Optimization for deep learning

Toulouse School of Economics

Ludovic De Matteis

Course Overview

Teacher

Ludovic De Matteïs
ldematteis@laas.fr - 0781390605

- PhD Student in robotics at LAAS-CNRS
  - Research interests in optimization and optimal control
- Studies at Ecole Normale Superieure (ENS) Paris-Saclay
  - Master in Electrical Engineering
  - Master in Mathematics, Vision and Learning
  - Agrégation in Engineering - specialty in Computer Science

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

Course Overview

Evaluation

- No final exam
- 3 pratical session
  - 1 report per session
  - 2 weeks delay to complete each report
  - 60% of the final grade
- 1 MCQ at the end of the course
  - 40% of the final grade

Lecture 1 -

On gradients and optimization algorithms

Summary

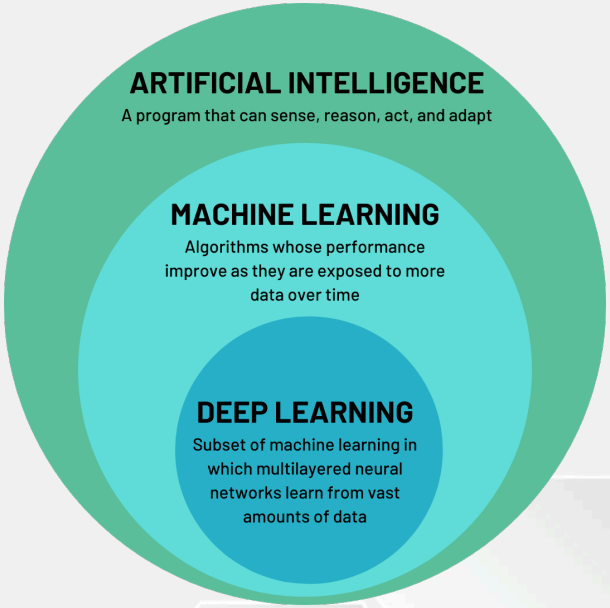
Motivation in Machine Learning

Motivation in Machine Learning

What is machine learning?

- Subfield of artificial intelligence that focuses on the development of algorithms that enable computers to perform specific tasks without explicit instructions.
- Systems learn from and make predictions or decisions based on data.

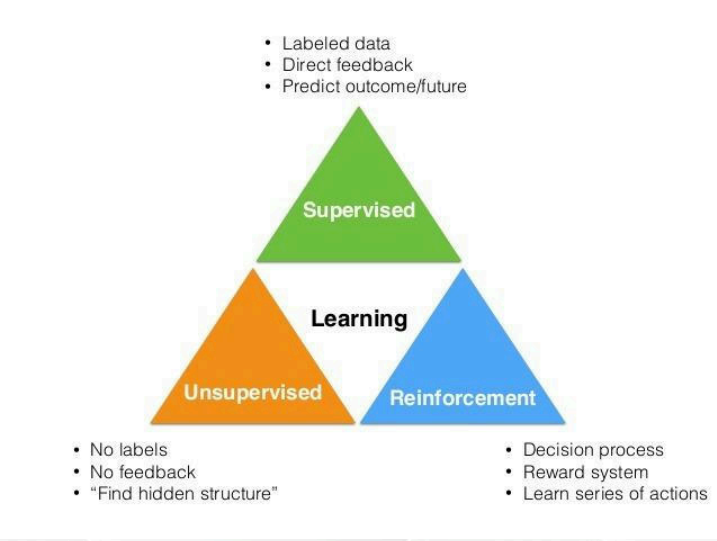
- The primary goal of machine learning is to enable computers to improve their performance on a task over time as they are exposed to more data.
- Widely used in various applications, including image and speech recognition, natural language processing, recommendation systems, robotics, finances...



Motivation in Machine Learning

What is machine learning?

- Three main classes of machine learning:
  - **Supervised learning**: The model is trained on a labeled dataset, where the input data is paired with the correct output. The goal is to learn a mapping from inputs to outputs.
  - **Unsupervised learning**: The model is trained on an unlabeled dataset, where the input data does not have corresponding output labels. The goal is to discover patterns or structures in the data.
  - **Reinforcement learning**: The model learns to make decisions by interacting with an environment. It receives feedback in the form of rewards based on its actions and aims to maximize cumulative rewards over time.



Motivation in Machine Learning

What is optimization?

- Optimization is the process of finding the best solution to a problem from a set of possible solutions, often by minimizing or maximizing a specific objective function.
- It can be written (in the case a minimization as)

$$\inf_{x \in \mathcal{X}} f(x)$$

where $f : \mathcal{X} \to \mathbb{R}$ is the **objective function**, x are the **decision variables** and $\mathcal{X}$ is the **set of feasible points**.

- In machine learning, optimization is used to find the best parameters for a model to minimize a loss function that measures the difference between the model's predictions and the actual data.
- The best decision variable is called the **optimal solution** and is denoted x^*.

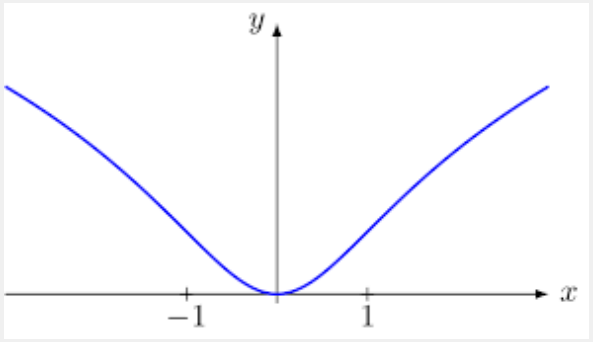
Motivation in Machine Learning

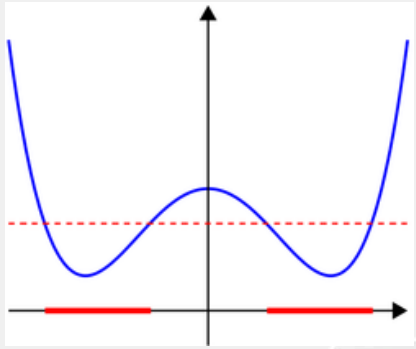
What is optimization?

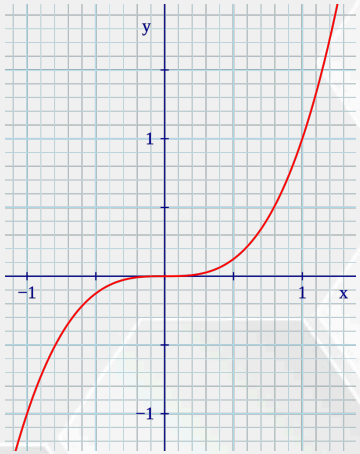
- We define the set of **global minimizers** of the function f as

$$\arg\min_{x\in\mathbb{R}^n} f(x) \stackrel{\text{def}}{=} \{x_0 \in \mathbb{R}^n \mid \forall x \in \mathbb{R}^n, f(x_0) \leq f(x)\}$$

- The global minimizer of a function does not necessarily exists and if it does i can be non unique.



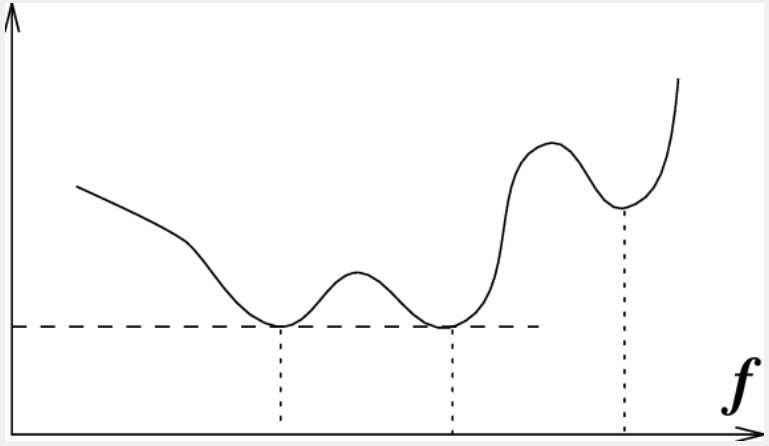


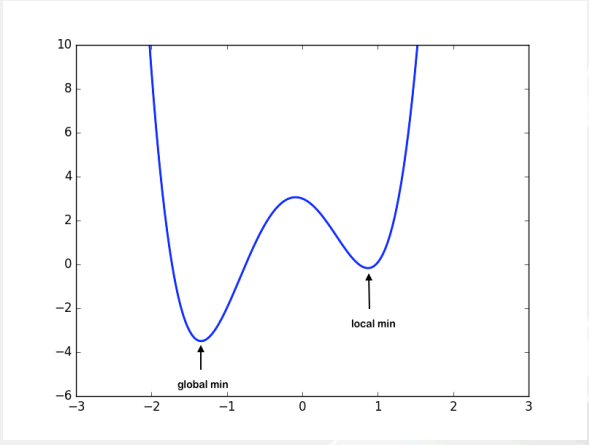


Motivation in Machine Learning

What is optimization?

- We will also define the notion of **local minimizer** as follows. The point x^* is a local minimizer of the function f if there exists a radius $r > 0$ such that for all $x \in \mathbb{R}^n$ such that $\|x - x^*\| \leq r$, we have $f(x^*) \leq f(x)$.
- Note that a local minimizer is not necessarily a global minimizer (but a global minimizer is necessarily a local minimizer).





Motivation in Machine Learning

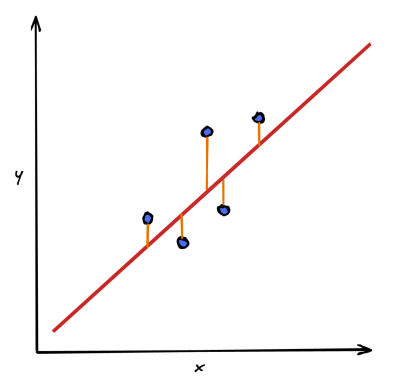
Classical problems in machine learning

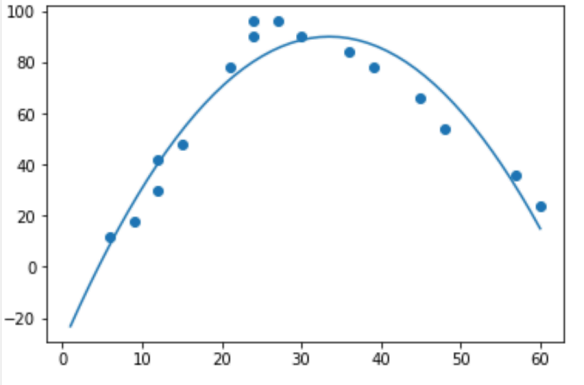
Regression

- The problem of regression consists in finding a function that best fits a set of data points.
- We will start by considering a linear regression problem, in which

$$f(x) = \frac{1}{2}\sum_{i-1}^{N}(y_i - < x, a_i >)^2 = \frac{1}{2} \parallel Ax - y \parallel^2$$

is the least square quadratic risk function.





Motivation in Machine Learning

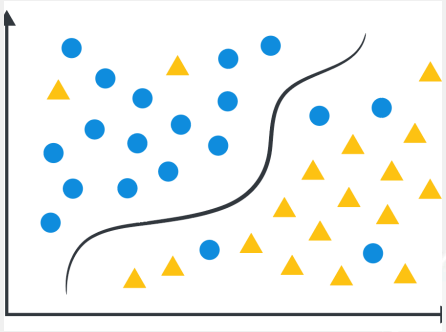
Classical problems in machine learning

Classification

- For binary classification, the data points y_i are **labelled** with a value 1 or -1, defining a class.
- In this problem, we aim the minimize the function

$$f(x) = \sum_{i-1}^{n} l(-y_i < x, a_i >) = L(-\text{diag}(y)Ax)$$

where l is the 0-1 loss function $1_{\mathbb{R}^+}$ or a smooth approximation of it, giving a value of 1 if the signs of y_i and $< a_i, x >$ are opposed and zero otherwise.



Derivatives and gradients

Derivatives and gradients

Reminder on derivatives

- A function $f : \mathbb{R} \to \mathbb{R}$ is said to be differentiable at a point $x_0 \in \mathbb{R}$ if the following limit exists

$$\lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

The value of this limit when it exists is called the derivative of f at x_0 and is denoted $f'(x_0)$ or $\frac{df}{dx}(x_0)$.

Derivatives and gradients

Reminder on derivatives

Examples

- Quadratic function: $f(x) = 3x^2 - x$

- Exponential function: $f(x) = e^{-2x}$

Note: we know that the limit when $h \to 0$ of $\frac{(e^h - 1)}{h}$ is 1 (it can be shown using the Taylor expansion of the exponential function).

Derivatives and gradients

On gradients and Jacobians

Gradient

- Extend derivative definition to a function of multiple variables $f : \mathbb{R}^d \to \mathbb{R}$
- The function f is differentiable at a point $x_0 \in \mathbb{R}^d$ if there exists a vector $g \in \mathbb{R}^d$ such that

$$f(x_0 + h) = f(x_0) + g^T h + o(\|h\|)$$

where $o(\|h\|)$ is a function such that $\frac{o(\|h\|)}{\|h\|} \to 0$ when $\|h\| \to 0$.

- The vector g is called the gradient of f at point x_0 and is denoted $\nabla f(x_0)$ or $\frac{df}{dx}(x_0)$.
- The gradient vector is related to the derivatives of the function by

$$\nabla f(x) = \left(\frac{df}{dx_1}(x), \frac{df}{dx_2}(x), ..., \frac{df}{dx_d}(x) \right)^T$$

Derivatives and gradients

On gradients and Jacobians

Jacobian

- We define the **Jacobian** of a function $f : \mathbb{R}^d \to \mathbb{R}^p$ as the matrix $J \in \mathbb{R}^{p \times d}$ writen as

$$J = \begin{pmatrix} \frac{df_1}{dx_1} & \frac{df_1}{dx_2} & \cdots & \frac{df_1}{dx_d} \\ \frac{df_2}{dx_1} & \frac{df_2}{dx_2} & \cdots & \frac{df_2}{dx_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{df_p}{dx_1} & \frac{df_p}{dx_2} & \cdots & \frac{df_p}{dx_d} \end{pmatrix}$$

$$= \begin{pmatrix} \nabla f_1(x)^T \\ \nabla f_2(x)^T \\ \vdots \\ \nabla f_p(x)^T \end{pmatrix}$$

where f_i is the i-th component of the vector-valued function f.

Derivatives and gradients

The chain rule

- The chain rule is a fundamental theorem in calculus that describes how to compute the derivative of a composite function.
- If we have two functions $g : \mathbb{R}^m \to \mathbb{R}^p$ and $f : \mathbb{R}^d \to \mathbb{R}^m$, the composite function $h : \mathbb{R}^d \to \mathbb{R}^p$ is defined as $h(x) = g(f(x))$.
- The chain rule states that if f is differentiable at a point $x \in \mathbb{R}^d$ and g is differentiable at the point $f(x) \in \mathbb{R}^m$, then the composite function h is differentiable at point x, and its derivative is given by

$$J_{h(x)} = J_{g(f(x))} J_{f(x)}$$

Derivatives and gradients

The chain rule

Example

- Let $f : \mathbb{R}^2 \to \mathbb{R}^2$ and $g : \mathbb{R}^2 \to \mathbb{R}$ be defined as

$$f(x) = \begin{pmatrix} x_1^2 & x_2^3 \end{pmatrix}^T \qquad g(y) = y_1 + y_2$$

- The Jacobian of f and the gradient of g are

$$J_{f(x)} = \begin{pmatrix} 2x_1 & 0 \\ 0 & 3x_2^2 \end{pmatrix} \qquad \nabla g(y) = (1, 1)^T$$

- The composite function $h : \mathbb{R}^2 \to \mathbb{R}$ is defined as

$$h(x) = g(f(x)) = x_1^2 + x_2^3$$

- The gradient of h at point x is given by

$$\nabla h(x) = J_{f(x)}^T \nabla g(f(x)) = \begin{pmatrix} 2x_1, 3x_2^2 \end{pmatrix}^T$$

Derivatives and gradients

Optimality conditions

- Let's consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

- If f is differentiable and x^* is a local minimizer of f, then the following condition holds

$$\nabla f(x^*) = 0$$

- Let's prove this result

Proof -

By definition of a local minimizer, there exists a radius $r > 0$ such that for all $x \in \mathbb{R}^d$ such that $\|x - x^*\| \leq r$, we have $f(x^*) \leq f(x)$. This means that for $h \in \mathbb{R}^d$ such that $\|h\| \leq r$, we have

$$f(x^*) \leq f(x^* + h) = f(x^*) + \nabla f(x^*)^T h + o(\|h\|)$$

Simplifying by $f(x^*)$ and dividing by $\|h\|$ (which is non zero as h is non zero), we obtain,

$$\nabla f(x^*)^T \overline{h} \geq 0$$

with $\overline{h} = \frac{h}{\|h\|}$.
We can apply the same reasoning with $-h$ instead of h and obtain $\nabla f(x^*)^T \left(-\overline{h}\right) \geq 0$, which gives $\nabla f(x^*)^T \overline{h} \leq 0$.
Eventually, we have $\nabla f(x^*)^T \overline{h} = 0$ for all $\overline{h}$ such that $\|\overline{h}\| = 1$, and thus $\nabla f(x^*) = 0$

■

Gradient descent algorithm

Gradient descent algorithm

Algorithm

- The gradient descent algorithm is an iterative optimization algorithm used to find the minimum of a differentiable function.
- The basic idea behind gradient descent is to iteratively update the current solution in the direction of the negative gradient of the objective function, which points towards the steepest descent.
- The update rule for gradient descent is given by

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

where x_k is the current solution, α_k is the step size (or learning rate), and $\nabla f(x_k)$ is the gradient of the objective function at point x_k.

- The choice of step size α_k is crucial for the convergence and performance of the algorithm. It can be fixed or determined using techniques such as line search or adaptive methods.

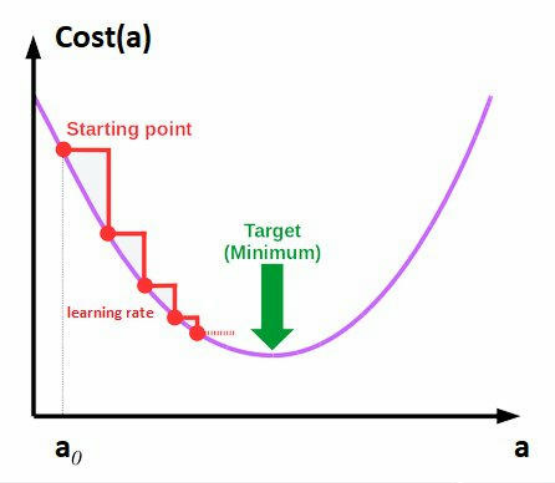
Gradient descent algorithm

Algorithm

- The pseudocode for the gradient descent algorithm is as follows:

Algorithm 1: Binary Search

```
1: procedure Gradient Descent(x_0, f)
2:     ▷ Initialize the solution
3:     x ← x_0
4:     while x_0 is not optimal do
5:         ▷ Compute the gradient at the
           current solution
6:         g ← ∇f(x)
7:         ▷ Choose a step size
8:         α ← some method
9:         ▷ Update the solution
10:        x ← x − αg
11:    end
12end
```



,

Gradient descent algorithm

Limitations

- Gradient descent can be sensitive to the choice of step size. A step size that is too large can lead to divergence, while a step size that is too small can result in slow convergence.
- The algorithm can get stuck in local minima or saddle points, especially in non-convex optimization problems.
- Gradient descent may require many iterations to converge, particularly for ill-conditioned problems where the objective function has steep and flat regions.
- Gradient descent only uses first-order information (the gradient) and does not take into account the curvature of the objective function, which can lead to inefficient updates.

Newton's method

Newton's method

Comparison with gradient descent

- Newton's method is an iterative optimization algorithm used to find the minimum of a **twice-differentiable** function.
- Unlike gradient descent, which only uses first-order information (the gradient), Newton's method also incorporates second-order information (the Hessian matrix) to make more informed updates.
- The update rule for Newton's method is given by

$$x_{k+1} = x_k - H_{f(x_k)}^{-1} \nabla f(x_k)$$

where $H_{f(x_k)}$ is the Hessian matrix of the objective function at point x_k.

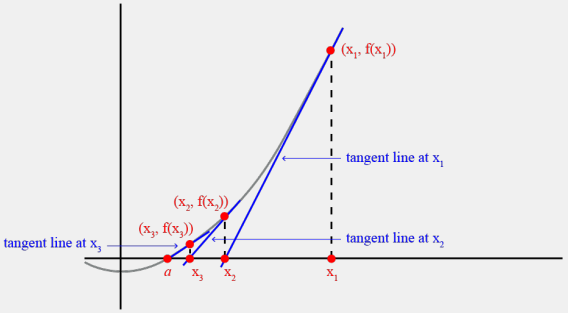
- By using the Hessian, Newton's method can adapt the step size and direction based on the local curvature of the objective function, potentially leading to faster convergence compared to gradient descent.

Newton's method

Finding zeroes of a function

- The original Newton's method is used to find the roots (zeroes) of a function $g : \mathbb{R}^n \to \mathbb{R}^n$.
- The update rule for finding the roots of g is given by

$$x_{k+1} = x_k - J^{-1}_{g(x_k)} g(x_k)$$



Calcworkshop.com

- This can be applied to get the zero of the gradient of a function to minimize and yields the previous iterate

Newton's method

Algorithm

- The pseudocode for Newton's method is as follows:

Algorithm 2: Binary Search

```
1: procedure Newton's Method(x_0, f)
2:     ▷ Initialize the solution
3:     x ← x_0
4:     while x_0 is not optimal do
5:         ▷ Compute the gradient and Hessian at the current solution
6:         g ← ∇f(x)
7:         H ← H_{f(x)}
8:         ▷ Compute the step size
9:         α ← Some method
10:        ▷ Update the solution
11:        x ← x − αH^{-1}g
12:    end
13end
```

Newton's method

Limitations

- Computing the Hessian matrix and its inverse can be computationally expensive, especially for high-dimensional problems, which can limit the scalability of Newton's method.
- The analytical computation of the Hessian may not be feasible for complex functions, requiring numerical approximations that can introduce errors
- Newton's method can be sensitive to the initial guess. A poor initial guess may lead to slow convergence or divergence.
- In practice, quasi-Newton methods (e.g., BFGS) are often used as they approximate the Hessian and can provide a good trade-off between convergence speed and computational cost.