Optimization for deep learning

Toulouse School of Economics

Ludovic De Matteis

Course Overview

Planning

	Day	Time	Subject
1	13/10/2025	9:30 - 12:30	Basic definitions, Gradient Descent and Newton's method
2	27/10/2025	9:30 - 12:30	Practical - Gradient Descent and Newton's method
3	03/11/2025	9:30 - 12:30	Neural networks and stockastic gradient descent
4	10/11/2025	9:30 - 12:30	Practical - Neural Networks and digit recognition
5	17/11/2025	9:30 - 12:30	Alternative Neural Structures
6	24/11/2025	9:30 - 12:30	Practical - Alternative Neural structure, Adversial networks

Lecture 2 -

Optimizing neural networks

Summary

1. From perceptrons to multi-layers perceptron
   - Definition of a perceptron
   - Limits of single layer perceptrons
   - Extension to multi-layer perceptrons
2. Architecture and forward propagation
3. Backpropagation and derivatives
   - Backpropagation algorithm
   - Derivatives of common loss functions
4. Optimization algorithms
   - Stochastic gradient descent
   - Mini batch gradient descent
   - Momentum and Adam optimizer

From perceptrons to multi-layer perceptrons

From perceptrons to multi-layer perceptrons

Motivations

- Neural networks are inspired by the structure of the human brain where neurons are interconnected to process information and learning occurs by adjusting the strength of these connections
- The perceptron is a fundamental building block of neural networks, representing a simplified model of a biological neuron
- In machine learning, we look for a parametrized model capable of approximating a function $f : \mathbb{R}^d \mapsto \mathbb{R}^n$ from examples

$$f_\theta \approx (x_i) y_i$$

From perceptrons to multi-layer perceptrons

History

- **1943**: McCulloch and Pitts propose a mathematical model of a neuron based on logical gates
- **1958**: Frank Rosenblatt invents the perceptron, an early neural network model capable of binary classification
- **1969**: Minsky and Papert publish “Perceptrons”, highlighting limitations of single-layer perceptrons, leading to a decline in neural network research
- **1986**: Rumelhart, Hinton, and Williams popularize the backpropagation algorithm, enabling training of multi-layer neural networks

From perceptrons to multi-layer perceptrons

Definition of a perceptron

- A perceptron is a simple computational unit that takes multiple input signals, applies weights to them, sums them up, and passes the result through an activation function to produce an output
- Mathematically, a perceptron can be represented as:

$$\text{output} = \Phi\Big(\sum(\text{weights} * \text{inputs}) + \text{bias}\Big)$$

or

$$y = \Phi(w^T x + b)$$

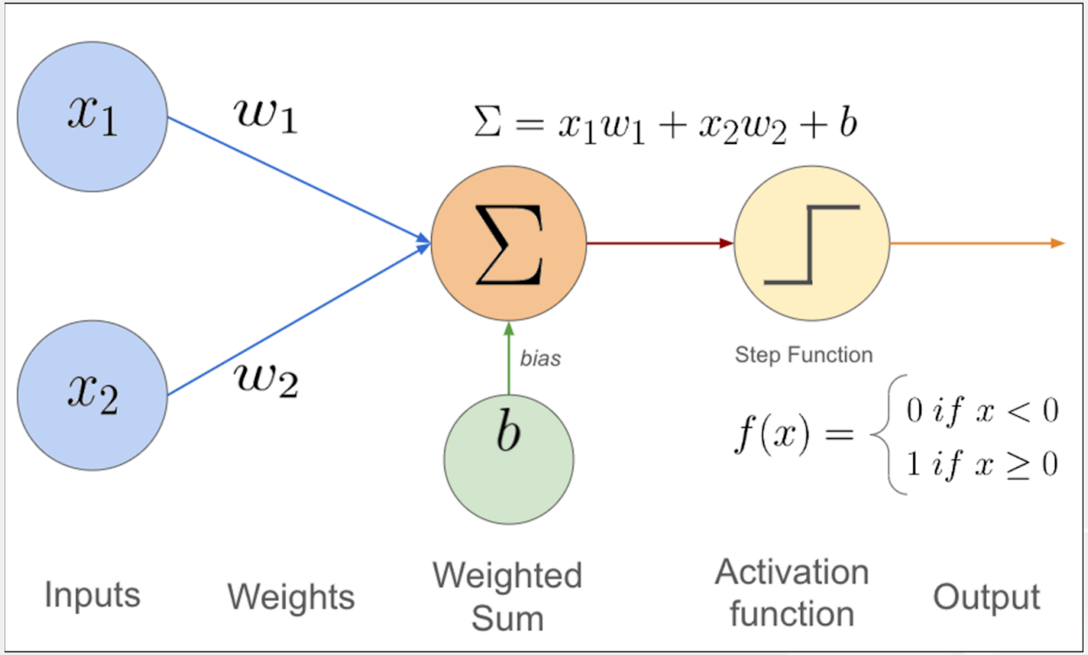
where:

- x are the **input** features
- w are the **weights**, the parameters learned during training
- b is the bias, an additional parameter to shift the activation function
- Φ is the **activation function** introducing non-linearity (e.g., step function, sigmoid, ReLU)

From perceptrons to multi-layer perceptrons

Definition of a perceptron

- The classical representation of a perceptron is as follows:



From perceptrons to multi-layer perceptrons

Linear classifier

- If we use a step activation function, the perceptron can be used as a linear classifier

$$\Phi(z) = 1 \text{ if } z \geq 0 \text{ else } 0$$

- The decision boundary is defined by the equation:

$$w^T x + b = 0$$

- In the first practical session, we implemented the perceptron using a sigmoid activation function (closer to the step function)

$$\Phi(z) = \frac{1}{1 + e^{-z}}$$

From perceptrons to multi-layer perceptrons

Optimizing a perceptron

- We look for the optimal values of w and b
- We can use gradient descent to minimize a loss function, such as the mean squared error or cross-entropy loss (if everything if differentiable)
- Or we can use the following algorithm:
  1. Compute the output of the perceptron for each training example
  $$\hat{y}_i = \Phi(w^T x_i + b)$$
  2. Update the weights and bias based on the prediction error
  $$w := w + \eta(y_i - \hat{y}_i)x_i$$
  $$b := b + \eta(y_i - \hat{y}_i)$$
  where η is the learning rate, controlling the step size of the updates

From perceptrons to multi-layer perceptrons

Limits of single-layer perceptrons

- Single-layer perceptrons can only learn linearly separable functions
- They cannot solve problems that require non-linear decision boundaries, such as the XOR problem
- To overcome these limitations, we extend the perceptron to multi-layer architectures, allowing for more complex representations and decision boundaries

Multi-layer perceptron and forward propagation

Multi-layer perceptron and forward propagation

Motivations

- To obtain non linear functions, we combine multiple linear functions with non-linear activation functions
- The output will therefore be defined for a composition of L functions as:

$$y = \Phi_L(W_L.\Phi_{L-1}(W_{L-1}....\Phi_1(W_1x + b_1)... + b_{L-1}) + b_L)$$

Multi-layer perceptron and forward propagation

Architecture of a multi-layer perceptron

- A multi-layer perceptron (MLP) consists of multiple layers of interconnected perceptrons (neurons)
- It typically includes an input layer, one or more hidden layers, and an output layer
- Each layer applies a linear transformation followed by a non-linear activation function (that can be different for each layer)
- The architecture allows the network to learn complex patterns and representations from the data
- The function computed by each layer l can be expressed as:

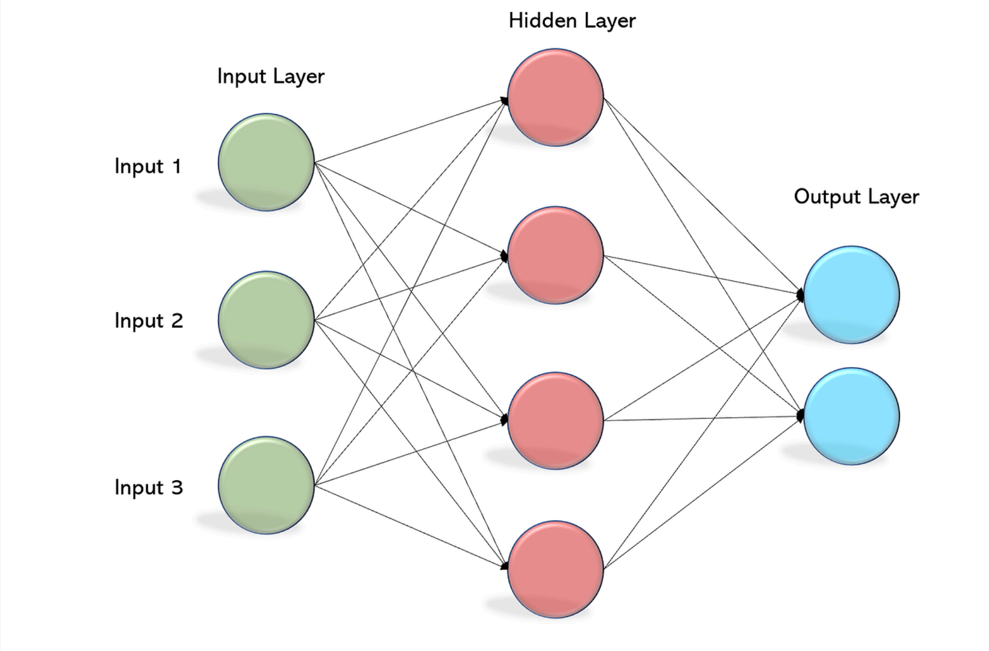
$$a^l = \Phi_l\left(W_l a^{l-1} + b_l\right)$$

  with $a^0 = x$

Multi-layer perceptron and forward propagation

Architecture of a multi-layer perceptron

- Example of a simple MLP with one hidden layer (we often omit the bias nodes in diagrams for clarity):



Multi-layer perceptron and forward propagation

Forward propagation

- Forward propagation is the process of computing the output of the MLP given an input
- It involves passing the input through each layer, applying the linear transformation and activation function at each step
- The steps for forward propagation are as follows:
  1. Initialize the input layer with the input features x
  2. For each layer l from 1 to L:
     - Compute the linear transformation:

$$z^l = W_l a^{l-1} + b_l$$

     - Apply the activation function:

$$a^l = \Phi_l(z^l)$$

  3. The final output is obtained from the output layer:

$$\hat{y} = a^L$$

Multi-layer perceptron and forward propagation

Forward propagation algorithm

```
test
```

Multi-layer perceptron and forward propagation

XOR problem

- The XOR problem is a classic example that illustrates the limitations of single-layer perceptrons and the capabilities of multi-layer perceptrons
- It can be solved using a MLP with a single hidden layer of two neurons
- It is possible to write by hand the formulation of the output