

Lecture 07:

Introduction to Artificial Neural Networks (ANN)



Markus Hohle

University California, Berkeley

Machine Learning Algorithms

MSSE 277B, 3 Units



Lecture 1: Course Overview and Introduction to Machine Learning

Lecture 2: Bayesian Methods in Machine Learning

classic ML tools & algorithms

Lecture 3: Dimensionality Reduction: Principal Component Analysis

Lecture 4: Linear and Non-linear Regression and Classification

Lecture 5: Unsupervised Learning: K-Means, GMM, Trees

Lecture 6: Adaptive Learning and Gradient Descent Optimization Algorithms

Lecture 7: Introduction to Artificial Neural Networks - The Perceptron

ANNs/AI/Deep Learning

Lecture 8: Introduction to Artificial Neural Networks - Building Multiple Dense Layers

Lecture 9: Convolutional Neural Networks (CNNs) - Part I

Lecture 10: CNNs - Part II

Lecture 11: Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs)

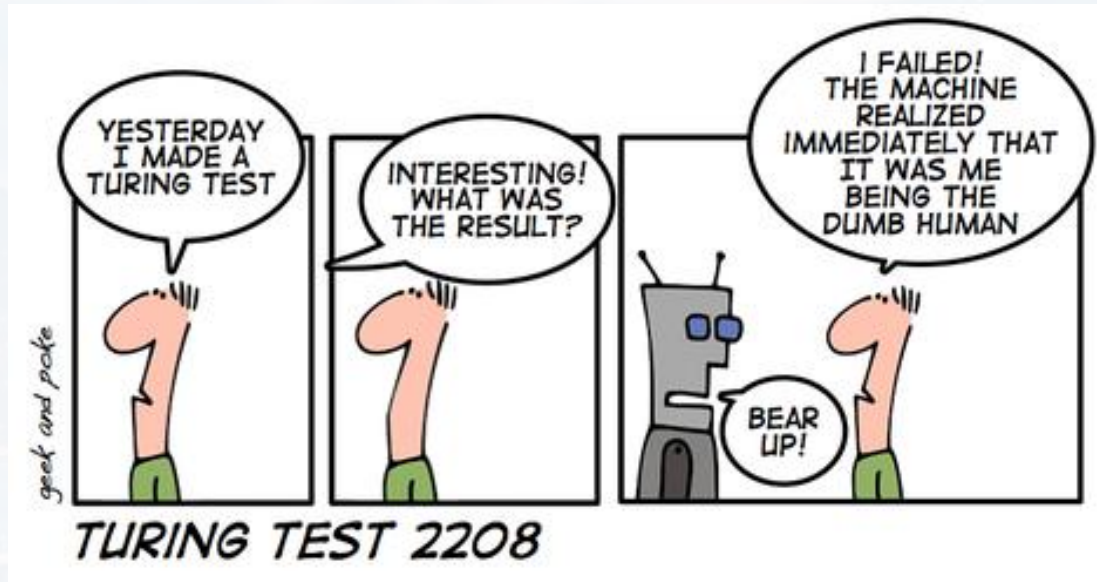
Lecture 12: Combining LSTMs and CNNs

Lecture 13: Running Models on GPUs and Parallel Processing

Lecture 14: Project Presentations

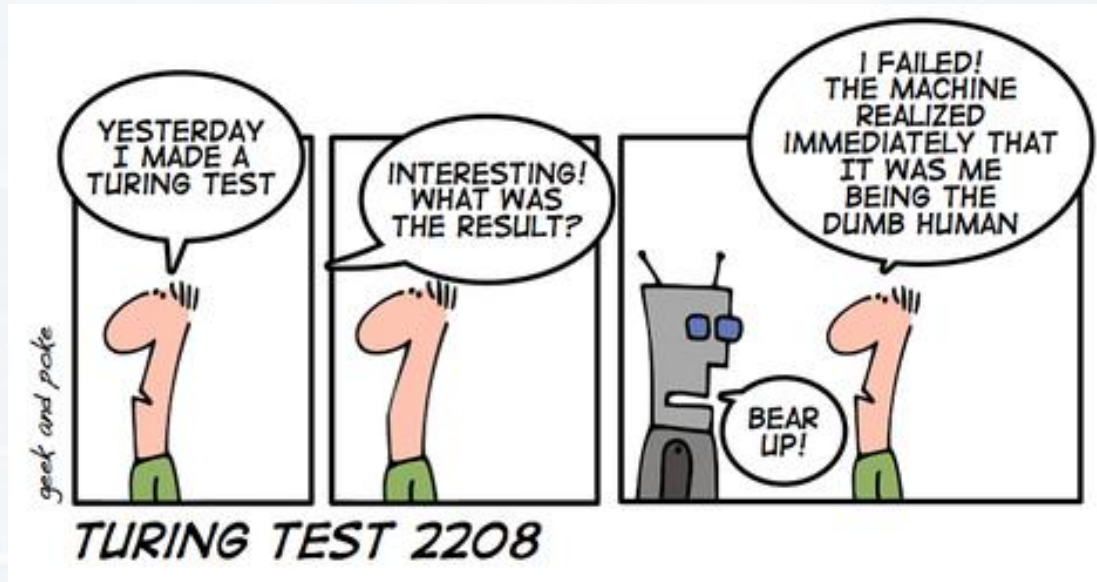
Lecture 15: Transformer

Lecture 16: GNN



Outline

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

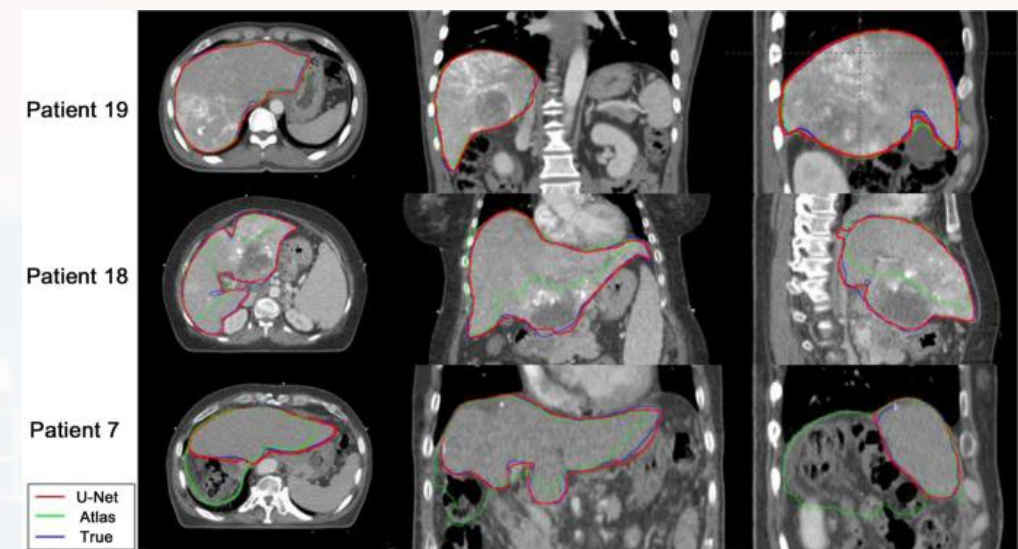
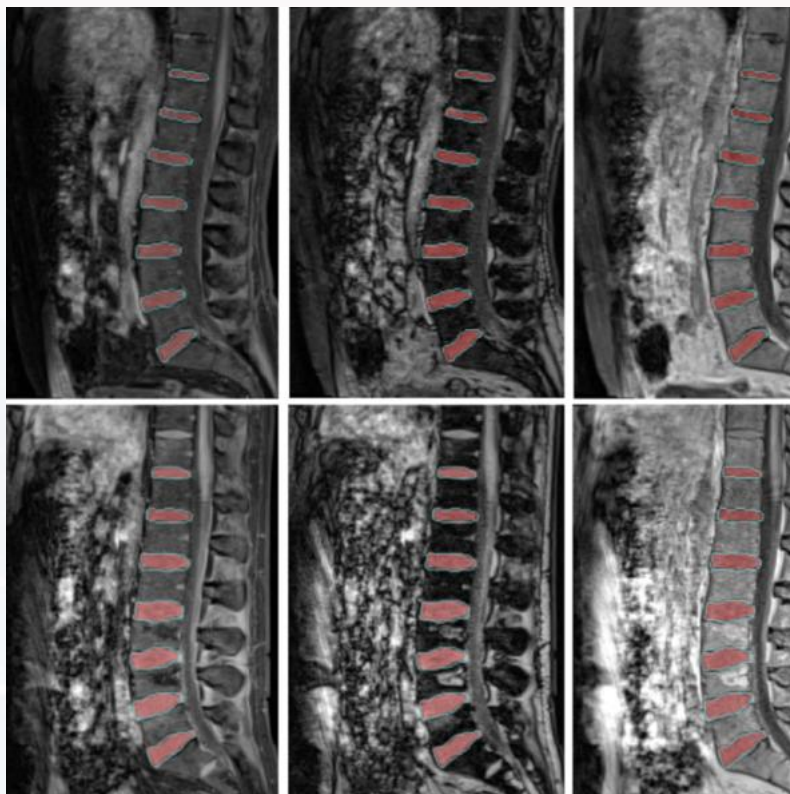


Outline

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



Why Artificial Neural Networks?



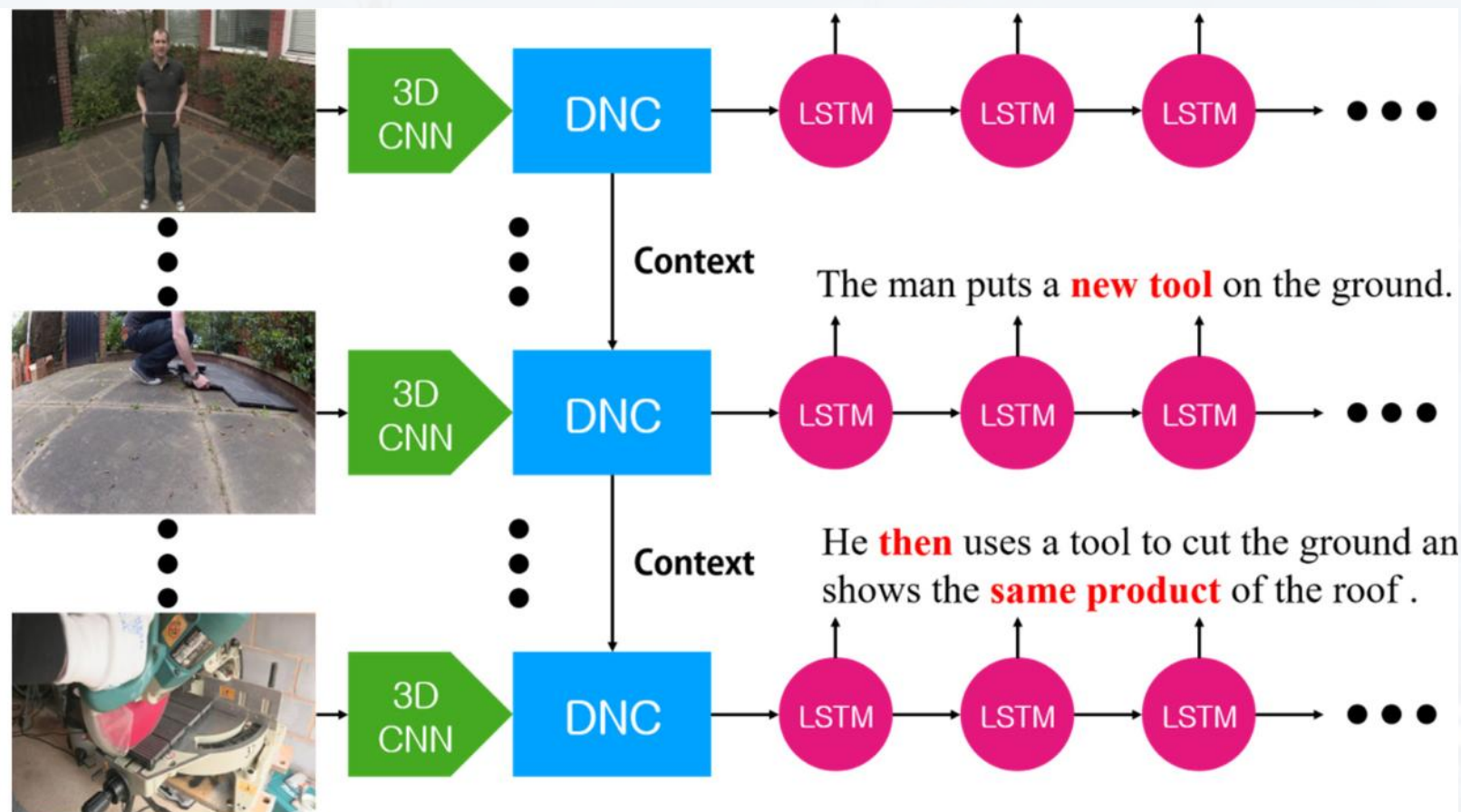


Why Artificial Neural Networks?

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

T: ²i ¹will ⁰get 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)

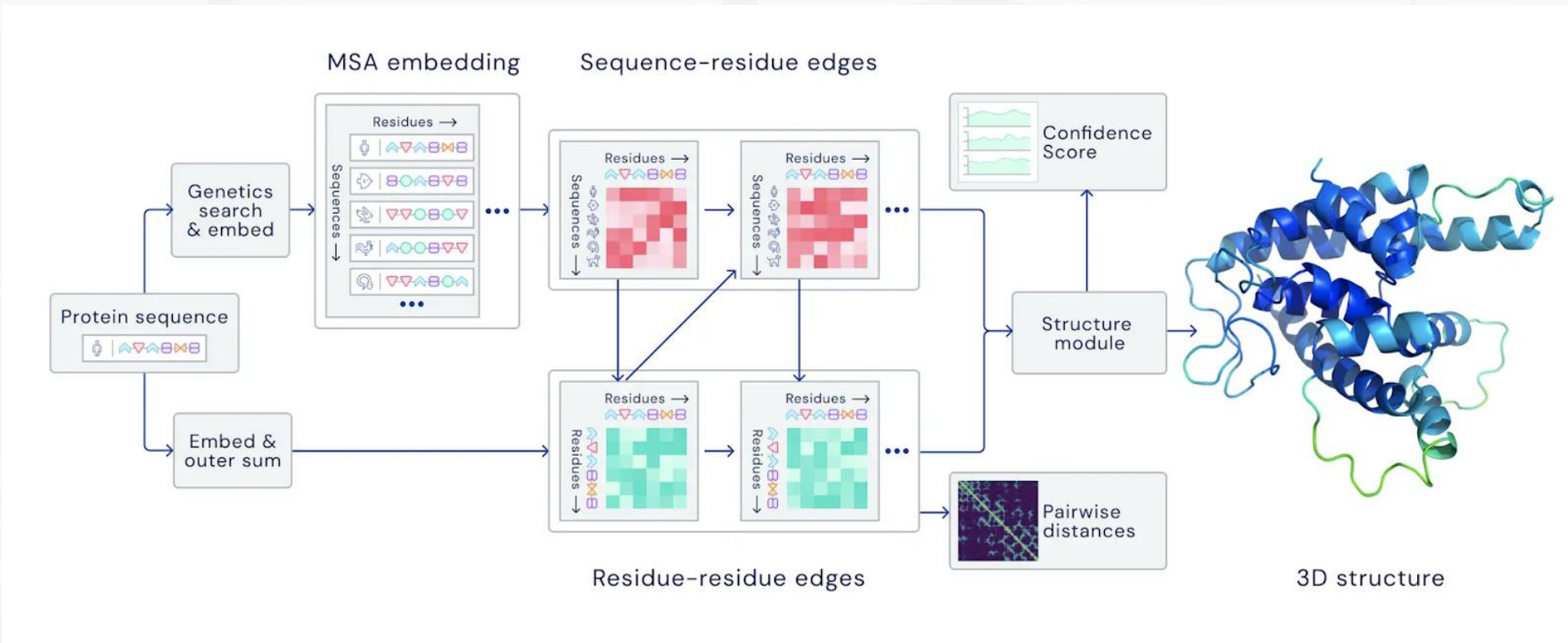


Why Artificial Neural Networks?



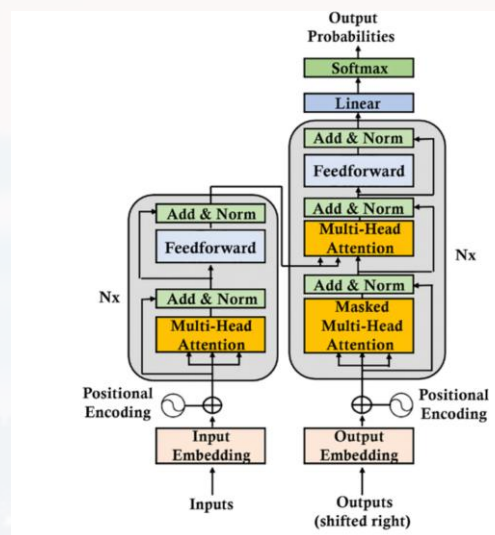
Demis Hassabis
@demishassabis

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

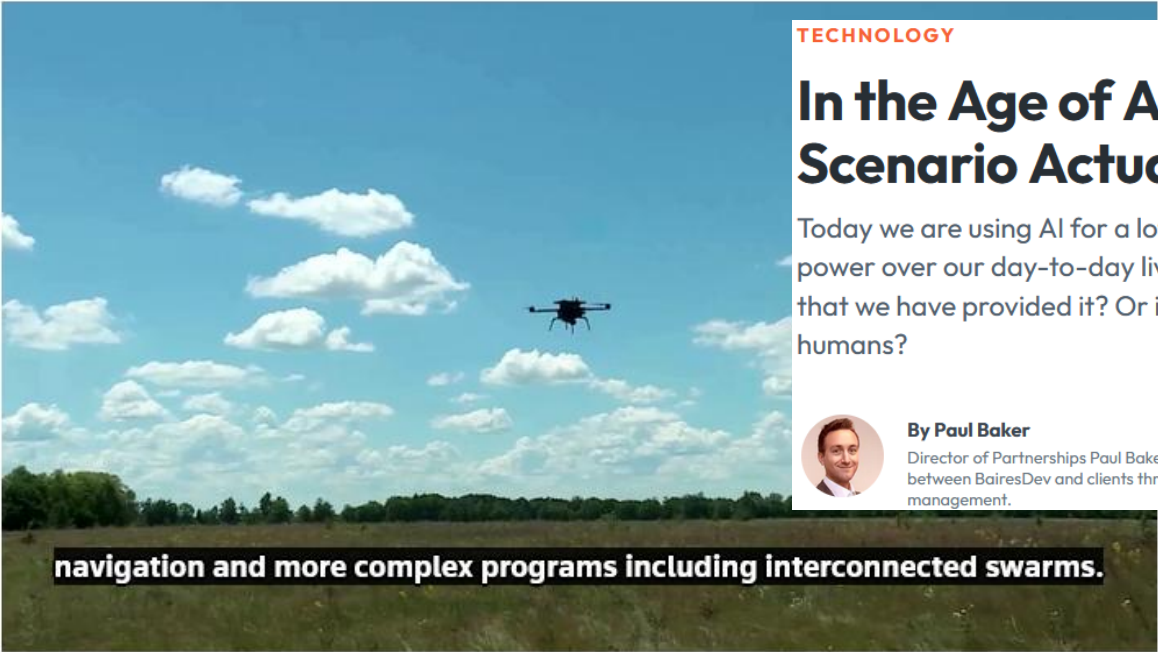
Artificial Intelligence | Ukraine and Russia at War

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



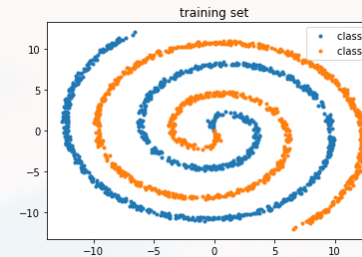
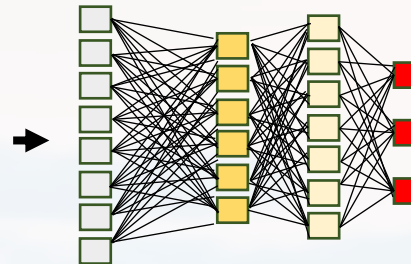
In the next lectures we will learn about:

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

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

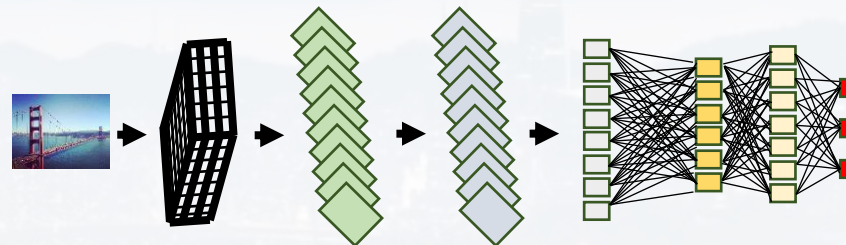


fully connected
(ANN)

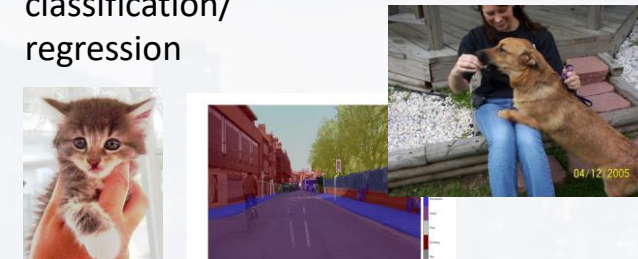


classification/
regression

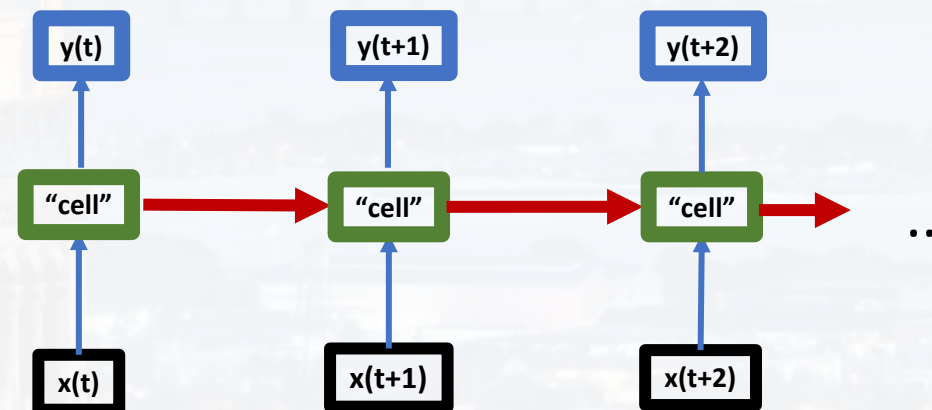
convolution
(CNN)



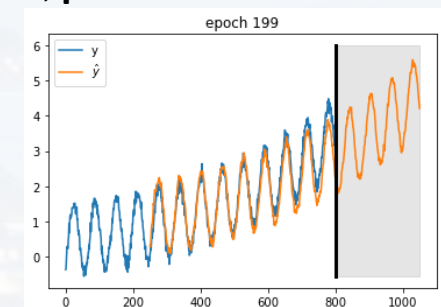
classification/
regression



recurrent
(RNN, LSTM)

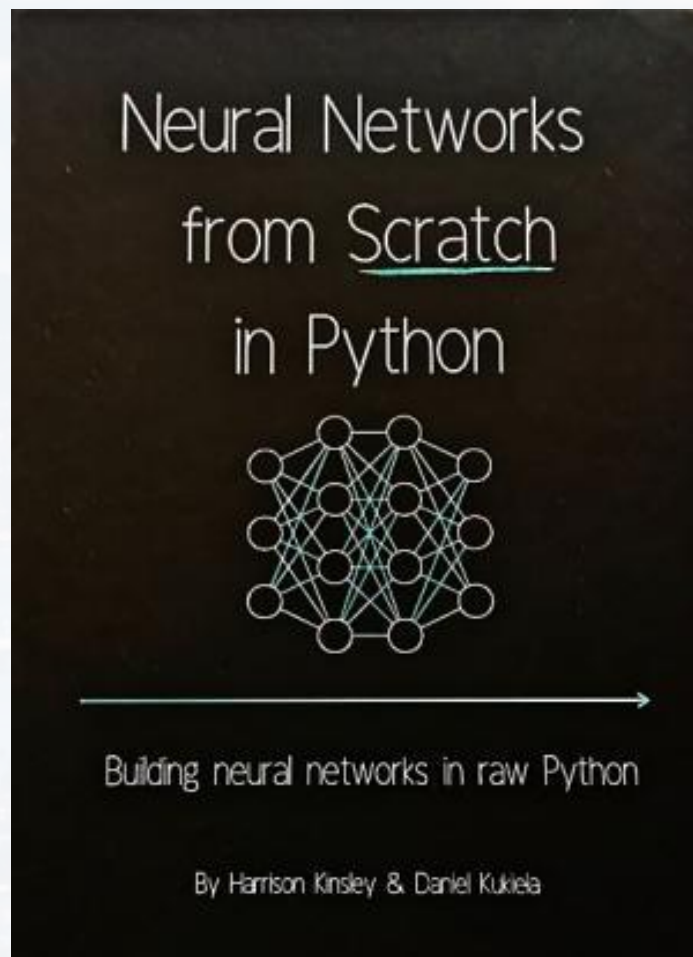



classification/
Regression, **prediction**





Tutorials and Books





Dr Fridolin

@DrFridolin · 227 subscribers · 37 videos

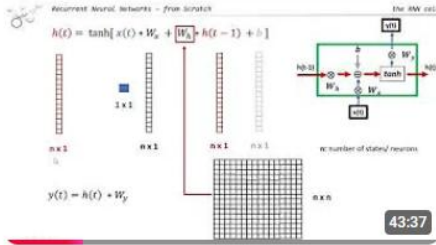
Dr Fridolin shows you how to program AI step by step, line by line without using external libraries. ...more

github.com/DrBigMau

Customize channel Manage videos


Home Videos Playlists Posts

For You




RNN 2 Building a RNN Cell

725 views · 11 months ago



RNN 1 Intro

615 views · 11 months ago



ANN 1 Single Neuron

272 views · 11 months ago

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



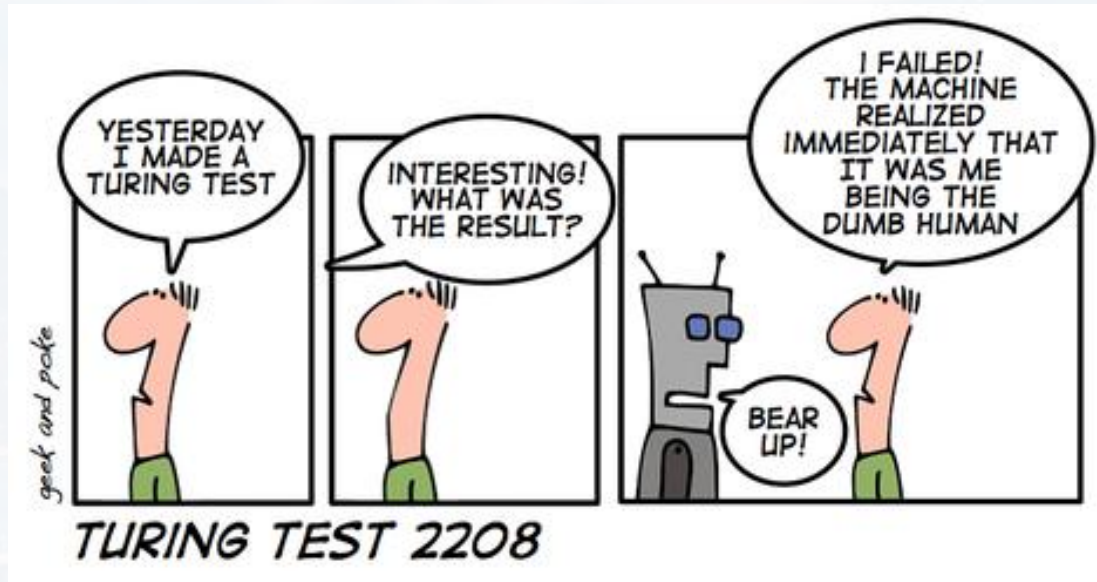
Misra Turp

@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

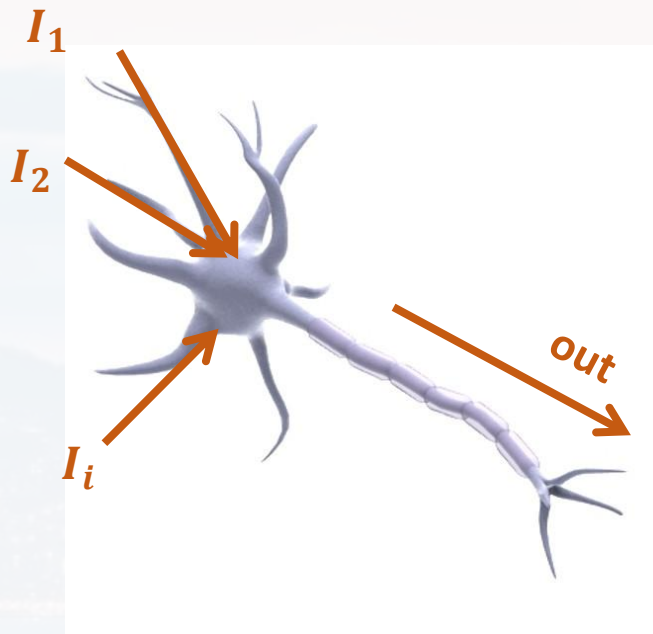


Outline

- 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

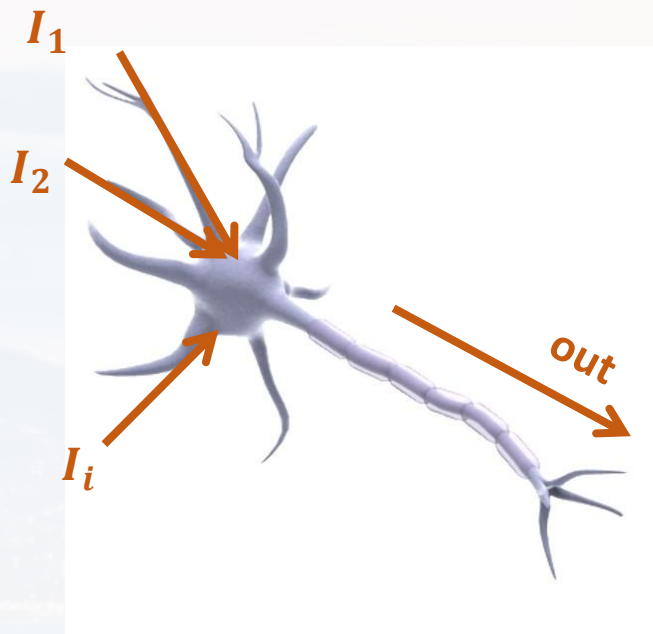
I_i input i
 w_i corresponding weight
 b bias (base potential)

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


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:

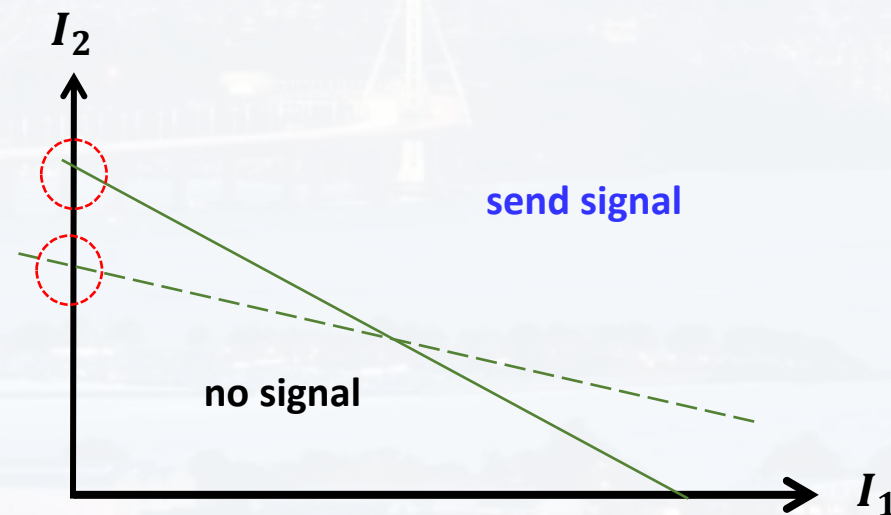
one neuron with a **switch**, threshold **T**
and **two** input channels

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





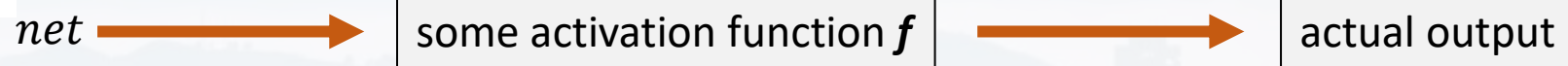
the perceptron:



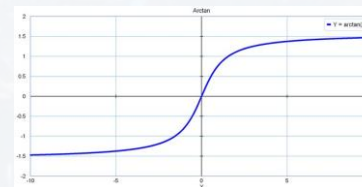
$$net = \sum_i I_i \cdot w_i + b$$

I_i
 w_i
 b

input i
corresponding weight
bias (base potential)



- $y = \arctan(net)$



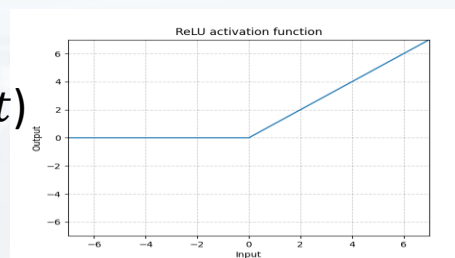
$(-\infty; +\infty) \rightarrow (-\pi/2; +\pi/2)$

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



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

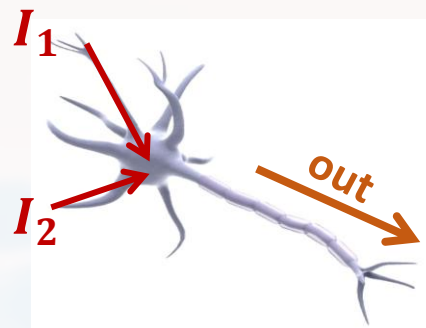
- $y = \text{ReLU}(net)$
 $= \max(0, net)$



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



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

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

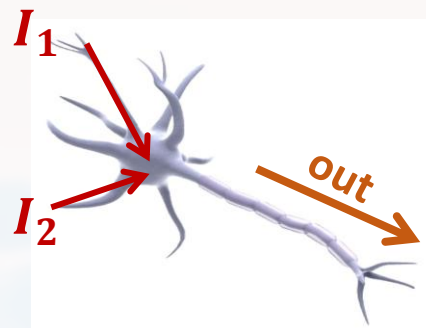
gradient descent with learning rate α

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

chain rule



the perceptron:



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

I_i
 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 α

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



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

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

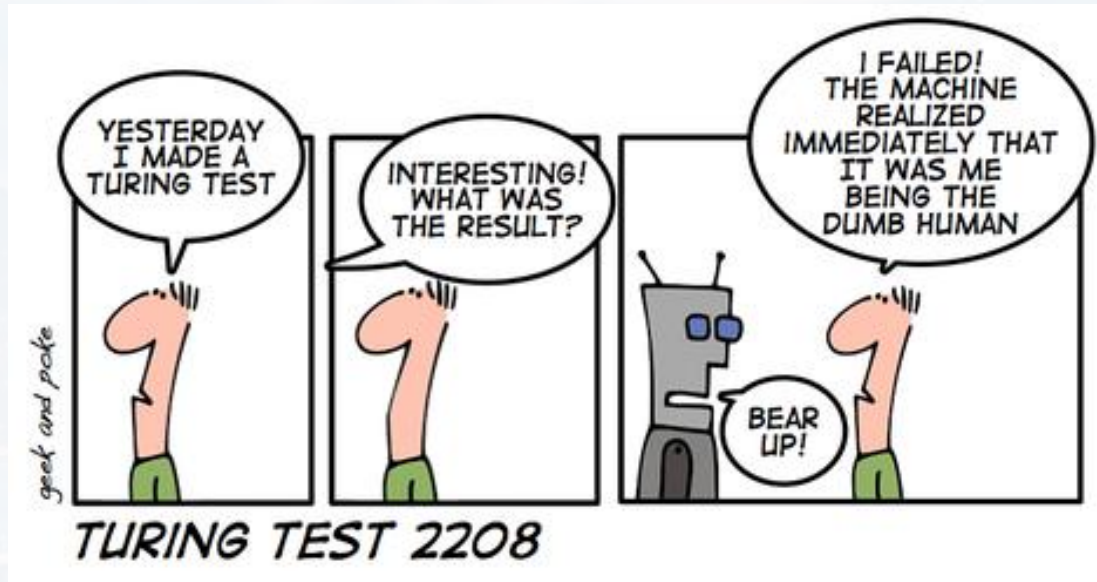
$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial net} \frac{\partial net}{\partial w_i} = \boxed{-(t - y)} \boxed{f'(net)} \boxed{I_i}$$

inner derivative outer derivatives

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

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

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

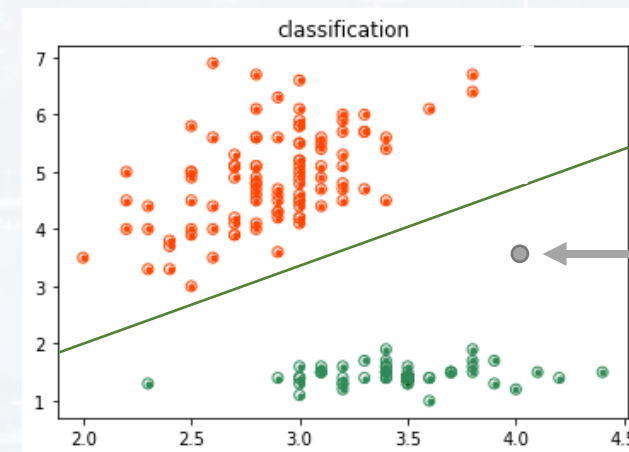
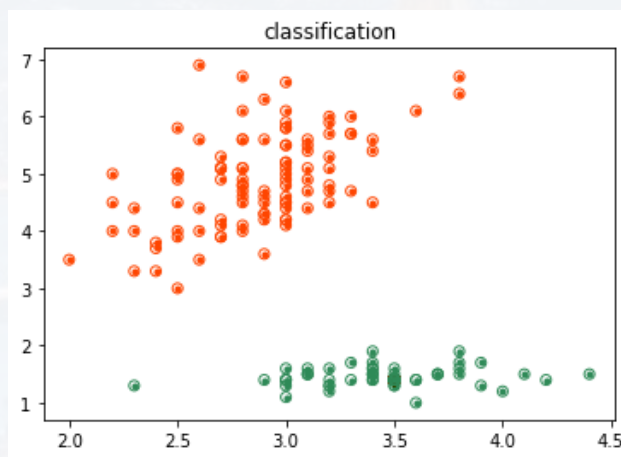
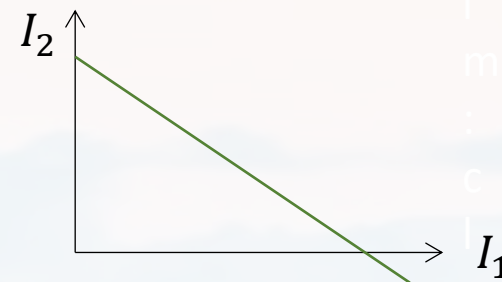


Outline

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



one neuron
two features
two classes
N data points



classifying a new
data point

see the Jupyter Notebook `Perceptron.ipynb` for details



main part of the code:

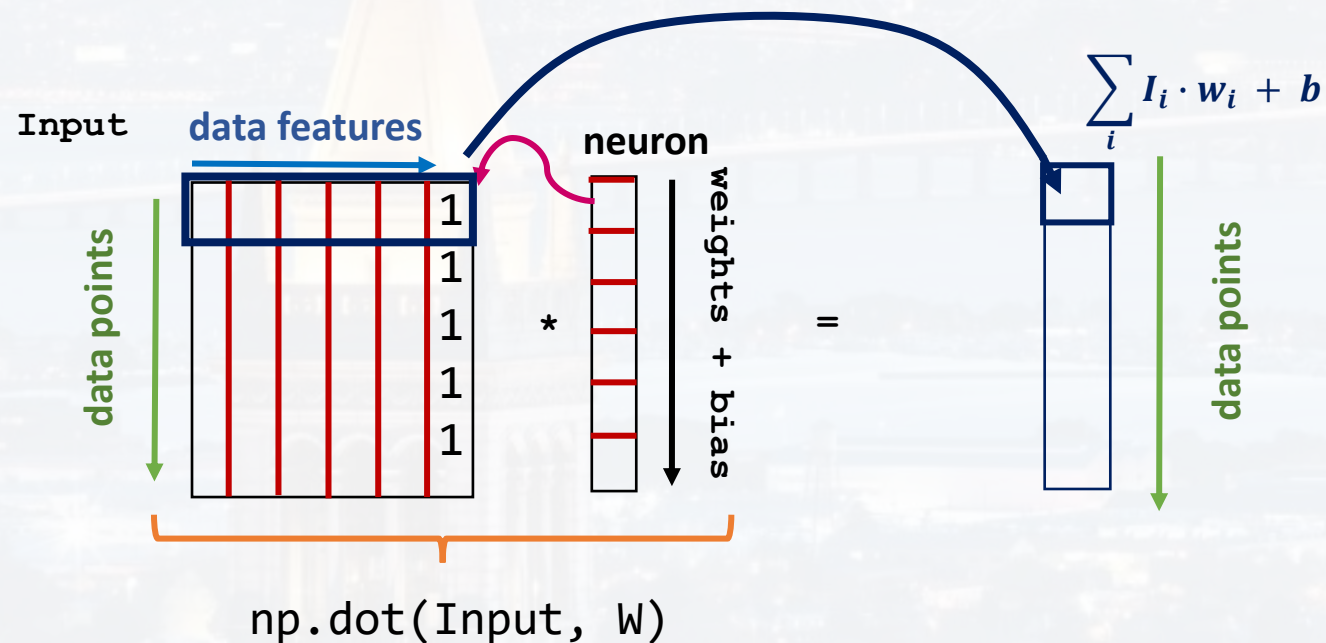
data features

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]

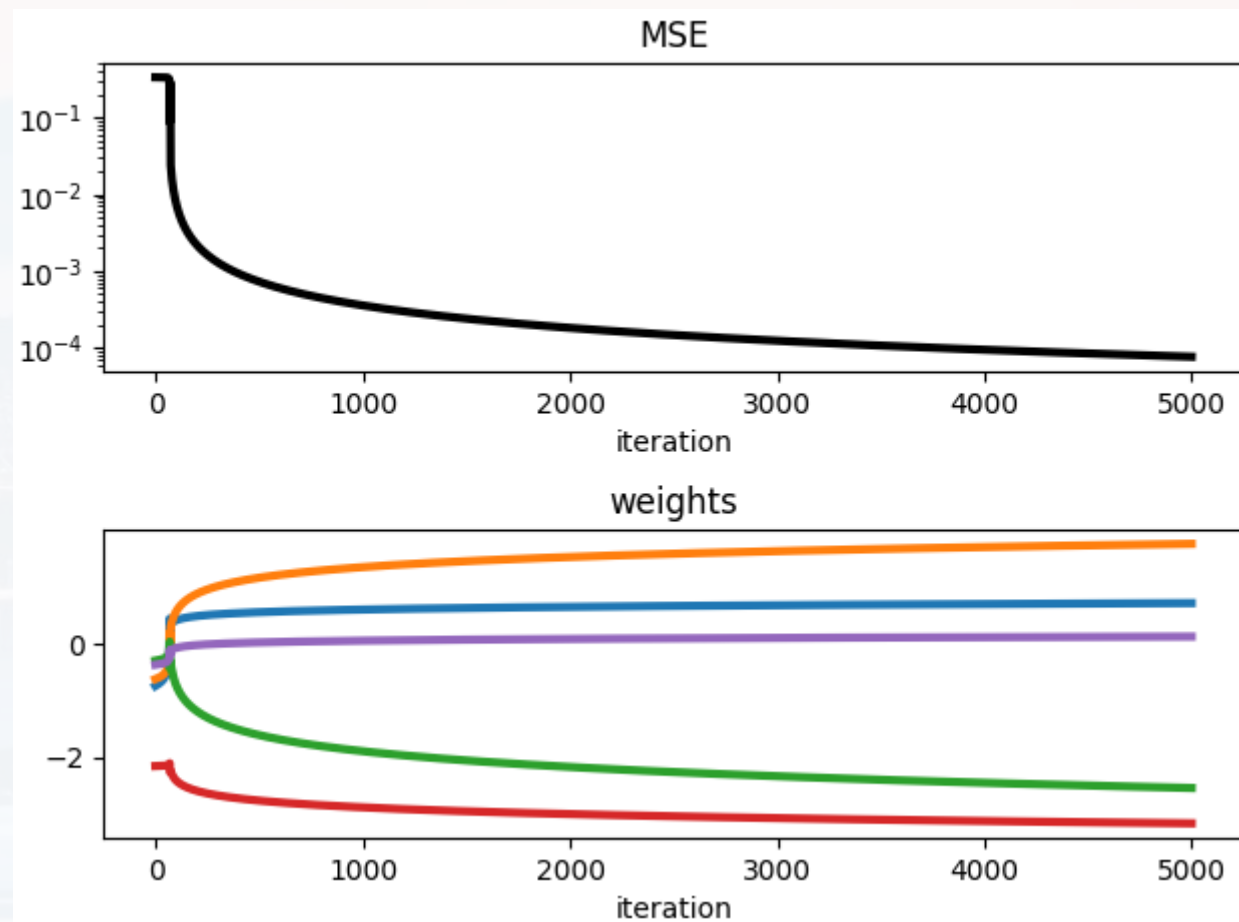
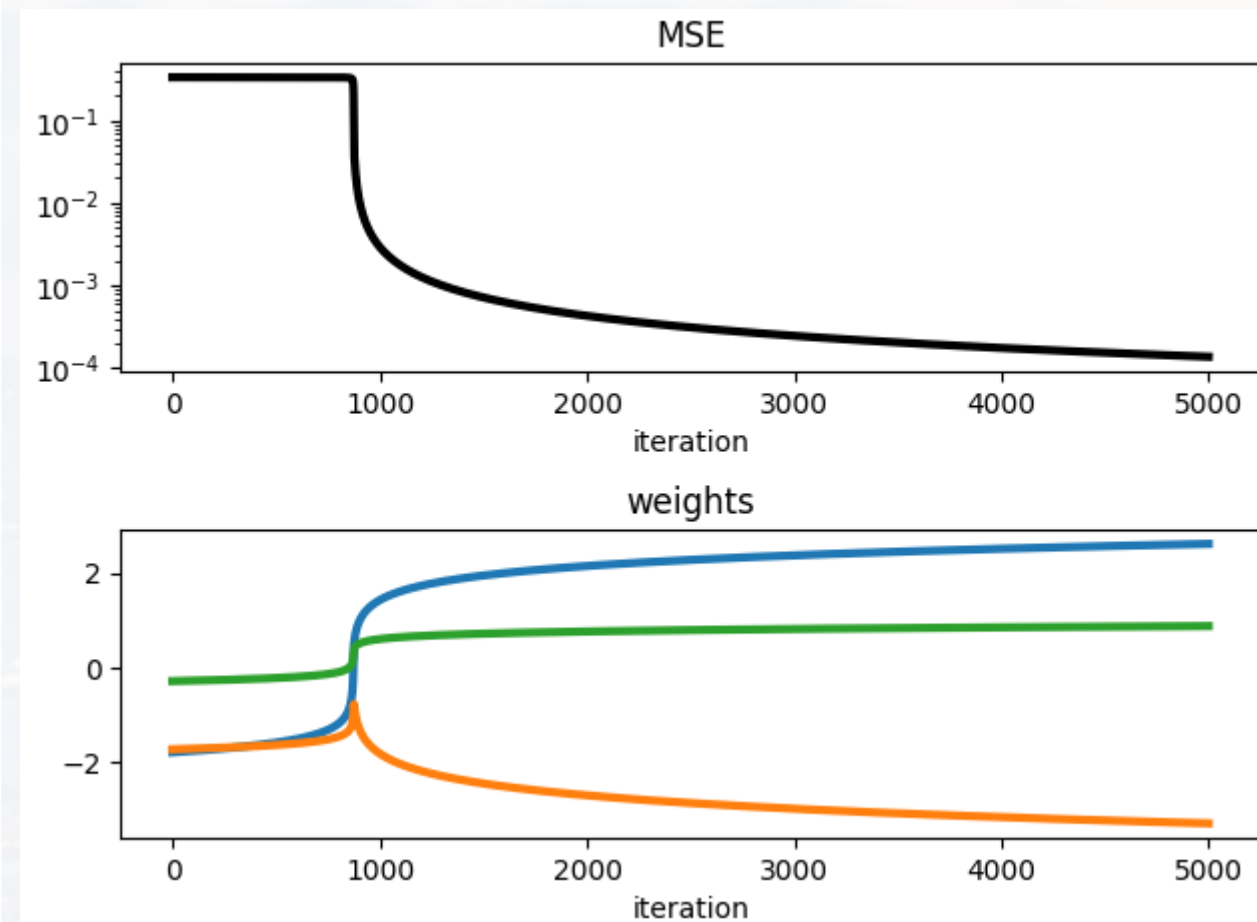
$$net = \sum_i I_i \cdot w_i + b$$

dot product



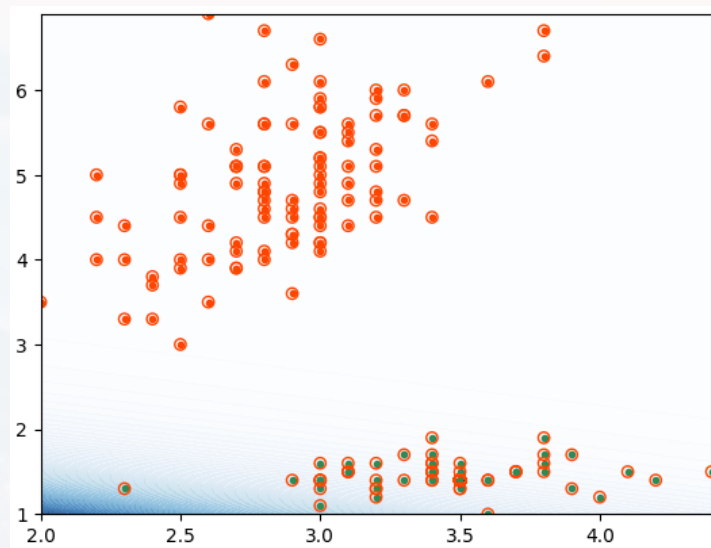


The training process

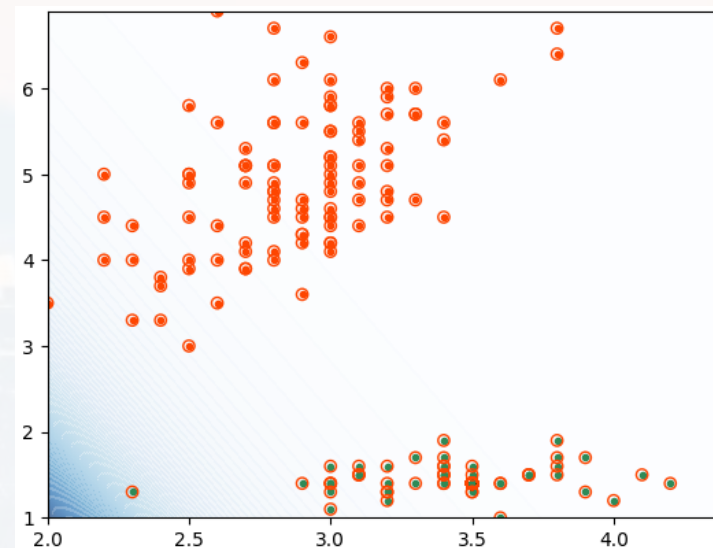




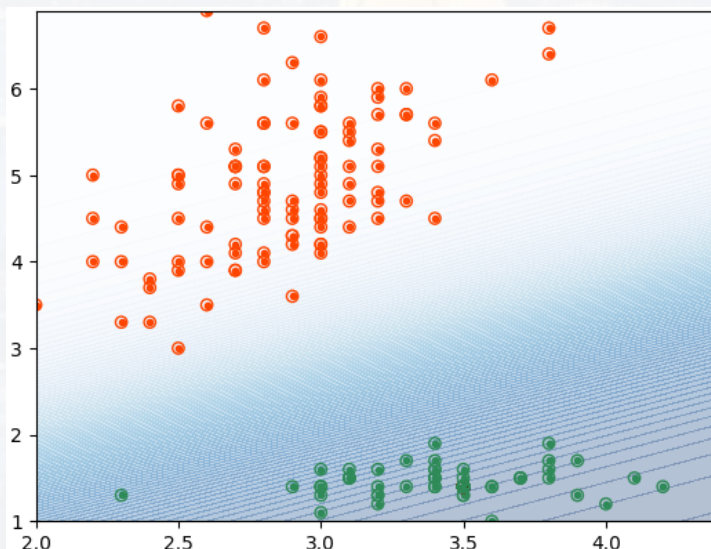
The training process



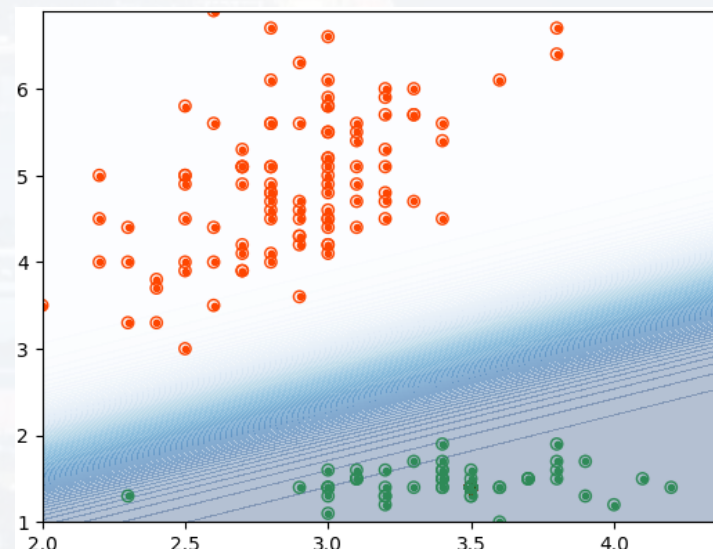
$t = 3$



$t = 5$



$t = 100$



$t = 10,000$

Thank you for your attention!

