Artificial intelligence and chemistry

Tony Bonnaire

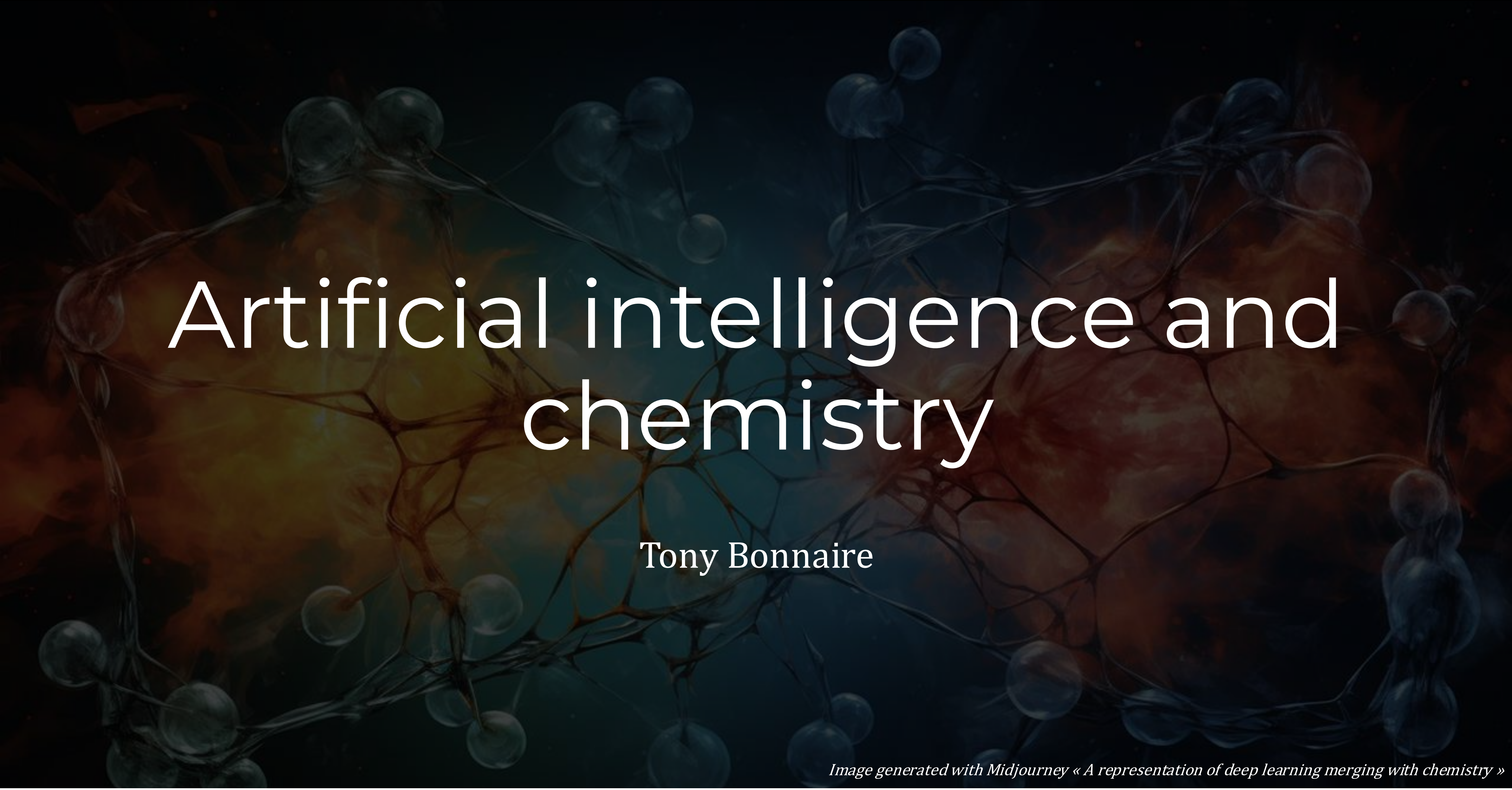


Image generated with Midjourney « A representation of deep learning merging with chemistry »

Given by: Tony Bonnaire (+ **Pablo Mas** for the project)

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

Exam: Paper analysis (50%) + oral presentation for the challenge (50%)

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

Some references:

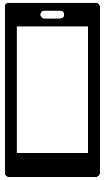
- Deep Learning: Foundations and Concepts, Bishop & Bishop, 2023,
- Deep Learning, Goodfellow et al., 2016,
- Deep Learning with Python, Chollet, 2016,
- Learning Theory from First Principles, Francis Bach, 2024,
- https://challengedata.ens.fr: a bank of data science challenges to apply all the things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

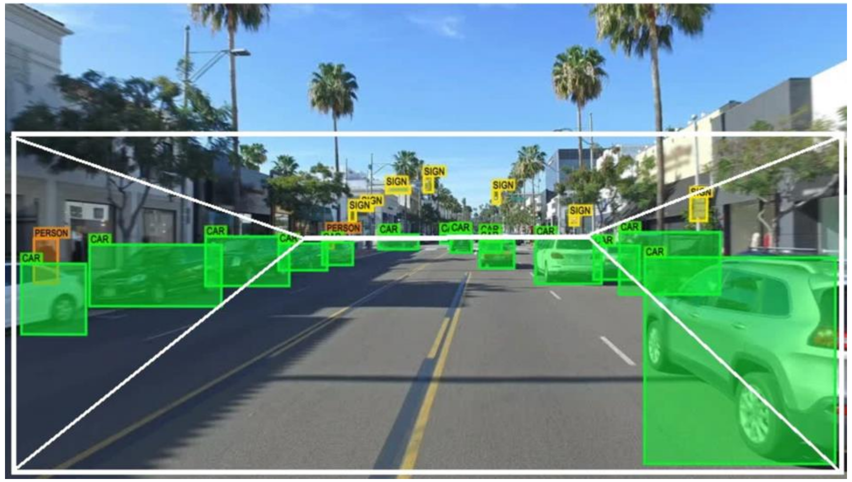
Contact: tony.bonnaire@ens.fr

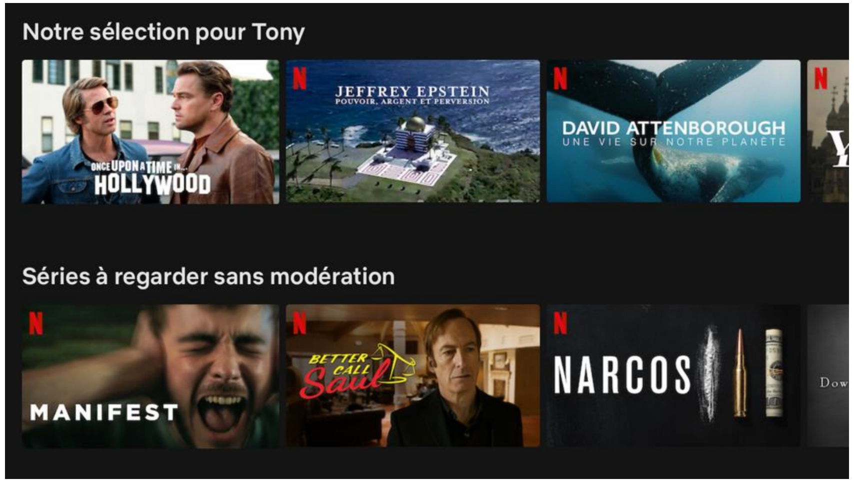
Material on Github: github.com/tbonnair/

AI goal

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

Some (selected) AI breakthroughs

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

1958 Perceptron

1982 Hopfield networks Backpropagation

1997 — DeepBlue defeats Garry Kasparov at chess

2012 — AlexNet (CNN) wins ImageNet competition

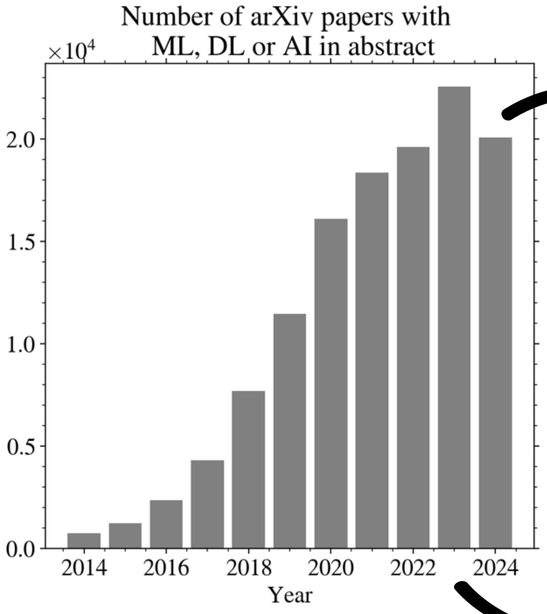
2016 — AlphaGo (RL) defeats Lee Sedol at Go

2017 Transformers

2018 — AlphaFold solves protein reconstruction

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

AI in science



2024 not over yet!

60 papers a day in average in 2023 (!)

Some scientific applications

Healthcare

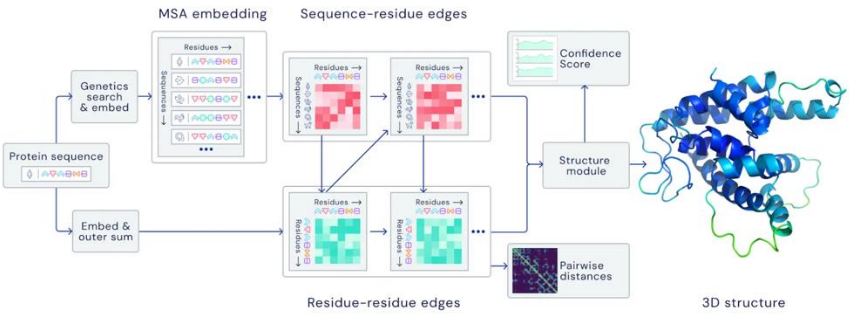
- Drug discovery
- Protein structure reconstruction

Astrophysics and cosmology

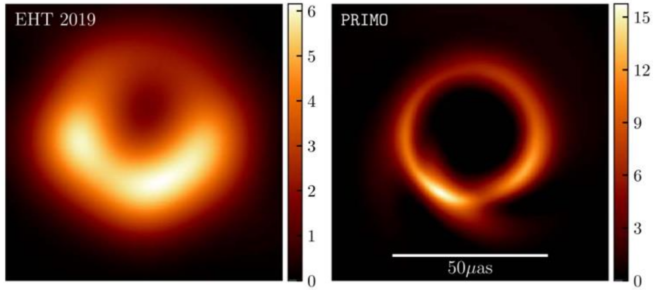
- Galaxy deblending
- Image restoration
- Source separation

Theoretical physics

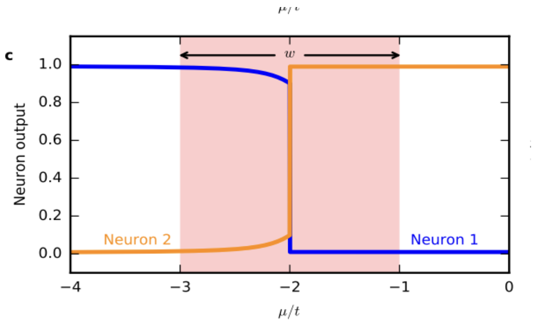
- Study phase transitions
- Discover experiments and equations



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 → Generalisation phase: Prediction on new data

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

What is “learning”? An example

Training phase

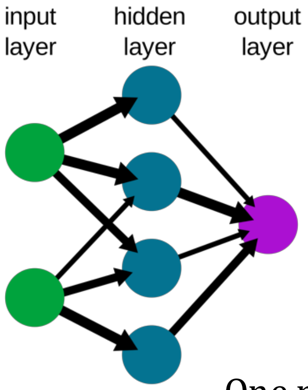
Data ⇄ Model

→

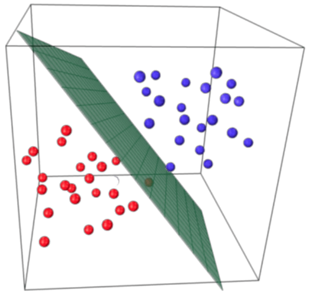
Generalisation phase

Prediction on new data

Images of a ”cat” or “dog”



One model



Another model

”cat” or “dog” ?

Training phase

Data ⇄ Model

→

Generalisation phase

Prediction on new data

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

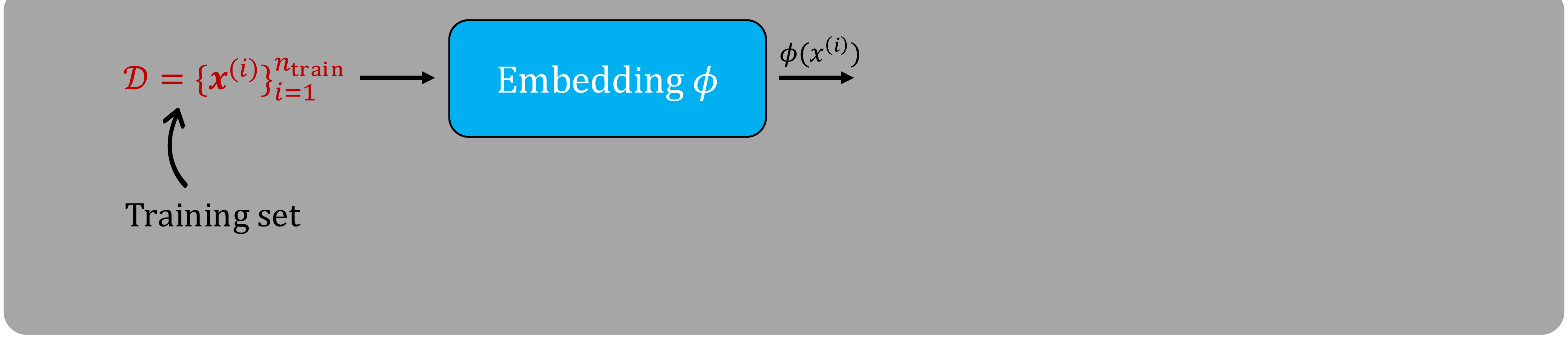
The scientific method

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

Validate/invalidate your model

ML building blocks

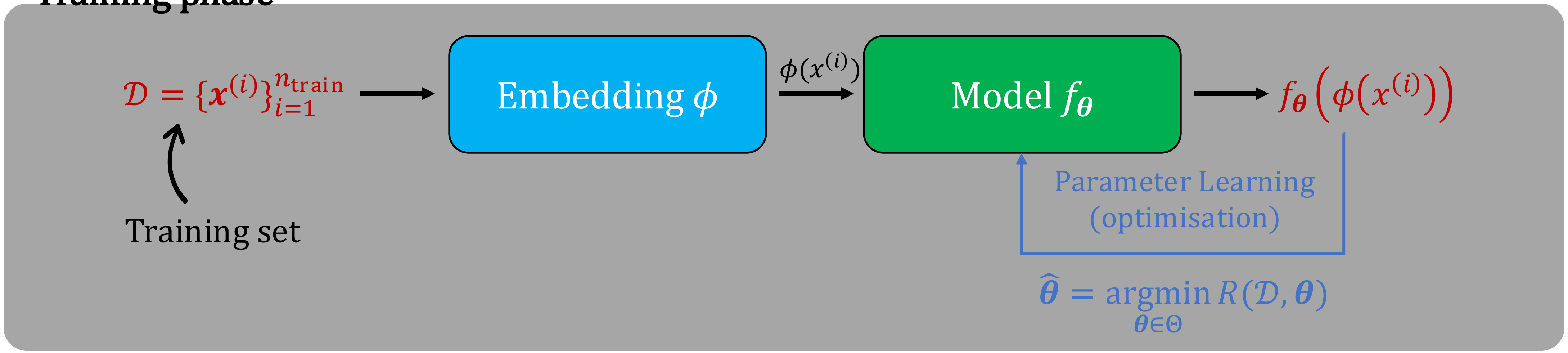
Training phase



1. Data 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** that 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 but makes it solvable using simple models.

Training phase

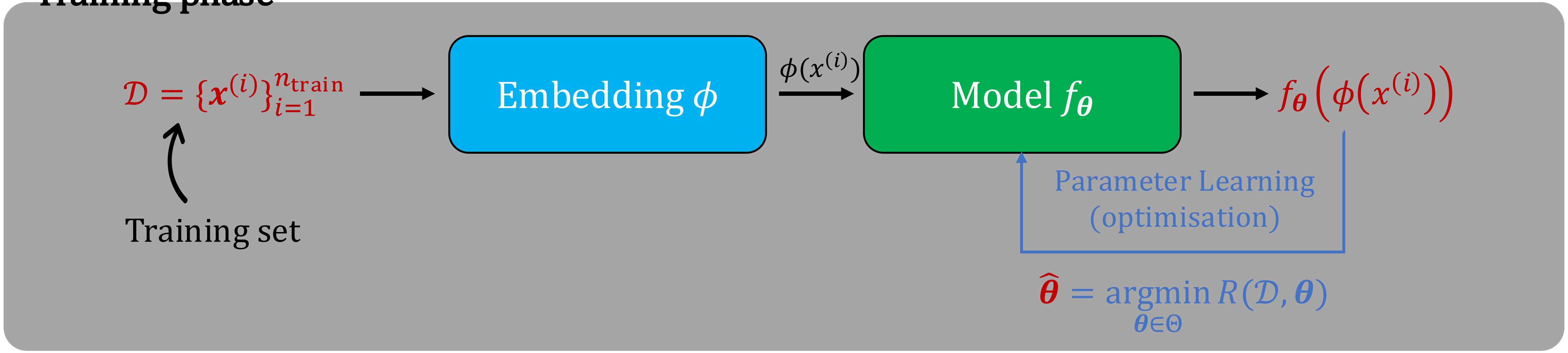


Some notations and terminologies:

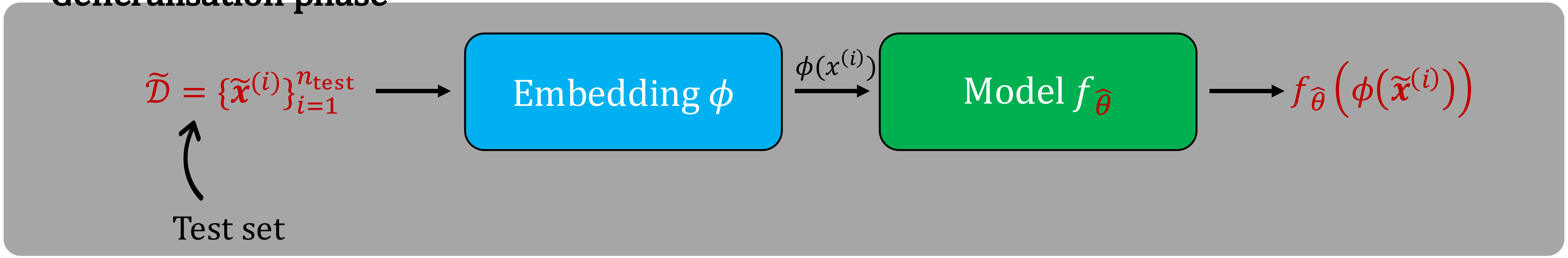
- $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ is *one* ***training data*** (there are n_{train} of them),
- $\phi\left(x^{(i)}\right) \in \mathbb{R}^{d'}$ is an embedding of $x^{(i)}$ sometimes called ***feature vector,***
- $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^p$ are the ***parameters*** of the model,
- $R(\mathcal{D}, \boldsymbol{\theta})$ 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 $R_{\text{train}} = R(\mathcal{D}, \widehat{\boldsymbol{\theta}})$ on the training set.

Training phase



Generalisation phase



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

Families of ML problems

Supervised learning

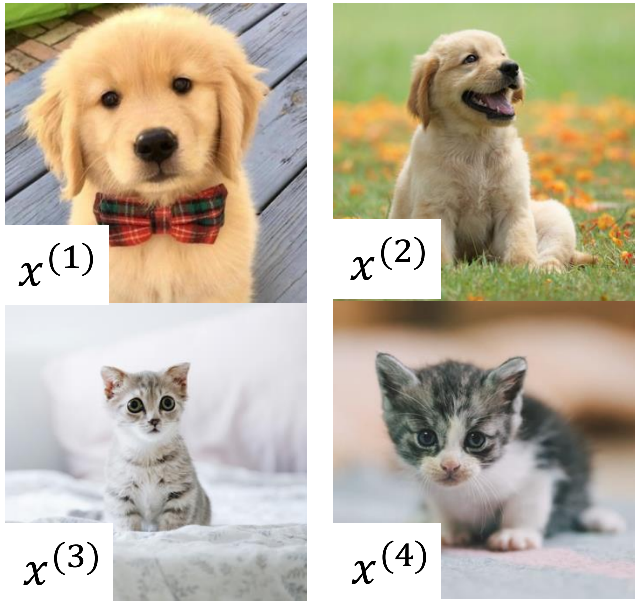
- Training data are actually X and Y coming as pairs

$$\mathcal{D} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{n_{\text{train}}}, \qquad (\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) \in \mathbb{X} \times \mathbb{Y}$$

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

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

$X =$



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

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

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

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

Training data | Model | Optimisation

Supervised learning

- Training data are actually X and Y coming as pairs

$$\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right)\right\}_{i=1}^{n_{\text{train}}}, \qquad \left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right) \in \mathbb{X} \times \mathbb{Y}$$

- 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** $\ell(y, \hat{y})$

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

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

$$\hat{R}(\mathcal{D}, \boldsymbol{\theta}) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \ell\left(y^{(i)}, \hat{y}^{(i)}\right).$$

Supervised learning

- Training data are actually X and Y coming as pairs

$$\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right)\right\}_{i=1}^{n_{\text{train}}}, \qquad \left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right) \in \mathbb{X} \times \mathbb{Y}$$

- 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

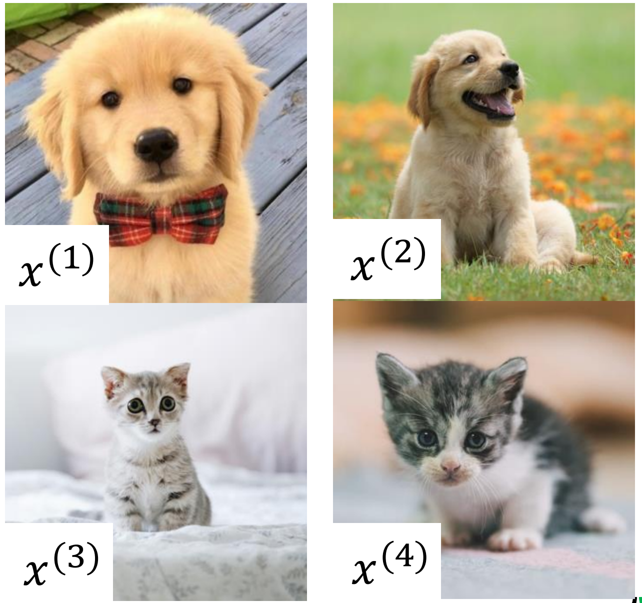
Examples of models: Artificial Neural network | Random forest | Linear regression | Logistic regression | Naïve Bayes | Nearest neighbours

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

$X =$



$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$

Training data

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

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

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

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

Model

Optimisation

Unsupervised learning

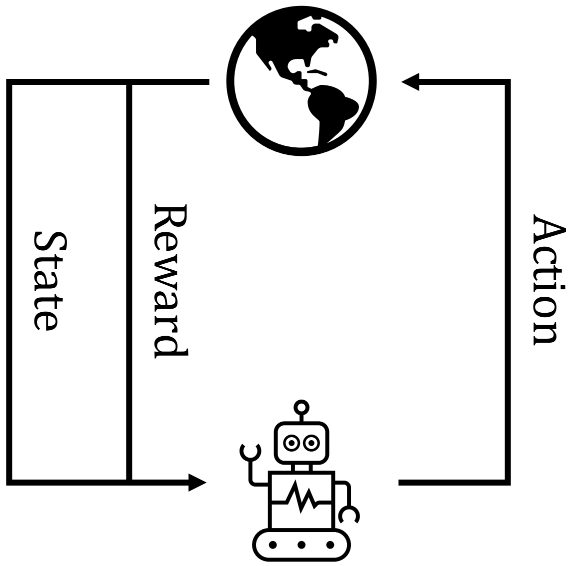
- Training data are the set of $\boldsymbol{x}^{(i)}$'s only; no known results to predict
- In unsupervised learning, one seeks **patterns or structures** in X without prior labels
- Usually boils down to model the probability distribution of the dataset

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**

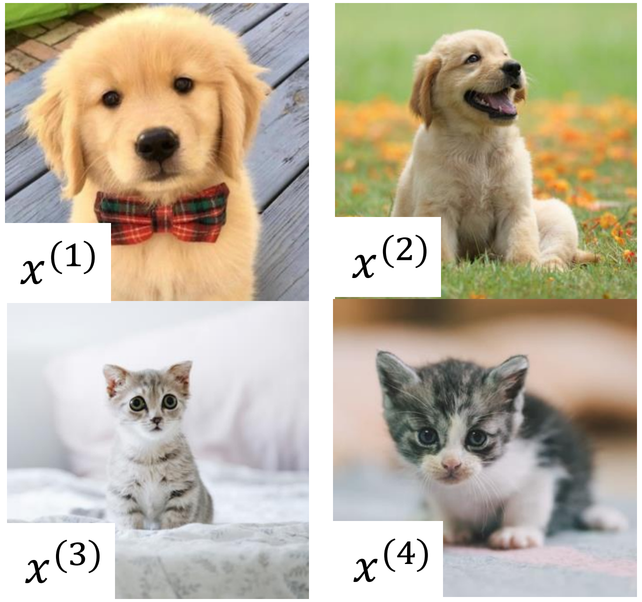


Examples of tasks: Game theory | Robotics | Autonomous driving

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**

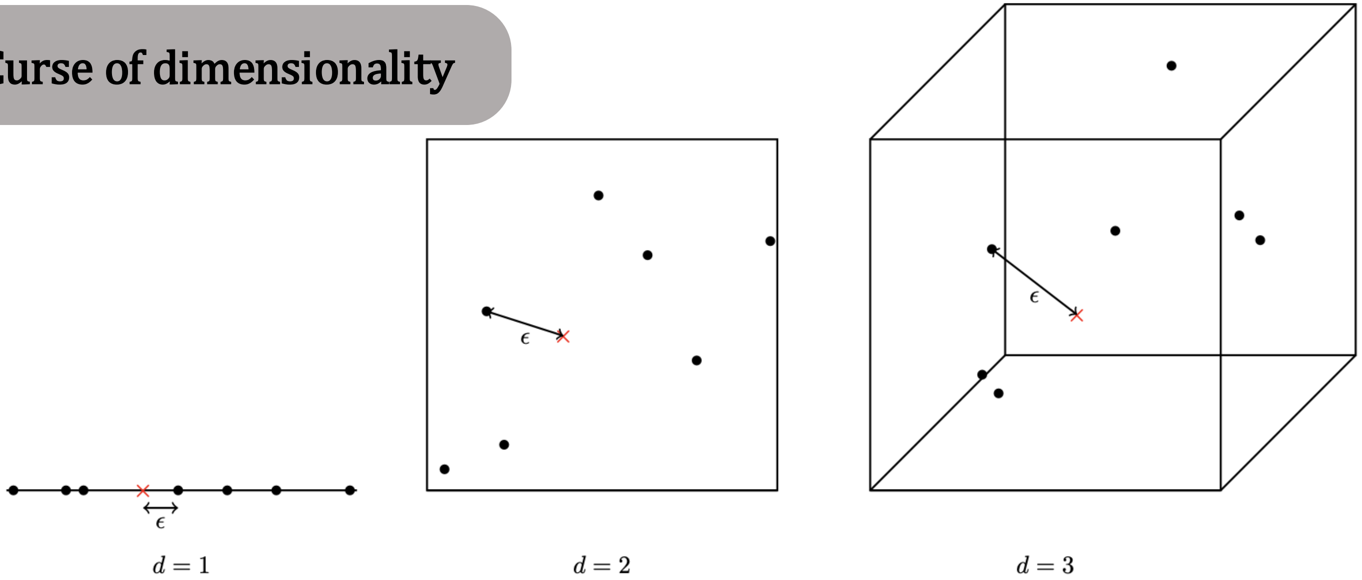
1-nearest neighbour

- 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



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

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

- **Find an appropriate embedding** or feature representation of the data to simplify the problem,
- Exploit **structures** in the data (invariances, sparsity, long-range correlations, etc.) to define the model,
- **Penalise** complex models leading to poor generalisation performances using regularisation.

Linear models on feature vectors

Contents:

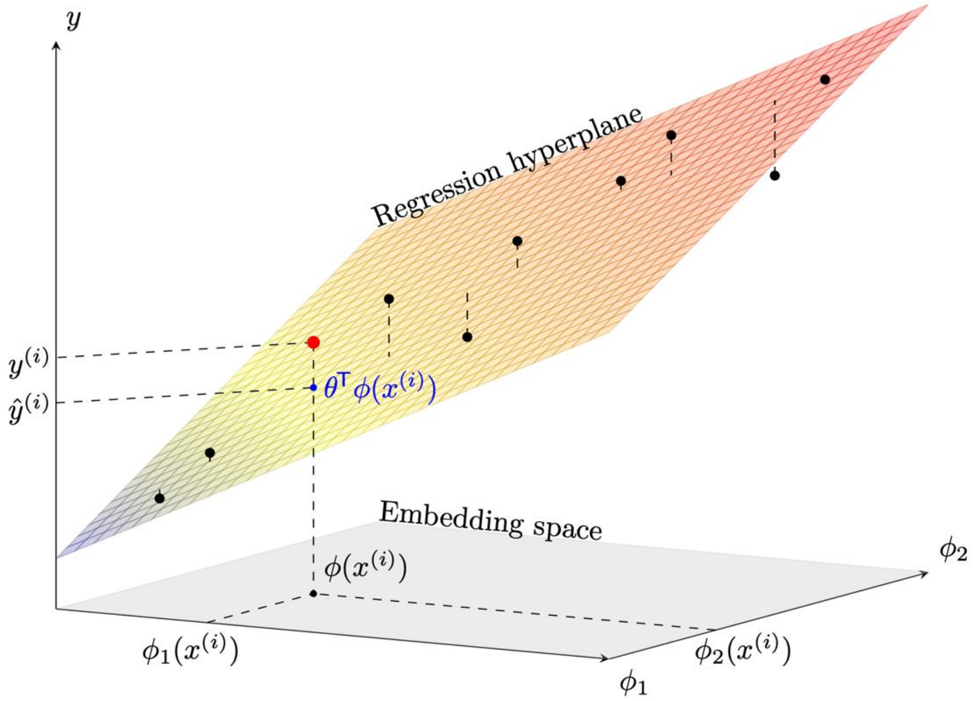
- *Linear regression model*
- *L2-loss for regression and normal equations*
- *Linear classification model*
- *Softmax function, cross-entropy loss for classification*

Linear regression

- What kind of problems one can solve efficiently, even in large dimensions? → **Linear systems!**
- Let us talk first about regression: the answer is modelled as

$$f_{\boldsymbol{\theta}}\left(\phi_1^{(i)}, \phi_2^{(i)}, \cdots\right) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}^{(i)} = \hat{y}^{(i)}$$

- It is sometimes convenient to add an affine term (also called **bias** in the neural network literature), which can be absorbed in the feature vector making it of dimension $d' + 1$ where $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \cdots]^{\mathrm{T}}, \boldsymbol{\phi}^{(i)} = \left[1, \phi_1^{(i)}, \phi_2^{(i)}, \cdots\right]^{\mathrm{T}}$.
- **Geometric interpretation**: projection of an embedding vector onto a **hyperplane** parameterised by $\boldsymbol{\theta}$.



Linear regression

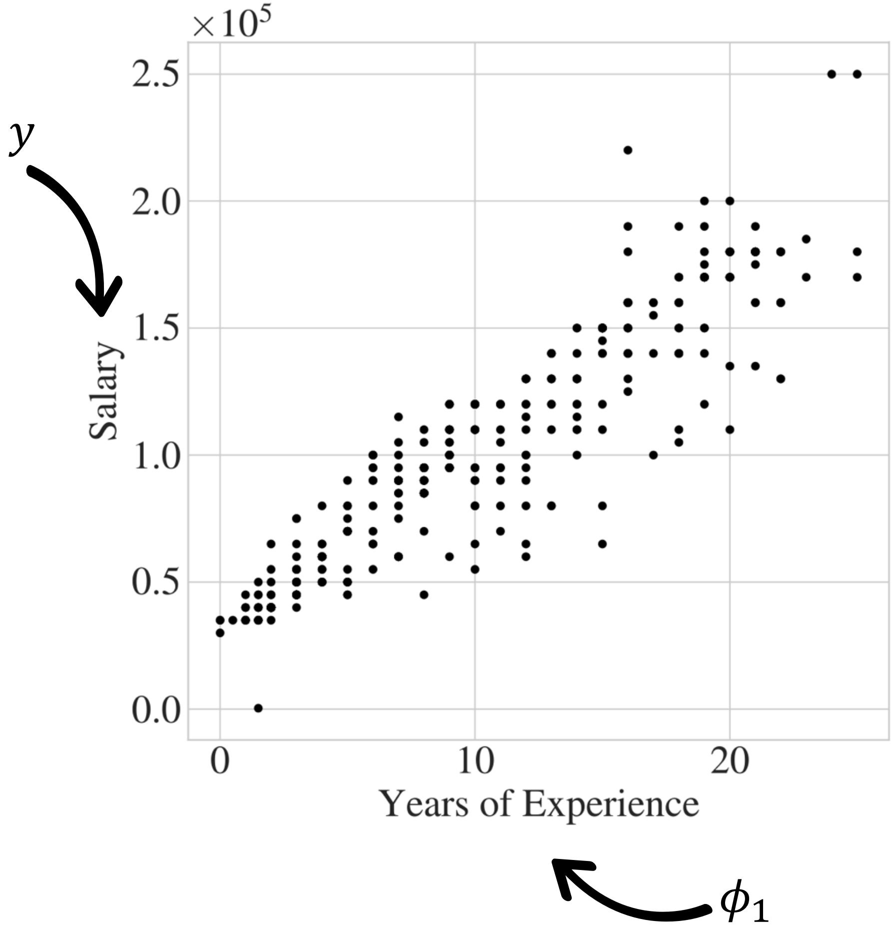
- Example: salary prediction based on the years of experience
- Data are $n = 373$ couples $\left(\phi^{(i)}, y^{(i)}\right) \Longrightarrow$ **Supervised learning**
- The target variable $y \in \mathbb{R}$ is continuous $\Longrightarrow$ **Regression**
- The linear model is

$$\hat{y}^{(i)} = \theta_0 + \theta_1 \phi_1^{(i)},$$

where $\phi_1^{(i)}$ is the nb. of years of experience of the i^{th} training example

- Now the model is fixed, how to find $\widehat{\boldsymbol{\theta}}$, the best possible parameters for our model and data?
- This is done using **empirical risk minimisation (ERM)**

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, R(\mathcal{D}, \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \ell\left(\hat{y}^{(i)}, y^{(i)}\right)$$



Linear regression

- A common **choice** of loss for regression is a **squared loss function**

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

- ***Here***, the optimisation problem can be solved analytically in closed-form. Rewriting the risk matricially, we have

$$R(\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{n_{\text{train}}} \|\boldsymbol{\Phi}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2$$

Feature matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1^{(1)} & \cdots & \phi_{d'}^{(1)} \\ \vdots & \ddots & \vdots \\ \phi_1^{(n_{\text{train}})} & \cdots & \phi_{d'}^{(n_{\text{train}})} \end{pmatrix} \in \mathbb{R}^{n_{\text{train}} \times d'}$$

Target vector

$$\boldsymbol{y} = \left[y^{(1)}, \dots, y^{(n_{\text{train}})}\right]^{\mathrm{T}} \in \mathbb{R}^{n_{\text{train}}}$$

The analytical minimisation of the squared loss in linear regression gives the unique solution (when $d' < n$) known as ***normal equations***

$$\hat{\boldsymbol{\theta}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$$