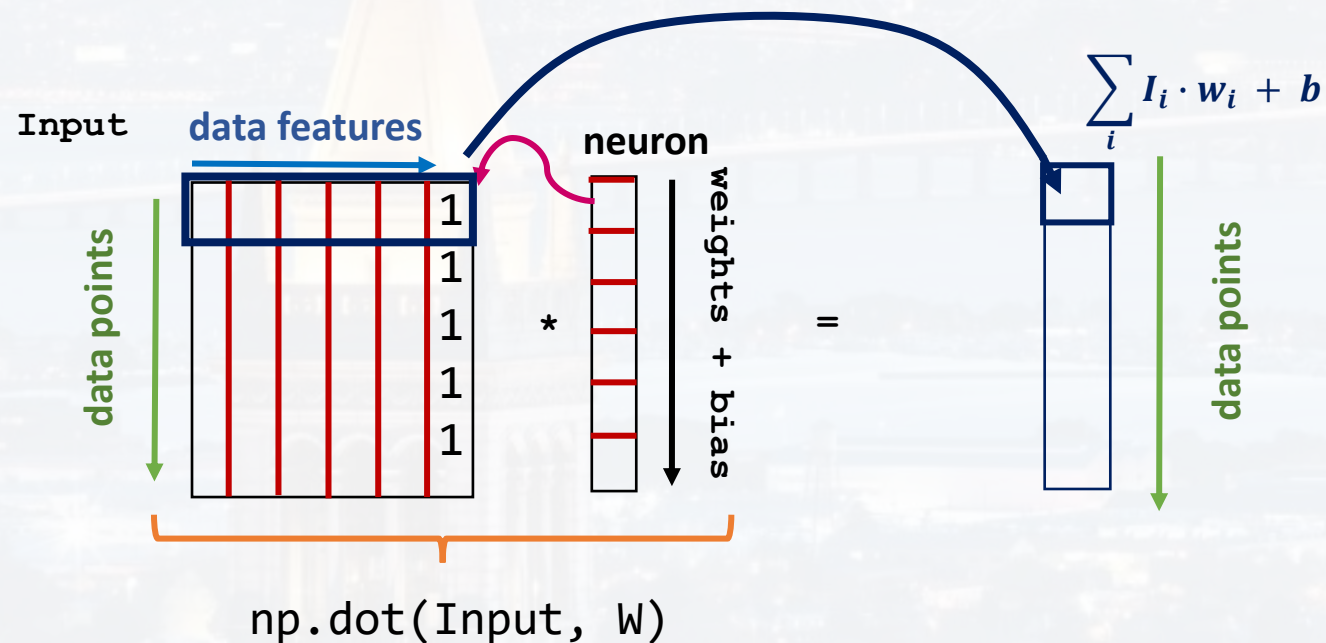$$net = \sum_i I_i \cdot w_i + b$$

dot product

data features

```
[[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]]
```
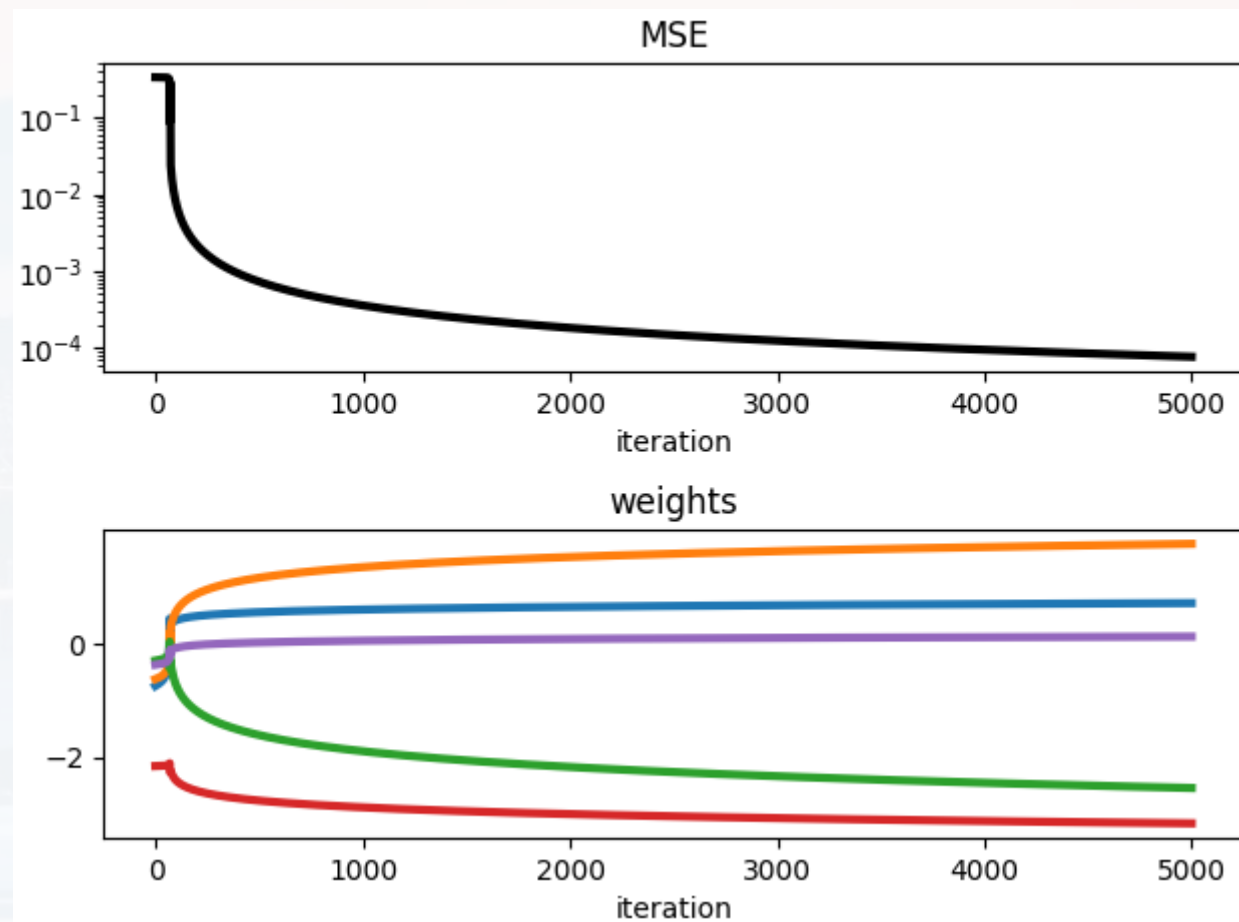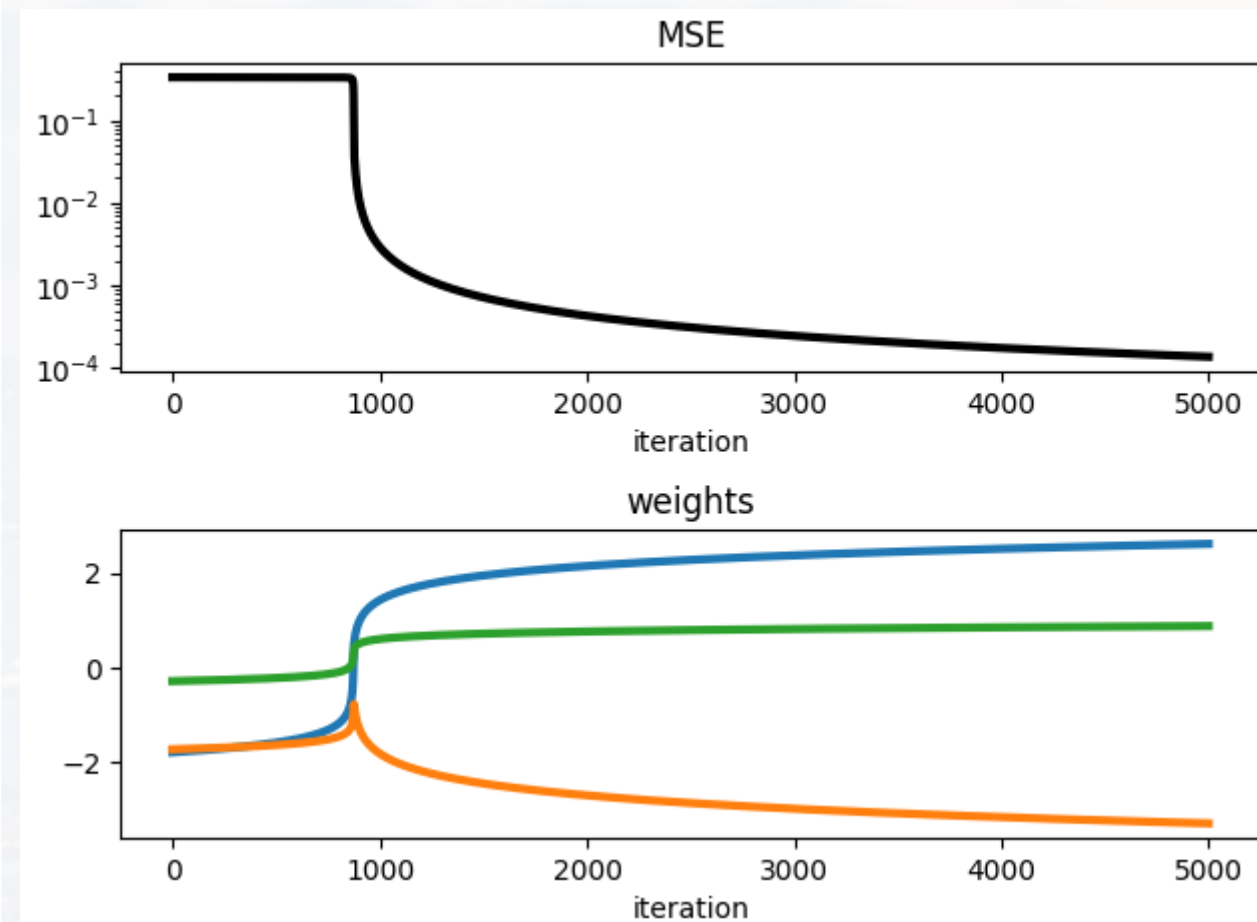
data points

`Input`   **data features**   **neuron**

data points

weights + bias

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

data points

```
np.dot(Input, W)
```

The training process

The training process
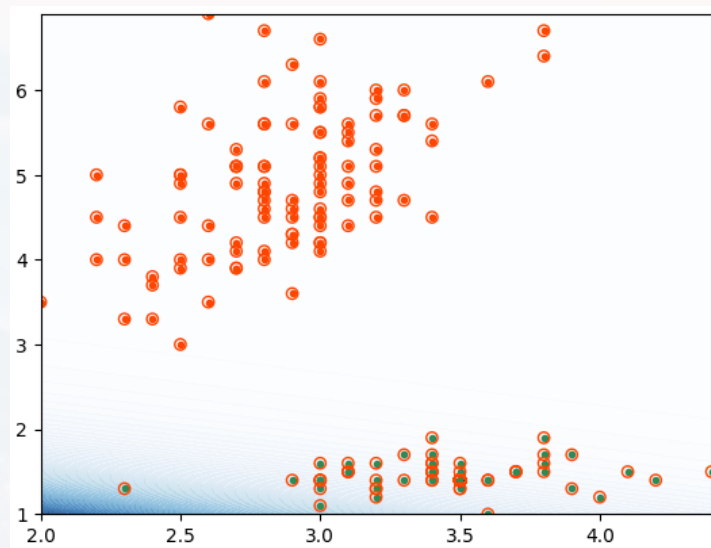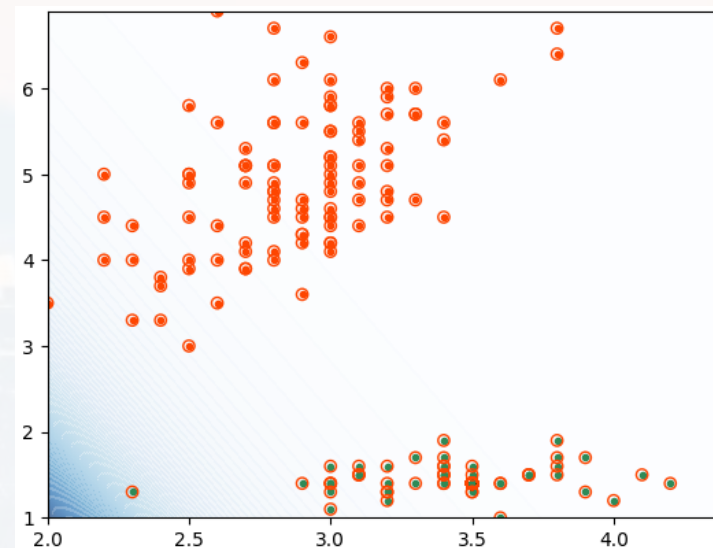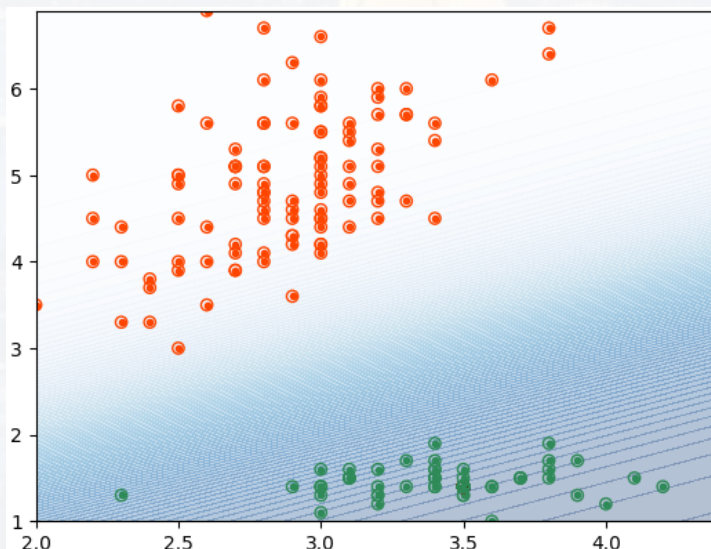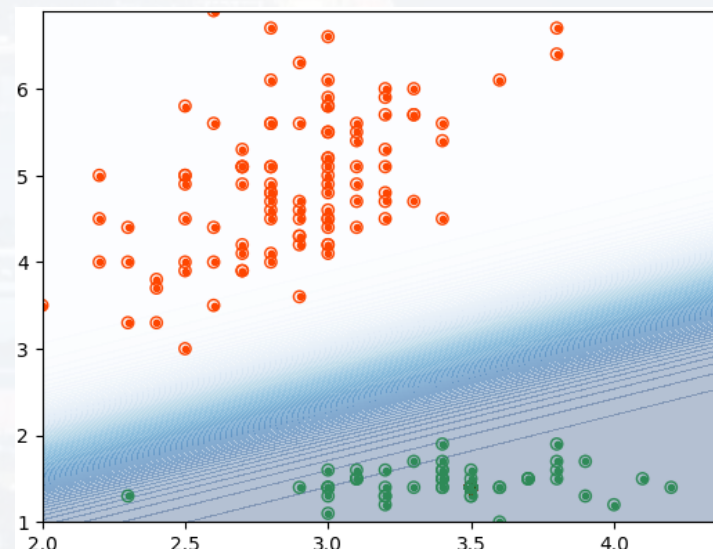
Thank you for your attention!