Semaine 1 — Introduction à l'apprentissage automatique

De l'IA à l'approximation de fonctions

UdeM AI — Université de Montréal

Qu'est-ce que l'intelligence artificielle ?

L'**intelligence artificielle (IA)** étudie des systèmes capables de :

- percevoir leur environnement,
- traiter l'information,
- prendre des décisions en vue d'un objectif.

Vue synthétique :

$$\text{IA} = \text{Perception} + \text{Décision} + \text{Raisonnement}(+\ \text{Apprentissage})$$

L'apprentissage n'est pas toujours nécessaire, mais il est aujourd'hui central dans la majorité des systèmes modernes.

Où se situe l'apprentissage automatique ?

L'**apprentissage automatique (Machine Learning)** est un sous-domaine de l'IA.

Idée clé : au lieu de programmer explicitement une solution, on conçoit un système qui apprend à partir de données.

Hiérarchie utile :

$$\text{IA} \supset \text{Apprentissage automatique} \supset \text{Réseaux de neurones}$$

Le cœur du ML n'est pas l'algorithme, mais la représentation et l'apprentissage à partir d'exemples.

Apprentissage automatique — cadre minimal

On cherche à apprendre une relation inconnue :

$$x \mapsto y$$

On introduit un modèle paramétré :

$$\hat{y} = f_\theta(x)$$

Les paramètres θ sont ajustés à partir de données :

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \ell\big(f_\theta(x_i), y_i\big)$$

Objectif : bien prédire sur des données nouvelles, pas seulement observées.

Apprendre, c'est approximer une fonction

On suppose l'existence d'une fonction cible inconnue :

$$f : \mathbb{R}^d \to \mathbb{R}$$

La nature fournit des observations :

$$(x_i, y_i), \qquad y_i \approx f(x_i)$$

Problème central :

$$\text{trouver } f_\theta \text{ tel que } f_\theta(x) \approx f(x)$$

L'apprentissage automatique est un problème d'**approximation de fonctions à partir de données**.

Concrètement : approximation à partir de données

Dire *approximer une fonction à partir de données* veut dire :

- on ne connaît pas la règle exacte f (la vraie loi),
- on observe seulement des exemples (x_i, y_i),
- on construit une règle f_θ qui reproduit au mieux ces exemples.

Mais le but n'est pas de mémoriser les données :

on veut une règle qui marche sur de nouveaux x.

Donc le problème est à la fois :

- **mathématique** (approximation),
- **statistique** (généralisation).

Choisir une représentation pour f_θ

Le point clé n'est pas uniquement l'optimisation, mais le **choix de la forme de** f_θ.

Exemples de représentations possibles :

- polynômes,
- séries de Fourier,
- superpositions de fonctions non linéaires,
- grilles (fixes ou adaptatives),
- ansatz guidé par la structure du problème.

Ce choix encode des hypothèses sur la forme de la fonction à apprendre.

Concrètement : qu'est-ce que choisir la forme de f_θ ?

Choisir la forme de f_θ, c'est décider **comment** le modèle peut représenter une fonction.

Exemples (intuition) :

- **Polynômes** : on suppose que la fonction est bien décrite par une courbe lisse globale.
- **Fourier** : on suppose que la fonction est bien décrite par des oscillations (fréquences).
- **Superpositions non linéaires** : on combine des briques flexibles (non-linéarités).
- **Grilles adaptatives** : on met plus de résolution là où ça change vite.
- **Ansatz** : on impose une structure inspirée du domaine (physique, géométrie, etc.).

Deux modèles peuvent être optimisés pareil, mais **un seul** aura une forme adaptée aux données.

Bases classiques : polynômes et Fourier

Polynômes :

$$f_\theta(x) = \sum_{n=0}^{N} \theta_n x^n$$

Séries de Fourier (intervalle fini) :

$$f_\theta(x) = \sum_{|k| \leq K} \theta_k e^{ikx}$$

Ces bases sont :

- universelles sous certaines hypothèses,
- simples analytiquement,
- mais parfois inefficaces pour des structures locales.

Expressivité vs efficacité (paramètres & données)

Deux critères fondamentaux :

Expressivité Peut-on approximer une large classe de fonctions ?

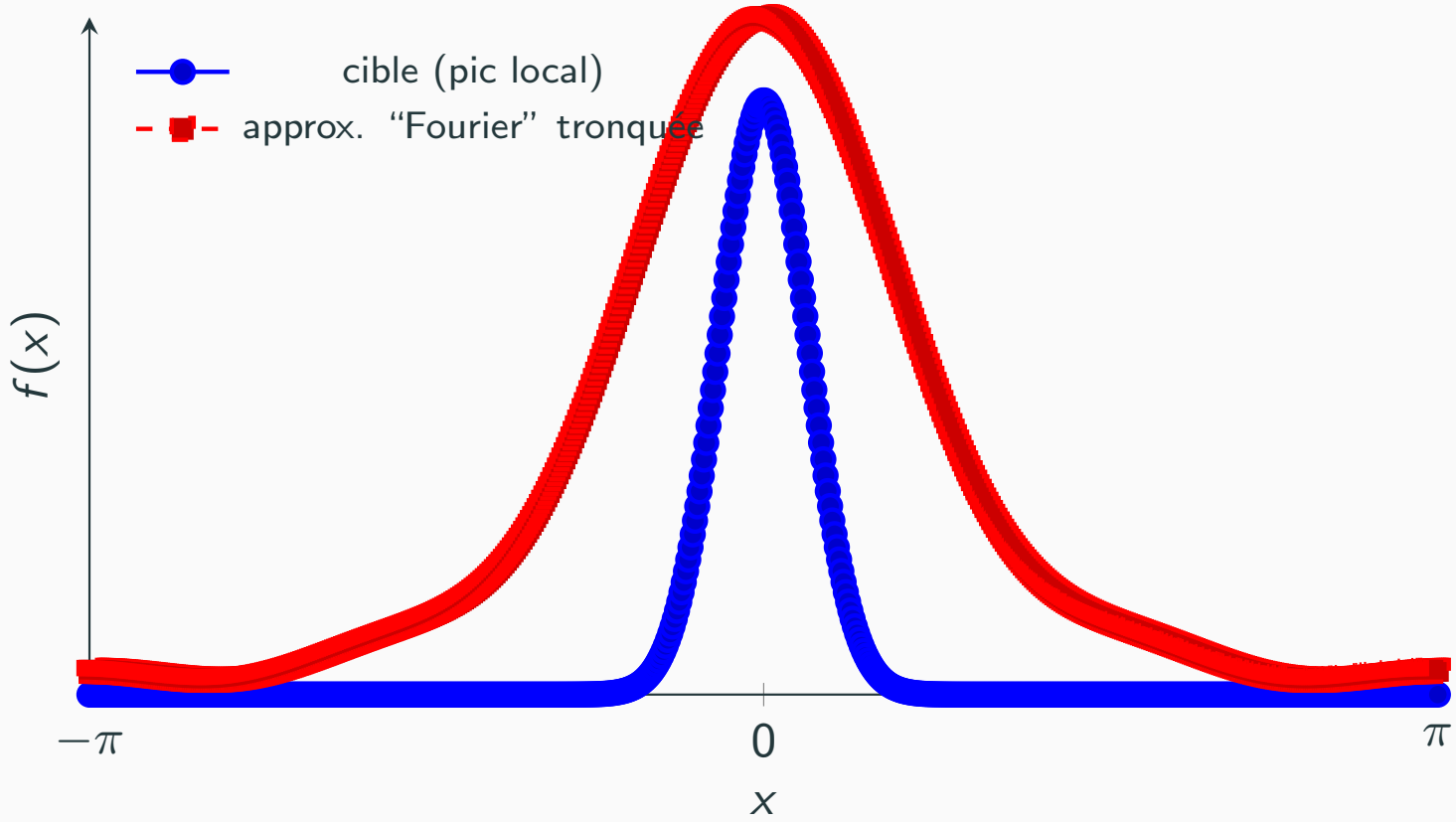
Efficacité Combien de **paramètres** et de **données** sont nécessaires pour obtenir une bonne approximation ?

Point important :

Une approximation expressive n'implique pas automatiquement une bonne généralisatio

(Être capable de tout représenter peut aussi rendre l'apprentissage instable ou trop sensible au bruit.)

Exemple visuel : un pic local et une approximation de Fourier



Idée : une base globale (comme Fourier) peut nécessiter beaucoup de termes pour capturer une structure très locale.

Que signifie « structure locale » concrètement ?

Une fonction présente une **structure locale** lorsque :

- elle varie fortement dans certaines régions,
- et très peu ailleurs.

Exemples intuitifs :

- un pic étroit,
- une transition rapide entre deux régimes,
- une zone « intéressante » au milieu d'un signal autrement simple.

Dans ces cas, utiliser la même « résolution » partout est souvent inefficace.

Pourquoi est-ce important en pratique ?

En pratique, les données réelles sont souvent :

- hétérogènes,
- bruitées,
- concentrées dans certaines régions de l'espace.

On veut donc :

- allouer plus de capacité là où c'est nécessaire,
- éviter d'en gaspiller là où la fonction est simple.

C'est exactement ce que cherchent les bonnes représentations : **être adaptées à la structure des données** (et donc mieux généraliser).

Message clé

L'apprentissage automatique n'est pas magique.

C'est un cadre pour :

- approximer des fonctions inconnues,
- à partir de données,
- via des représentations efficaces.

En pratique, la performance dépend souvent davantage des hypothèses implicites du modèle que de l'algorithme d'optimisation lui-même.

Références

- F. Marquardt, *Advanced Machine Learning for Physics, Science, and Artificial Scientific Discovery*, série de cours YouTube.
- A. Ng, *Machine Learning*, Coursera.