Introduction to neural networks

Zalán Bodó

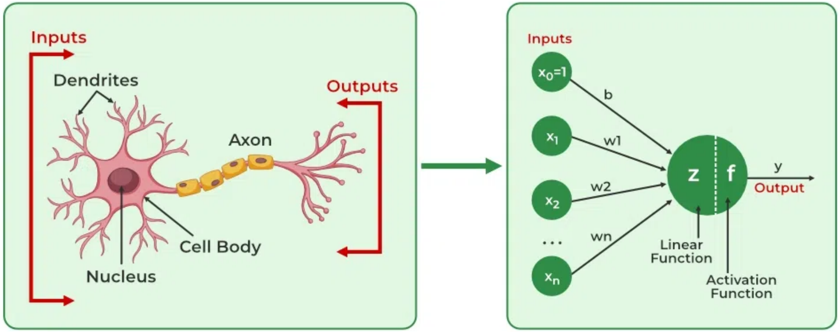
Faculty of Mathematics and Computer Science
Babeş–Bolyai University, Cluj-Napoca

Windesheim University of Applied Sciences, March 25–28, 2024

Presentation outline

Artificial models of the neuron

- 1943: McCulloch and Pitts describing a model of artificial neurons with binary threshold activation function capable of simulating logical gates
- 1949: Hebb's rule on synaptic plasticity
- 1958: Rosenblatt's perceptron algorithm
- 1969: *AI winter* caused by Minsky and Papert
- 1986: backpropagation algorithm
- 1995: CNNs (Yann Lecun: LeNet)
- 1997: LSTM
- 2017–2018: transformers (e.g. BERT)
- 2018: GPT
- ...

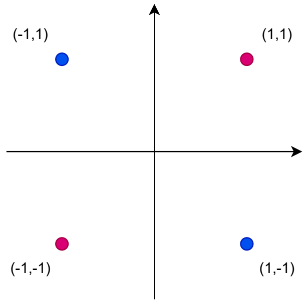


(From: https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/)

Deep neural networks: a large number of layers with a large number of neurons/connections

- no. of neurons in the human brain: $\approx$ 86 billion (10^9)
  no. of neural connections (synapses): $\approx$ ***60–100 trillion*** (10^{12})
- OpenAI:
  - GPT-2 (*Generative Pre-trained Transformer*, 2019): 1,5 billion parameters (= synapses)
  - GPT-3/3.5 (2020): 175 billion
  - GPT-4 (2023): ***1,76 trillion***

The XOR dilemma



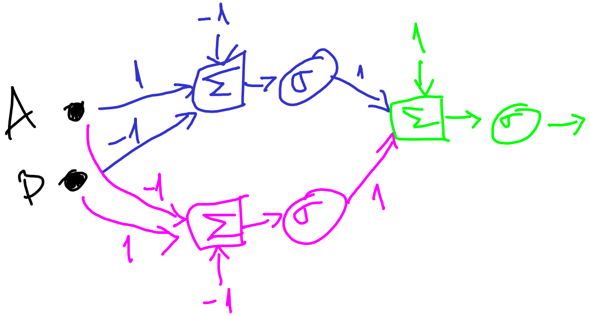
XOR view as a separation problem: separate the red and blue points from each other.

A	B	$A \oplus B$
false	false	false
false	true	true
true	false	true
true	true	false

$$A \oplus B = (A \wedge \overline{B}) \vee (\overline{A} \wedge B)$$

$$A \oplus B = (A \vee B) \wedge \overline{(A \wedge B)}$$

Solution: use a *multi-layer perceptron* (MLP)!



Some used concepts

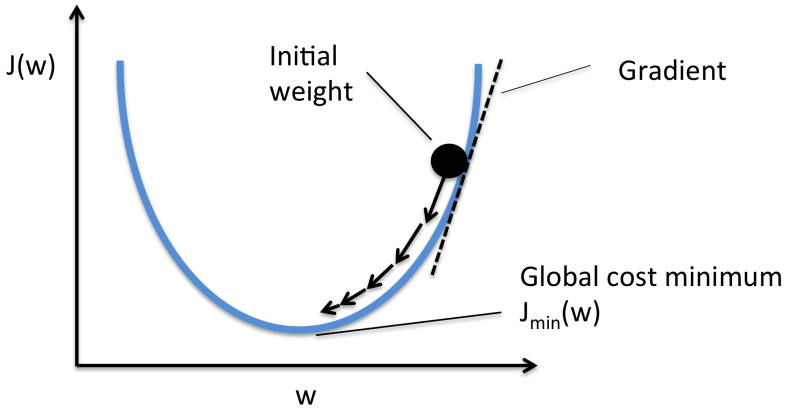
- neuron
- layer
- activation function
- batch – a batch contains a fixed number of data (e.g. 16)
- epoch – 1 epoch = all batches (i.e. the entire dataset) are processed exactly once
- iteration – in 1 iteration 1 batch is fed to the network;
  no. of iterations = the number of batches to complete 1 epoch

Gradient descent

- function to be optimized: $f : \mathbb{R}^d \to \mathbb{R}$
- gradient descent:
  - maximization: $x_{n+1} = x_n + \eta \nabla f(x_n)$
  - minimization: $x_{n+1} = x_n - \eta \nabla f(x_n)$ (opposite direction)
- the gradient always shows the direction of the maximum; why?
- answer: definition of derivative

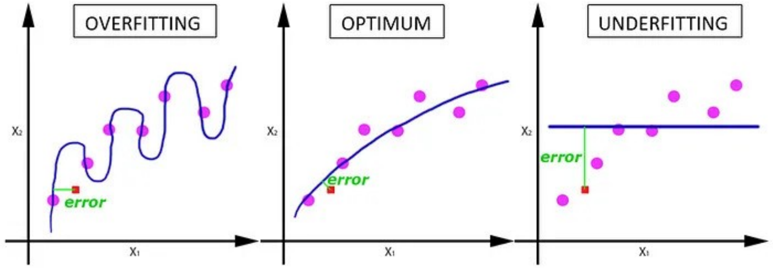
$$\nabla f(x) = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

  if $f(x)$ is increasing $\Rightarrow \nabla f(x)$ is positive; if decreasing $\Rightarrow$ negative
- e.g. $f(x) = x^2 - 1$, $\nabla f(2) = 4 \Rightarrow$ the $\nearrow$ direction is $+\infty$;
  similarly $\nabla f(-2) = -4 \Rightarrow$ the $\nearrow$ (or more precisely $\nwarrow$) direction is $-\infty$



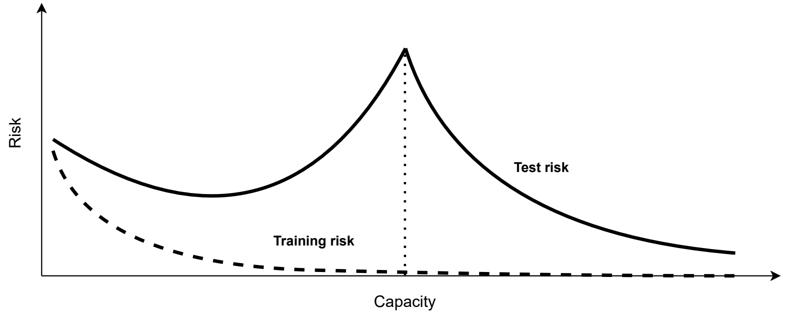
Gradient descent (from https://sebastianraschka.com/faq/docs/gradient-optimization.html)

Overfitting vs. underfitting



(From: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9)

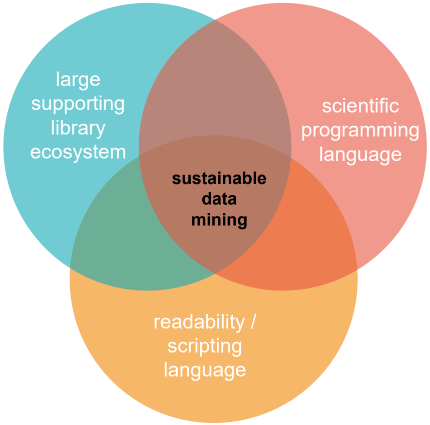
The double-U-shaped curve of DL



The double-U-shaped risk curve observed in deep learning, where the test risk goes lower in the case of overparametrized models. Capacity stands for the number of parameters of the model.

Neural networks in Python

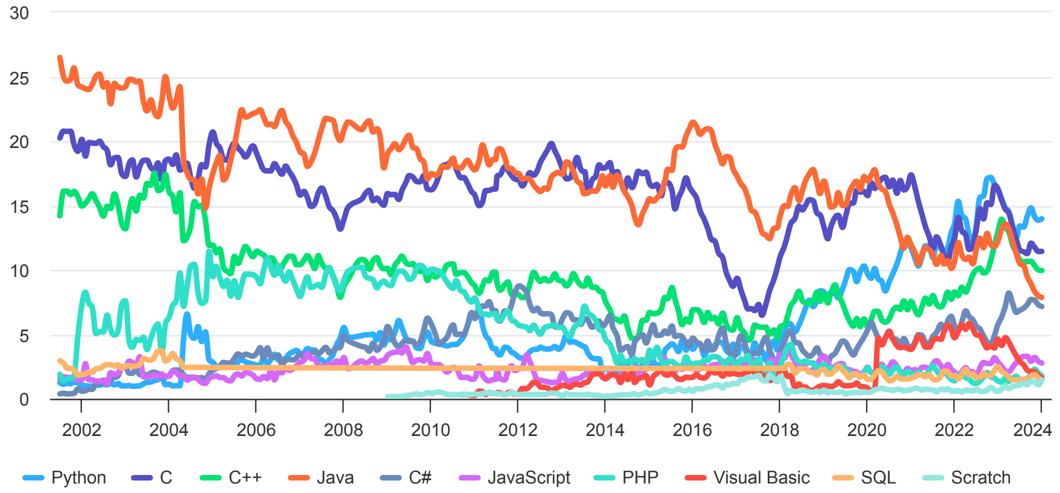
Why Python?


large supporting library ecosystem
scientific programming language
readability / scripting language
sustainable data mining

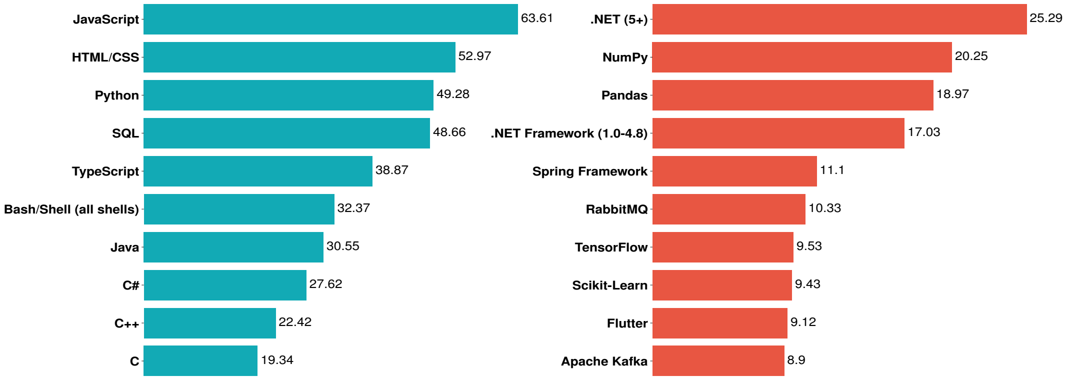
- NumPy
- SciPy
- SciPy toolkits a.k.a. *SciKits* $\rightarrow$ scikit-learn

- **BLAS** (Basic Linear Algebra Subprograms)[1]
  - 1979
  - Fortran and/or C
  - low-level linear algebra operations (vector, matrix additions/multiplications, dot products, etc.)
- **LAPACK** (Linear Algebra Package)[2] – based on BLAS
  - 1992
  - Fortran 90
  - higher-level linear algebra operations (solving systems of linear equations, eigenvalue, singular value problems, matrix factorizations, etc.)
- (almost) all scientific programming languages use BLAS and LAPACK or their descendants (e.g. MATLAB, Julia, R, ...)

[1] https://netlib.org/blas/
[2] https://netlib.org/lapack/



TIOBE Programming Community Index showing the most popular programming languages (until February 2024), calculated using 25 web search engines.



(a) (b)

Results of Stack Overflow's *2023 Developer Survey* (the values on the right of the bars are percentages): (a) top 10 "programming, scripting, and markup languages"; (b) top 10 "other frameworks and libraries". Charts reproduced after `https://survey.stackoverflow.co/2023`.

https://scikit-learn.org/

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

K Keras

https://keras.io/

- deep learning API written in Python and capable of running on top of either **JAX**, **TensorFlow**, or **PyTorch**
  - Simple — but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
  - Flexible — Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
  - Powerful — Keras provides industry-strength performance and scalability: it is used by organizations including **NASA**, **YouTube**, or **Waymo**.

- some important layers:
    - Dense – fully connected
    - Activation – activation layer, application of an activation function, e.g. sigmoid, softmax, ReLU
    - Dropout – regularization layer (to avoid overfitting)
    - (Input, Flatten, ...)
    - Conv1D, Conv2D – convolutional layer
    - MaxPooling1D, MaxPooling2D – pooling layers
    - Embedding
    - ...

Excercise 1.

Train a neural network using the *Heart Disease Dataset* to predict whether a patient has a heart disease or not.

- $+1/-1$ and 13 numerical values:
    1. age
    2. sex
    3. chest pain type (4 values)
    4. resting blood pressure
    5. ...

https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/heart_scale
https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

Excercise 2.

Train a neural network using the *Optical Recognition of Handwritten Digits* to recognize a handwritten digit.

- 8×8 pixels + class label (0, 1, ..., 9)

https://archive.ics.uci.edu/dataset/80/optical+recognition+of+handwritten+digits