Artificial intelligence and chemistry

Tony Bonnaire

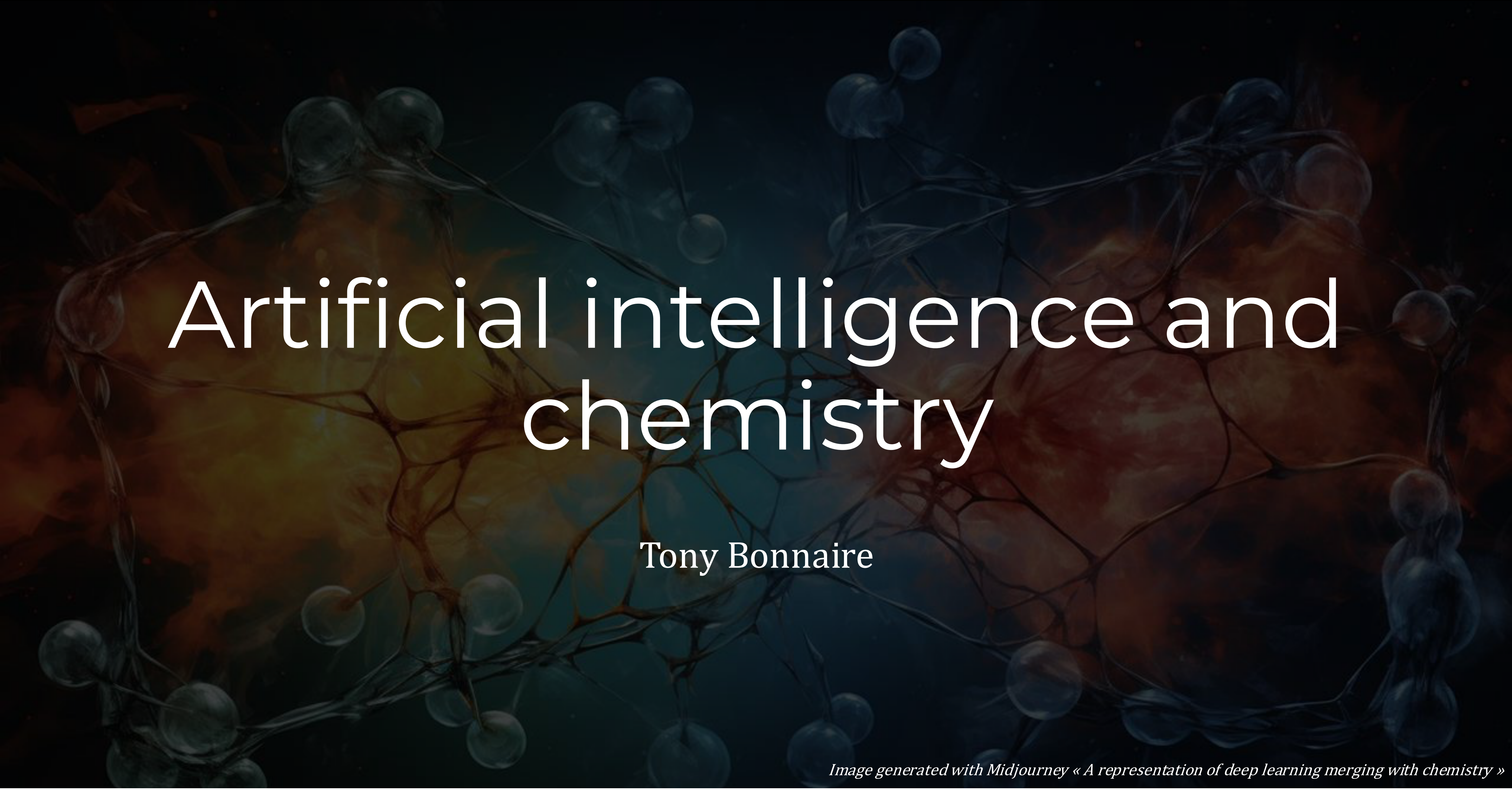


Image generated with Midjourney « A representation of deep learning merging with chemistry »

Organisation of the course

Given by: Tony Bonnaire (+ **Aurora González Sanz** for the project)

Format: Lectures + hands-on sessions then data challenge (in chemistry)

Exam: Paper analysis (30%) + MCQ (30%) + oral presentation for the challenge (40%)

Aim: Introduce you to the basics of ML principles and carry out a project

Some references:

- Deep Learning: Foundations and Concepts, Bishop & Bishop, 2023,
- Deep Learning, Goodfellow et al., 2016,
- Deep Learning with Python, Chollet, 2016,
- Learning Theory from First Principles, Francis Bach, 2024,
- https://challengedata.ens.fr: a bank of data science challenges to apply all the things we will learn in this course

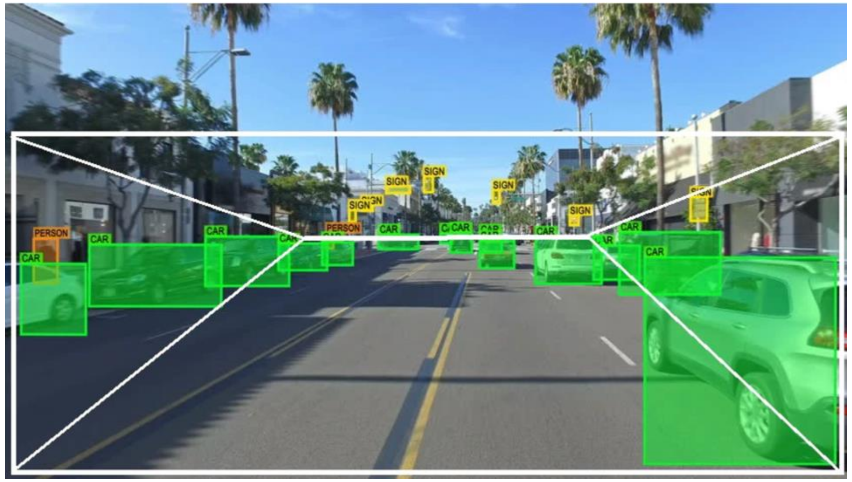
Contact: tony.bonnaire@ens.fr

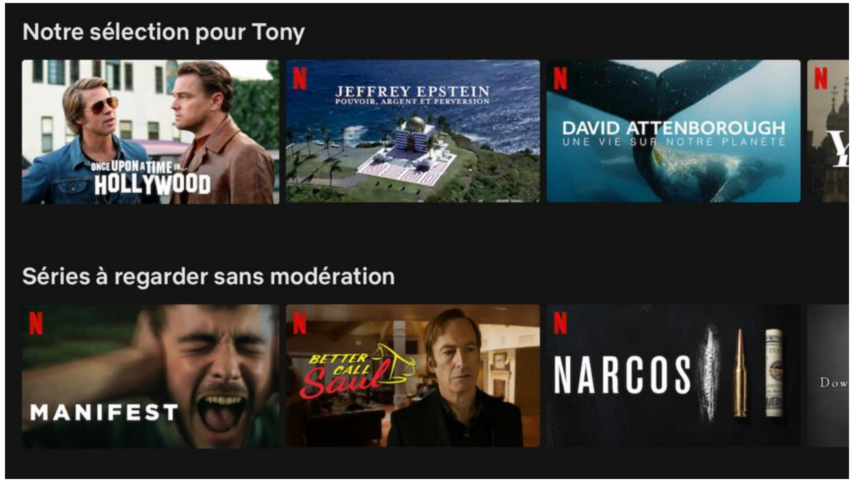
Material on Github: github.com/tbonnair/

Intelligent systems

AI goal

Design **systems** capable of performing complex tasks requiring ***intelligence*** (i.e. using reasoning, perception or language) **to take decisions** and **make predictions**.





AI revolution

Some (selected) AI breakthroughs

1958
Perceptron

2017
Transformers

1956 – 'Creation' of AI as a discipline at the Dartmouth workshop

1982
Hopfield networks
Backpropagation

1997 – DeepBlue defeats Garry Kasparov at chess

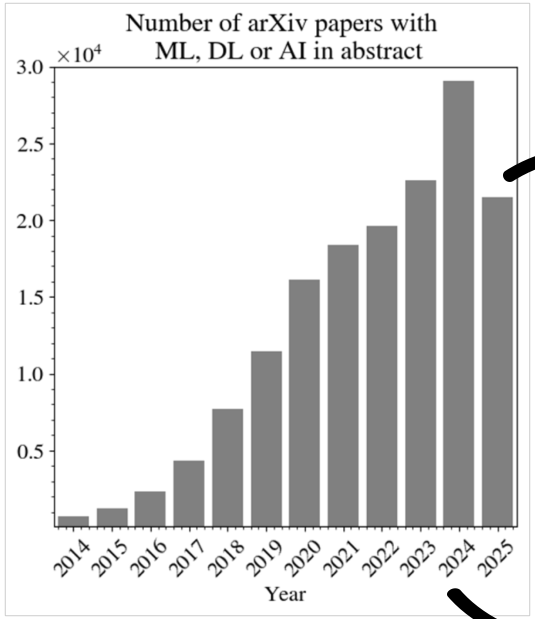
2012 – AlexNet (CNN) wins ImageNet competition

2016 – AlphaGo (RL) defeats Lee Sedol at Go

2018 – AlphaFold solves protein reconstruction

2020s – Generative AI (ChatGPT, LLAMA, Midjourney, DALL-E, etc.)

AI in science



2025 not over yet!

80 papers a day in average in 2024 (!)

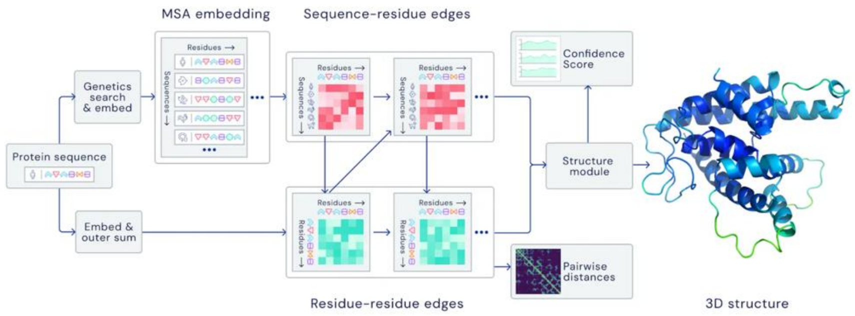
ML in science

Some scientific applications

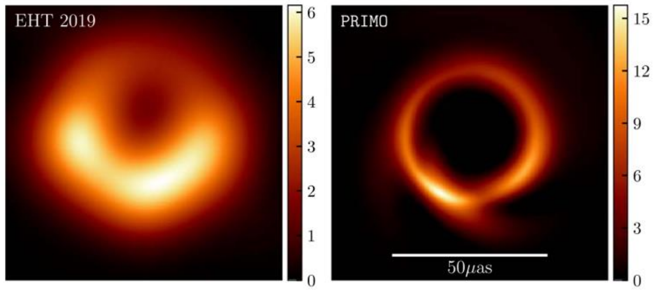
Healthcare
- Drug discovery
- Protein structure reconstruction

Astrophysics and cosmology
- Galaxy deblending
- Image restoration
- Source separation

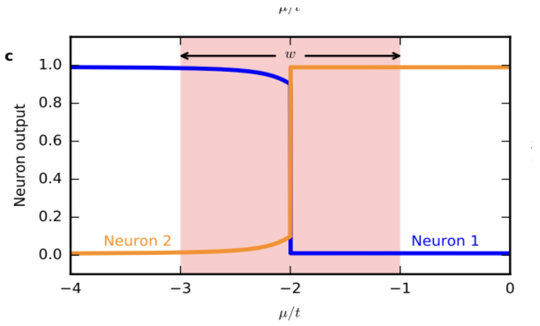
Theoretical physics
- Study phase transitions
- Discover experiments and equations



[Jumper et al., 2021]



[Medeiros et al., 2023]



[Van Nieuwenburg et al., 2017]

... And many more (climate forecast, fraud detection in cybersecurity, binding energies in quantum chemistry)

What is “learning”?

Machine Learning came as a solution to design intelligent systems, replacing handcrafted decision rules by **learnt rules** using **training data** and **optimisation** of **parameterised models**.

Training phase

Data ⇄ Model → Prediction on new data

Generalisation phase

Data	Model
Images	Linear models
Tabular	Trees and forests
Sound	Neural networks
Text	CNN
....	...

What is “learning”? An example

Training phase

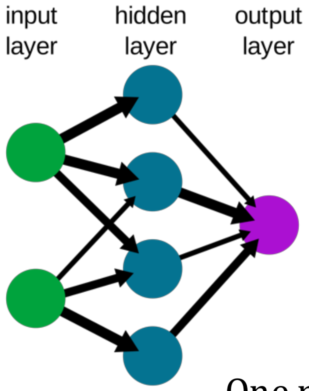
Data ⇄ Model

→

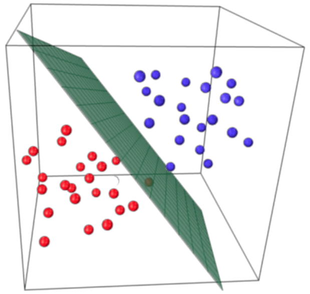
Generalisation phase

Prediction on new data

Images of a ”cat” or “dog”



One model



Another model

”cat” or “dog” ?

ML and the scientific method

Training phase

Data ⇄ Model →

Generalisation phase

Prediction on new data

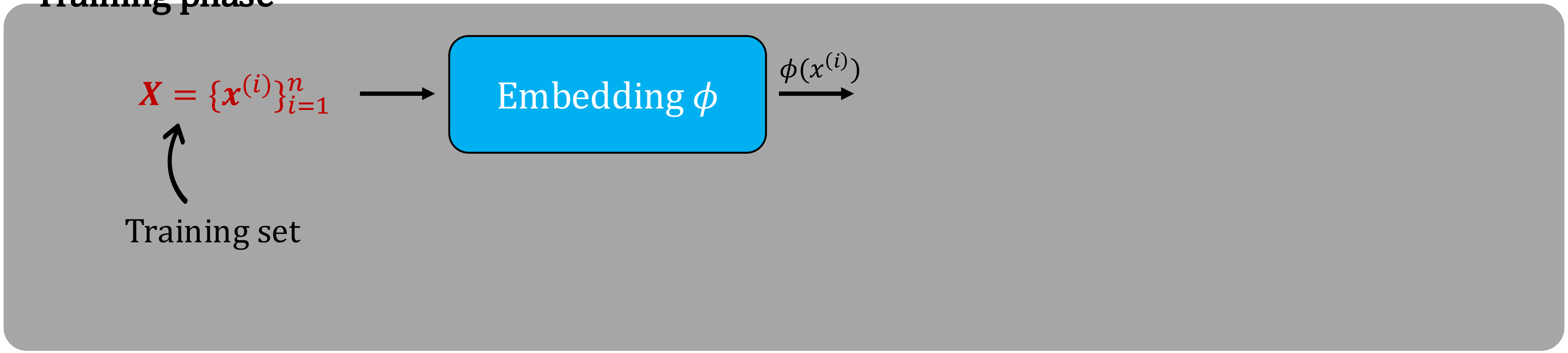
...In fact, all this is close to what you know!

The scientific method

Measurements from an experiment ⇄ Build a model or propose a theory → Test with new experiments

Validate/invalidate your model

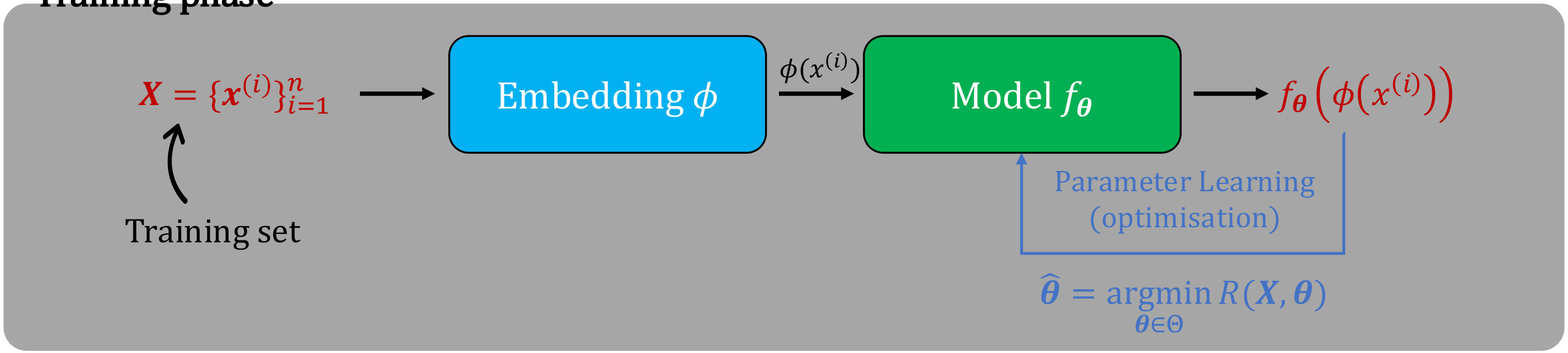
Training phase



1. Data X are **unstructured**, sometimes **noisy** and **unprocessed** like pixels of an image or sequence of characters or words.
2. The embedding $\phi(x^{(i)})$ is a **structured**, **numerical** vector representation of the data whose elements are **meaningful features**. It depends on the data and the purpose. It can be **handcrafted or learnt**.

Finding a good embedding is a central part of ML: it eases the problem by preserving the essential structure of the data that matters for the task.

Training phase

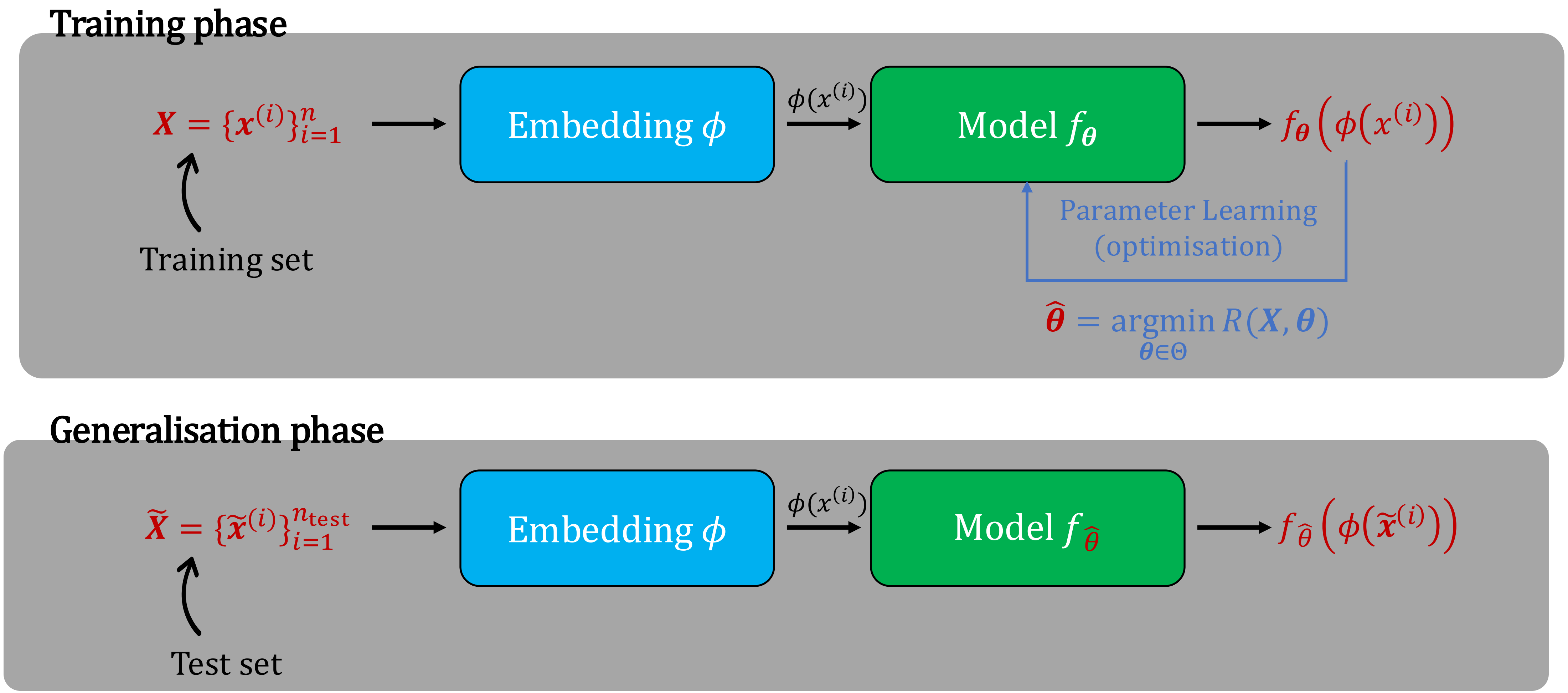


Some notations and terminologies:

- $\boldsymbol{x}^{(i)} \in \mathbb{R}^{d}$ is *one* ***training data*** (there are n of them),
- $\phi\left(x^{(i)}\right) \in \mathbb{R}^{d'}$ is an embedding of $x^{(i)}$ sometimes called ***feature vector,***
- $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^{p}$ are the ***parameters*** of the model,
- $R(\boldsymbol{X}, \boldsymbol{\theta})$ is the ***risk*** and measures the error of the model with parameters $\boldsymbol{\theta}$ on data $\boldsymbol{X}$.

At the end of the training procedure, we have a model $f_{\widehat{\theta}}$ committing an error of $R_{\text{train}} = R(\boldsymbol{X}, \widehat{\boldsymbol{\theta}})$ on the training set.

ML building blocks



Using the test set, we can evaluate the test error $R_{\text{test}} = R(\widetilde{\boldsymbol{X}}, \widehat{\boldsymbol{\theta}})$ and compare it to R_{train} to detect **generalisation issues** (**overfitting** or **underfitting**).

Supervised learning

$\boldsymbol{x}^{(i)}$ is the i th **data vector** of the training base, and $y^{(i)}$ is called the **target (or predicted) variable**

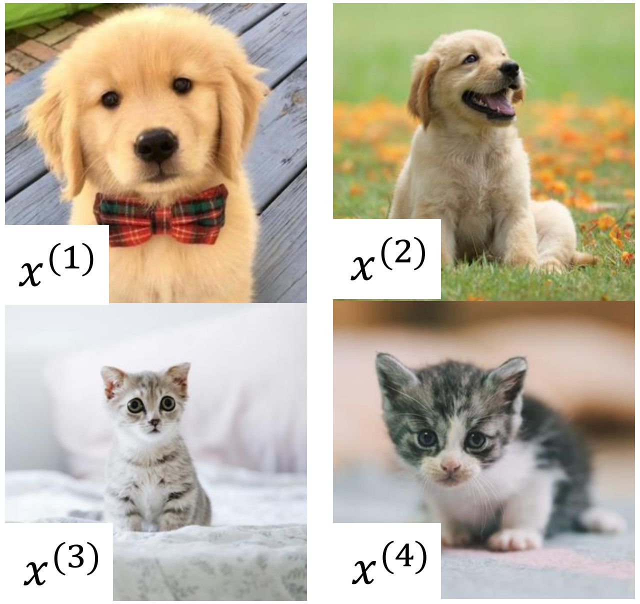
- Training data are actually X and Y coming as pairs

$$\boldsymbol{X} = \left\{\left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right)\right\}_{i=1}^{n}, \qquad \left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right) \in \mathbb{X} \times \mathbb{Y}$$

- If $\mathbb{Y}$ is continuous, then the task is called **regression**, and if $\mathbb{Y}$ is discrete, then it is a **classification** problem.

Example: Determine if an image encodes a cat or a dog
(called a **classification** task)

$X =$



$Y = \{1, 1, 0, 0\}$

$f_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) = \hat{\boldsymbol{y}}^{(i)} \longrightarrow f_{\hat{\theta}}$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, R(\boldsymbol{X}, \boldsymbol{\theta})$$

$\underbrace{\qquad\qquad}_{\text{Training data}}$ $\underbrace{\qquad\qquad}_{\text{Model}}$ $\underbrace{\qquad\qquad}_{\text{Optimisation}}$

Supervised learning

- Training data are actually X and Y coming as pairs

$$\boldsymbol{X} = \left\{\left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right)\right\}_{i=1}^{n}, \qquad \left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right) \in \mathbb{X} \times \mathbb{Y}$$

- If $\mathbb{Y}$ is continuous, then the task is called **regression**, and if $\mathbb{Y}$ is discrete, then it is a **classification** problem.

- Ideally, we would like to minimise the **expected risk**, i.e. the **expected value of a loss function** $\ell(y, \hat{y})$

$$R(\boldsymbol{X}, \boldsymbol{\theta}) = \mathbb{E}_{X,y}[\ell(y, \hat{y})]$$

Loss function: measures how bad your model is on a single example

However, we do not know $p(X, y)$ so in practice we rely on the **empirical risk** instead

$$\hat{R}(\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y^{(i)}, \hat{y}^{(i)}\right).$$

Families of ML problems

Supervised learning

- Training data are actually X and Y coming as pairs

$$\boldsymbol{X} = \left\{\left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right)\right\}_{i=1}^{n}, \qquad \left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right) \in \mathbb{X} \times \mathbb{Y}$$

- If $\mathbb{Y}$ is continuous, then the task is called regression, and if $\mathbb{Y}$ is discrete, then it is a classification problem.

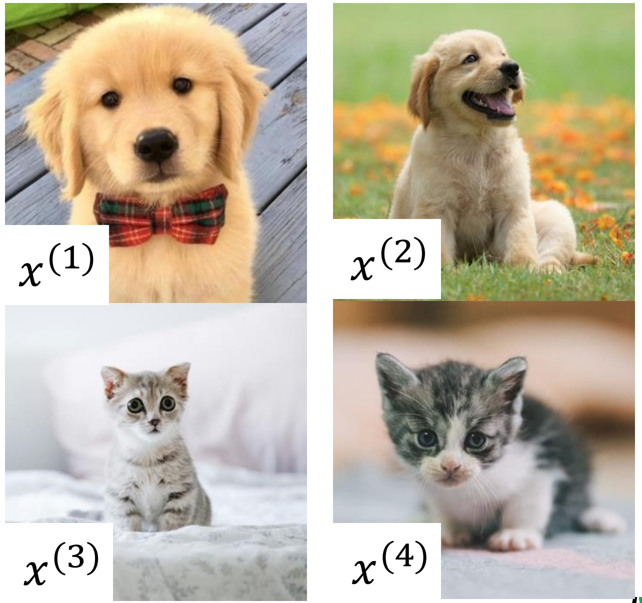
Examples of tasks: Classification | Regression | Timeseries prediction | Segmentation

Examples of models: Artificial Neural network | Random forest | Linear regression | Logistic regression | Naïve Bayes | Nearest neighbours

Unsupervised learning

- Training data are the set of $\boldsymbol{x}^{(i)}$'s only; no known results to predict
- In unsupervised learning, one seeks **patterns or structures** in X without prior labels
- Usually boils down to model the probability distribution of the dataset

$X =$



Example: Generate new images of cats and dogs (called a **generation** task)

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = p_{\theta}(\boldsymbol{x}) \longrightarrow f_{\hat{\theta}}$$

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, R(\boldsymbol{X}, \boldsymbol{\theta})$$

such that $p_{\theta}(\boldsymbol{x}) \approx p(\boldsymbol{x})$

Training data

Model

Optimisation

Families of ML problems

Unsupervised learning

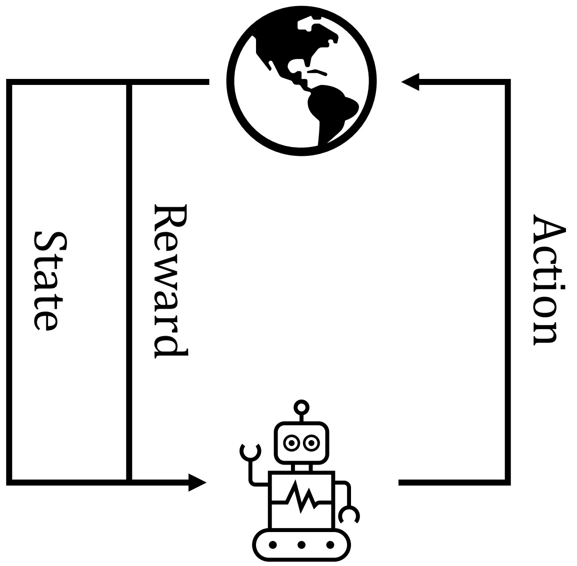
- Training data are the set of $\boldsymbol{x}^{(i)}$'s only; no known results to predict
- In unsupervised learning, one seeks **patterns or structures** in X without prior labels
- Usually boils down to model the probability distribution of the dataset

Examples of tasks: Clustering | Data augmentation | Dimensionality reduction | Sampling

Examples of models: Autoencoder | Boltzmann Machine | Diffusion models | Gaussian mixture model | Generative Adversarial network

Reinforcement learning

- The philosophy is different: the model does not try to “imitate” like in supervised learning nor to find patterns but “tries” things
- It is based on an **agent** interacting with an **environment**
- The agent tries to find the best possible sequence of states and actions to **maximise a reward**

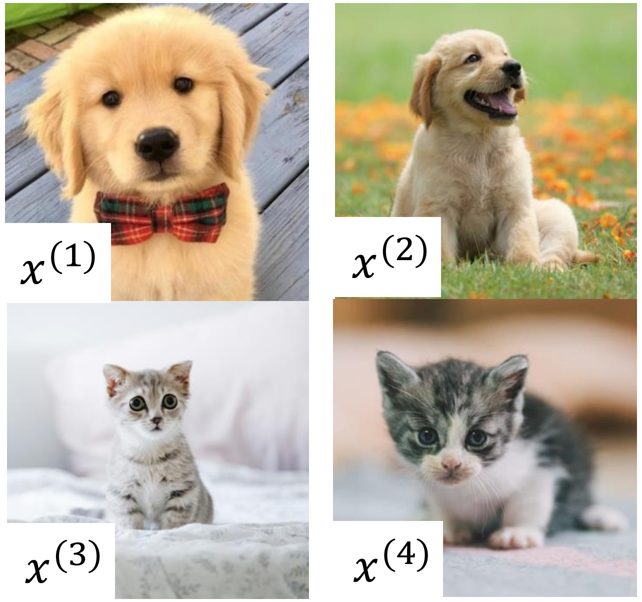


Examples of tasks: Game theory | Robotics | Autonomous driving

Examples of models: Markov decision processes | Q-networks | Deep policy gradient

Not discussed in this course, but a very good reference is [Reinforcement Learning – An introduction, Sutton and Barto, 2018]

Why do we need ML?



$d \approx 10^6$

- To sample a $[0,1]^d$ space with a shortest distance to a test point at most ϵ, we need $n_{\text{train}} \geq \epsilon^{-d} = e^{-d \log \epsilon}$
- **$d \approx 80$ requires more samples than the number of atoms in the universe**

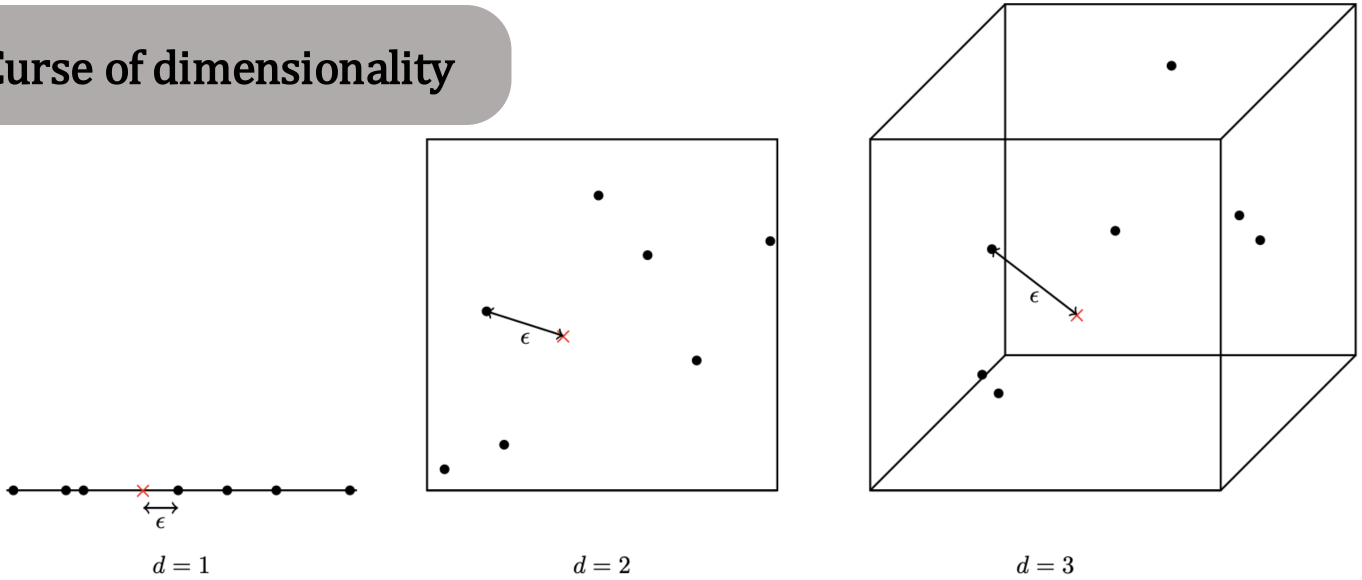
1-nearest neighbour

- A simple classification rule is for instance associating to a data the label of its closest neighbour in the d-dimensional space.

$$\hat{y} = y^{(m)} \text{ with } m = \text{argmin}_i \left\| x - x^{(i)} \right\|_2^2$$

- **In this case $R_{\text{train}} = 0$ but R_{test} is very large! Why?**

Curse of dimensionality



Why do we need ML?

Traditional methods typically break down in high-dimensional spaces (**curse of dimensionality**) and it is impossible to design handcrafted decision rules for complex tasks.

IF THERE IS ONLY ONE THING TO REMEMBER FROM THIS CLASS

The **curse of dimensionality** is the **central problem of machine learning.** To fight it, ML relies on **prior information** about the problem:

- **Reduce the dimensionality**: select a subset of meaningful features (or their interactions) through appropriate embeddings.
- Exploit **structures** in the data (invariances, sparsity, long/short range correlations, etc.) to define the model,
- **Penalise** complex models leading to poor generalisation performances using regularisation.