

Neural Networks

Jordi Villà i Freixa

Universitat de Vic - Universitat Central de Catalunya
Study Abroad

jordi.villa@uvic.cat

course 2023-2024

Index

- 1 Introduction to NN
- 2 Convolutional Neural Networks
- 3 Bibliography

What to expect?

In this session we will discuss:

- Neural networks
- Gradient Descent
- Backpropagation
- Network training
- Convolutional Neural Networks

Evolution of AI

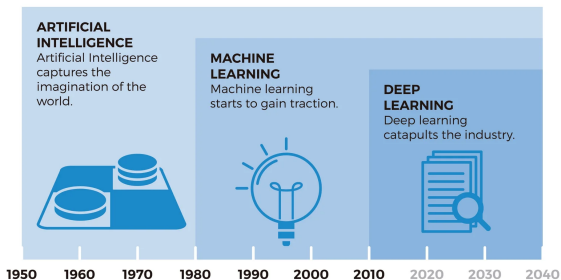


Figure 1: Evolution of AI, ML and DL. Source [BISMART](#). Check also this [3blue1brown](#) set of videos.

Supervised ML

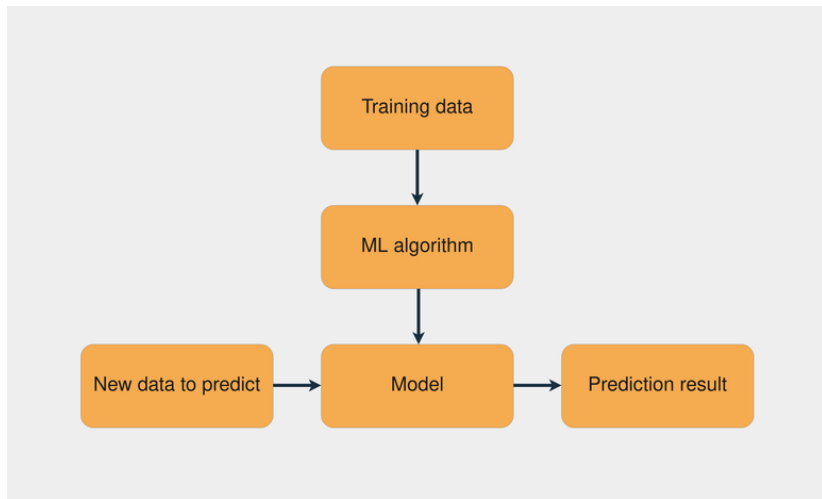


Figure 2: Workflow to train a model using supervised learning.

Supervised ML[1]

- Build models capable of learning from a series of data X .
- The learning process depends on the task we want to train the model:

Supervised the input data X is accompanied by the values y that we want the model to learn or also called targets (linear regression, decision trees, ...). The objective is that the prediction of the model given some data x is equal to the target.

Unsupervised the task to be learned is some type of association of the input data X with each other (i.e clustering).

Feature engineering

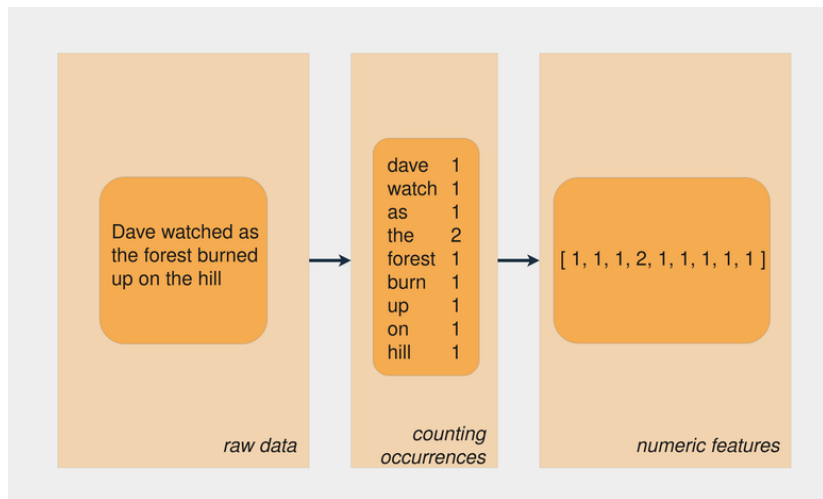


Figure 3: Feature engineering example.

Neural networks

- In the ML example above, we use lemmatization to eliminate inflection from words, generating a simpler model.
- In neural networks, **we leave the network itself** to find out the most important features in the data, without relying on feature engineering techniques.

Neural Networks **are based on** a collection of connected units (neurons), which, just like the synapses in a brain, can transmit a signal to other neurons, so that, acting like interconnected brain cells, they can learn and make decisions in a more human-like manner.[2]

ML vs DL

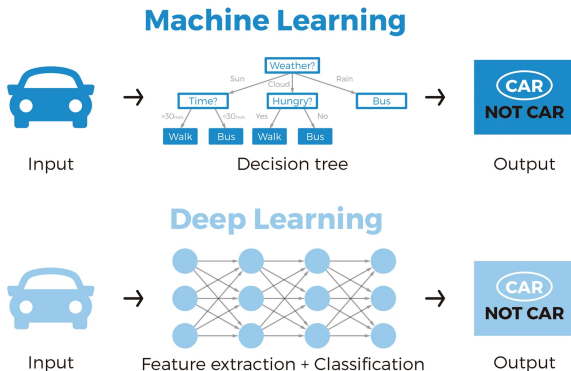


Figure 4: ML (fixed/engineered features + trainable classifier) vs DL (trainable feature extractor + trainable classifier). Source [BISMART](#).

Neural networks

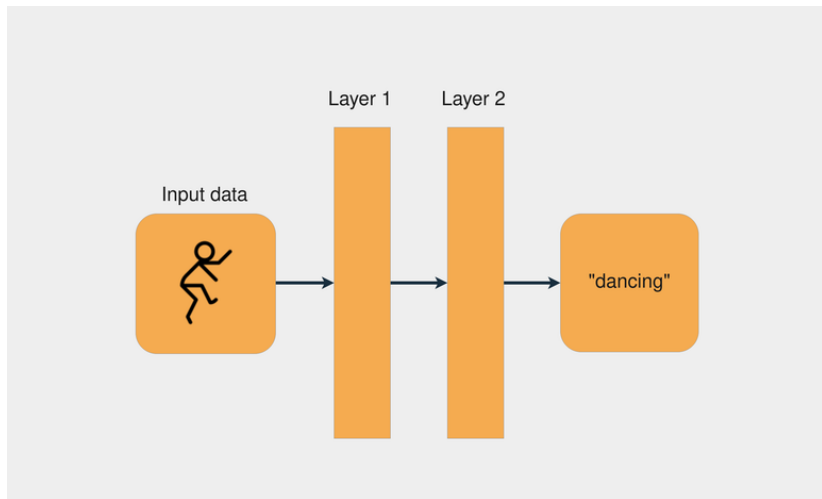


Figure 5: Schema of a 2 layer neural network.

Throwing darts and improving by learning

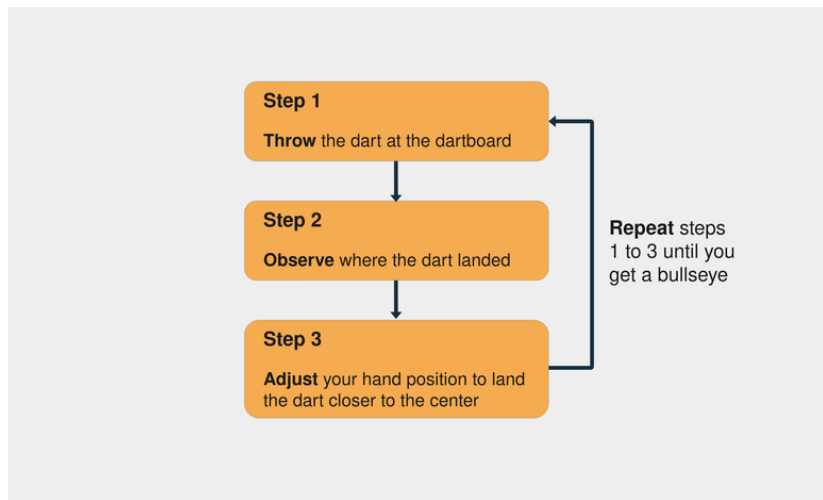


Figure 6: Steps for trying to hit the center of a dartboard.

Artificial Neural Networks (ANN)

- ANN are formed by layers of neurons.
- Each layer has a certain number of neurons.
- The input data X enters through the first layer and through mathematical operations the input values are transformed into output values y' .
- The goal of the network is to modify the mathematical operations through some parameters (weights and biases, W and b) to minimize the difference between y and y' .
- Deep Learning refers to all those models that use ANN of any type with multiple layers.

Perceptron

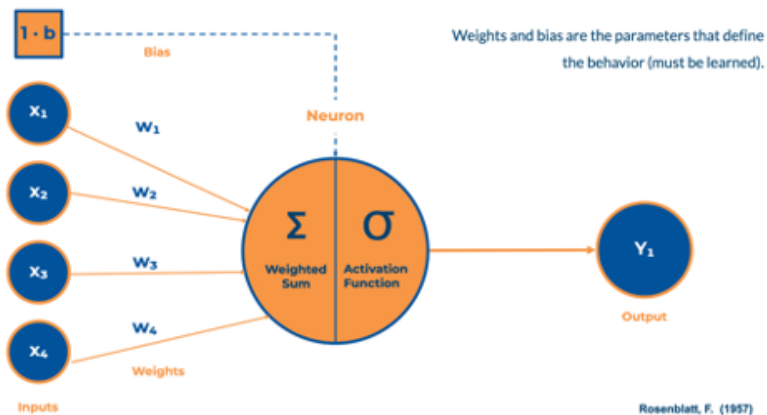


Figure 7: $w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 \leq \text{threshold} \Rightarrow Y = 0$. We can introduce bias and a given function: $\sigma(\mathbf{w} \cdot \mathbf{x} + b) = y'$. Adapted from [3].

Multilayer perceptron

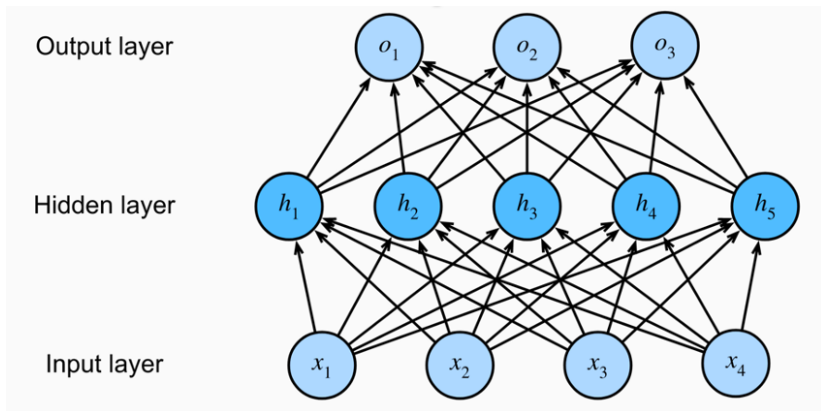


Figure 8: A multilayer perceptron with 5 hidden units. See also [this video](#).

Multilayer perceptron: some maths

$$w_{ij}^L \begin{cases} L = \text{layer number} \\ i = \text{neuron from previous layer} \\ j = \text{neuron from following layer} \end{cases}$$

$$\sigma_1 \left(\mathbf{W}^{(1)T} \cdot \mathbf{X} + \mathbf{b}^{(1)} \right) = \sigma_1 \left(\begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} & w_{14}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} & w_{24}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} & w_{34}^{(1)} \\ w_{41}^{(1)} & w_{42}^{(1)} & w_{43}^{(1)} & w_{44}^{(1)} \\ w_{51}^{(1)} & w_{52}^{(1)} & w_{53}^{(1)} & w_{54}^{(1)} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \\ b_5^{(1)} \end{bmatrix} \right) = \mathbf{H}^{(1)}$$

$$\sigma_2 \left(\mathbf{W}^{(2)T} \cdot \mathbf{H}^{(1)} + \mathbf{b}^{(2)} \right) = \sigma_2 \left(\begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} & w_{13}^{(2)} & w_{14}^{(2)} & w_{15}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} & w_{23}^{(2)} & w_{24}^{(2)} & w_{25}^{(2)} \\ w_{31}^{(2)} & w_{32}^{(2)} & w_{33}^{(2)} & w_{34}^{(2)} & w_{35}^{(2)} \end{bmatrix} \cdot \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{bmatrix} + \begin{bmatrix} b_1^{(2)} \\ b_2^{(2)} \\ b_3^{(2)} \end{bmatrix} \right) = \mathbf{Y}^{\text{out}}$$

Multilayer perceptron: some details

- The MLP is the simplest neural network
- Weights connect neurons from an inner to an outer layer
- Operations in each layer are implemented using matrices and vectors
- The **forward pass** of the last layer produces the output.
- The MLP can be interpreted as a multivariate function with W and b as the parameters, X as input and Y^{out} as output.
- **Hiperparameters** are additional parameters that control other aspects: number of layers, number of neurons per layer, activation functions, ...

The logistic function

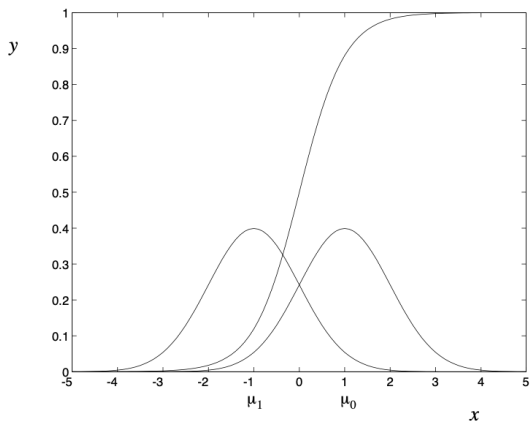






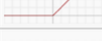

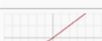


Figure 9: A binary classification problem. The class-conditional densities are Gaussians with unit σ^2 . The posterior probability is the logistic function $y = 1/(1 + \exp\{-2x\})[4]$.

Activation functions

Activation functions:

- The **sigmoid** function is used in the final layer in classification models.
- **ReLU** is used in inner layers, as they produce non-linearity.
- All are differentiable.

Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$

An example of a 2 layer Neural Network

Example of classification (for categories "1" and "0") with a 2 layers NN.

- In the first layer we use a linear regression approach.
- The second layer includes the non-linearity through the use of a **sigmoid function** that decides whether the prediction is 1 or 0, following the **Bernoulli distribution**.

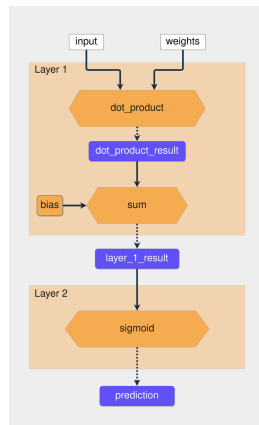


Figure 10: Training a two layer neural network.

Exercise 1

Loss/Cost function in a simple 2 layer NN Check the [provided code](#) for this session and expand it to:

- 1 Generate a collection of 100 synthetic data points from a two Gaussian distribution.
- 2 Assign the data points a category "0" or "1" depending on the Gaussian function they are obtained from.
- 3 Use the provided 2 layers NN to evaluate the expected value of the loss function with the provided weights.
- 4 Can you guess the weights that would improve the prediction (reducing the value of the error)?

Backpropagation: weights

Chain Rule in Backpropagation In each **backward pass**, you compute the partial derivatives of each function, substitute the variables by their values, and finally multiply everything.

Notice that, for the sigmoid function

$$f(x) = \frac{1}{1+e^{-x}}, f'(x) = 1 - f(x).$$

```

error_dweights =
error_dprediction
* dprediction_dlayer1
* dlayer1_dweights
  
```

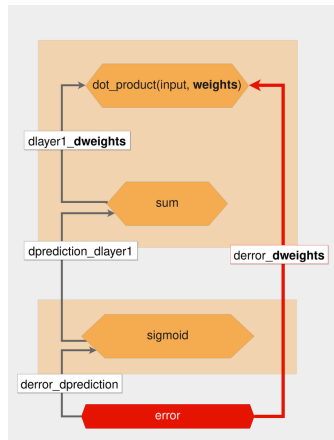


Figure 11: Chain rule: derivative of the error with respect to **weights**.

Backpropagation: bias

Chain rule applied to the derivative of the bias error in the 2 layer example.

```
derror_dbias =
derror_dprediction
* dprediction_dlayer1
* dlayer1_dbias
```

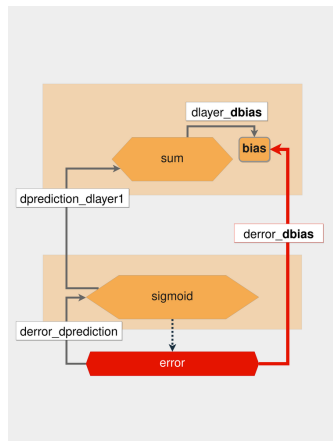


Figure 12: Chain rule: derivative of the error with respect to bias.

Convolutional Neural Networks

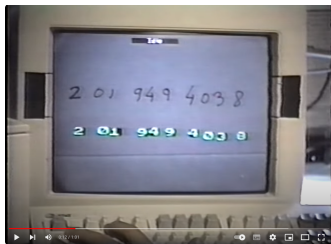


Figure 13: Neural network for ZIP code recognition. To learn more: [5] and *distill*.

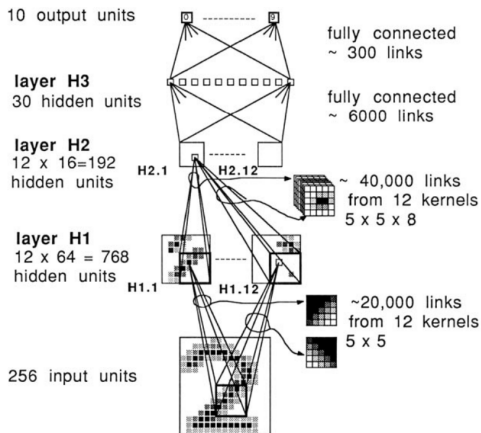


Figure 14: Structure of the LeCun et al. neural network for ZIP code recognition[6].

CNN

- Unlike traditional NN, which are fully connected, a CNN uses convolutional layers to learn features
- A CNN uses filters to extract features such as edges, corners or textures.
- It includes pooling layers, which downsample the output by taking the maximum or average value in a local neighborhood.
- Once the CNN layers have been passed, the data is flattened into a one-dimensional array that is input into one or more fully connected layers to predict from the learned features.

Convolution and kernels

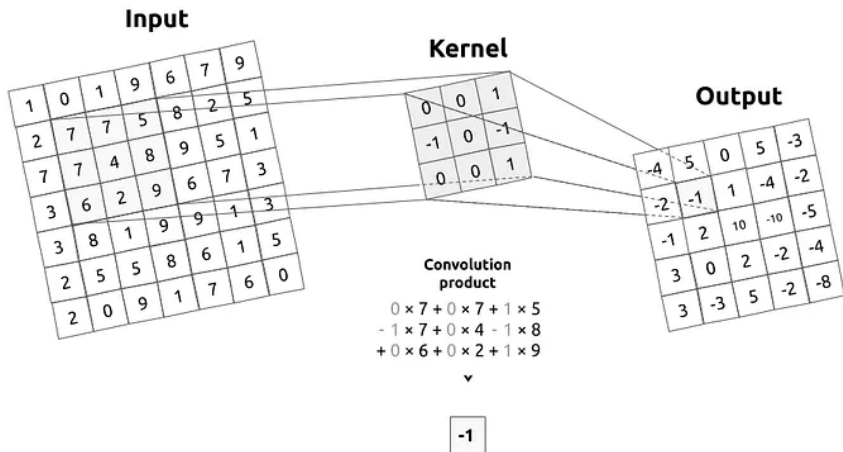


Figure 15: Convolution between an input image and a kernel. Source

Kernels

The trainable parameters in a CNN are all the parameters that will be updated when the network is trained. In a Conv2D, the trainable elements are the values that compose the kernels. In a 3×3 convolution kernel, we have $3 \cdot 3 = 9$ trainable parameters plus, eventually, a bias (for a complete explanation, check [this link](#)):

$$\omega \cdot F(x, y) = \omega_{bias} + \sum_{\delta x=-k_i}^{k_i} \sum_{\delta y=-k_j}^{k_j} \omega(\delta x, \delta y) \cdot F(x + \delta x, y + \delta y)$$

The number of parameters increases linearly with the number of convolution kernels. Hence linearly with the number of desired output channels.

The MNIST Handwritten Digit Classification Dataset

- The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset.
- It contains 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.
- The task is to classify a given image of a handwritten digit into one of 10 classes representing integer values from 0 to 9, inclusively.
- Top-performing models are deep learning convolutional neural networks that achieve a classification accuracy of above 99%, with an error rate between 0.4% and 0.2% on the hold out test dataset.

Keras vs Tensorflow vs pytorch

- Tensorflow** Low-level software library created by Google to implement ML and to solve complex numerical problems. Every element is converted into a graphical form. The variables in the graph are tensors and the operations are called *operators*. Tensors are unmutable. High productivity.
- Keras** High-level deep learning API written in Python for easy implementation and computation of neural newtworks. Keras uses Tensorflow as its low-level engine.
- Pytorch** Low-level API developed by Facebook for natural language processing and computer vision. More powerful than numpy. Most popular among researchers.



Dirk P. Kroese, Zdravko Botev, Thomas Taimre, and Radislav Vaisman.

Data Science and Machine Learning: Mathematical and Statistical Methods.

Machine Learning & Pattern Recognition. Chapman & Hall/CRC, 2020.



Michael A. Nielsen.

Neural Networks and Deep Learning.
2015.

Publisher: Determination Press.



F. Rosenblatt.

The perceptron: A probabilistic model for information storage and organization in the brain.

Psychological Review, 65(6):386–408, 1958.

Place: US Publisher: American Psychological Association.



Michael I Jordan.

Why the logistic function? A tutorial discussion on probabilities and neural networks.

Computational Cognitive Science Technical Report 9503, 1995.



Neural Networks, Manifolds, and Topology – colah's blog.



Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel.

Backpropagation Applied to Handwritten Zip Code Recognition.

Neural Computation, 1(4):541–551, December 1989.