# Machine Learning

**Michaël Clément**

michael.clement@enseirb-matmeca.fr

2024–25

Bordeaux INP – ENSEIRB-MATMECA – 3A Intelligence Artificielle

**Acknowledgements**:

- Aurélie Bugeau (*Univ. Bordeaux*)
- Charles Deledalle (*Brain Corp*)
- Hugo Larochelle (*Google Brain*)
- Andrew Ng (*Stanford*)
- Baptiste Pesquet (*ENSC*)

# Objectives and organization

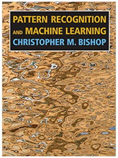
## Objectives

- Understand the key elements of **machine learning theory**
- Know some machine learning **algorithms** (not all)
- Be able to **implement** and **use** them in different **applications**

## Organization

- **24 hours** total (2h/3h sessions), mix of lectures and practices
- `IS318` page on *Thor* for information and class materials
- **Grading**: ∼20% assiduity, ∼80% practices

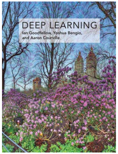
# References




***Pattern Recognition and Machine Learning***

Christopher M. Bishop

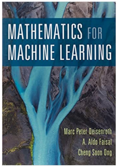
`https://www.microsoft.com/en-us/research/people/cmbishop/prml-book/`




***Deep Learning***

Ian Goodfellow, Yoshua Bengio, Aaron Courville

`https://www.deeplearningbook.org`




***Mathematics for Machine Learning***

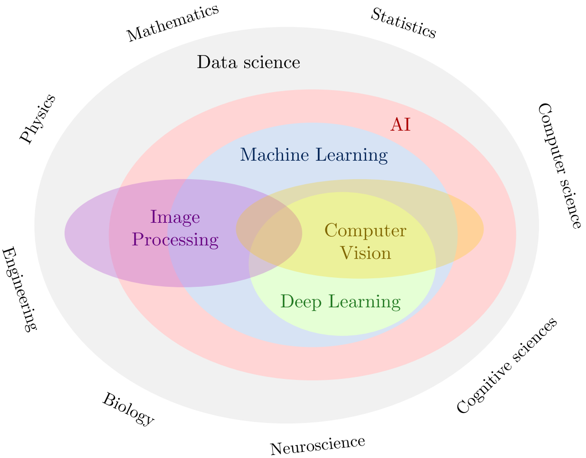
Marc Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong

`https://mml-book.com`

# Introduction

# What is machine learning?

## A multidisciplinary subject



# Definitions

> *"Learning is any process by which a system improves performance from experience."*
>
> — ***Herbert Simon***

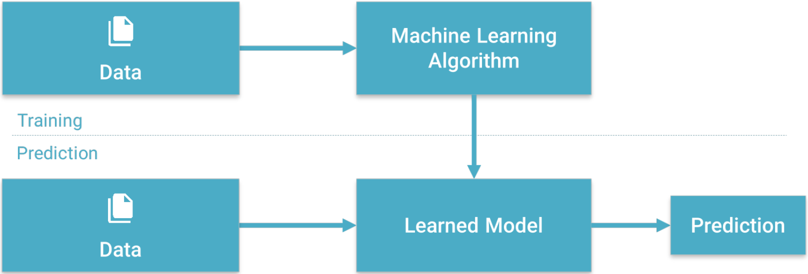

*Pavlov's dog*
*(Mark Stivers, 2003)*

> *"A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$."*
>
> — ***Tom Mitchell***

# Learn from data

Machine learning provides various techniques that can learn from and make predictions on data.

Most of them follow the same general structure:



# Main ingredients

**To learn from examples, we will need:**

❶ Training data (examples):

$$\mathcal{D}_{\text{train}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\},\ \boldsymbol{x}_i \in \mathbb{R}^d$$

❷ Model (machine or program):

$$\underbrace{\boldsymbol{x}}_{\text{input data}} \quad \rightarrow \quad \underbrace{f(\boldsymbol{x}; \boldsymbol{\theta})}_{\text{function / algorithm}} \quad \rightarrow \quad \underbrace{\hat{\boldsymbol{y}}}_{\text{prediction}}$$

❸ Loss, cost, objective function / energy:

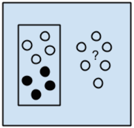
$$\arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\})$$

# Terminology

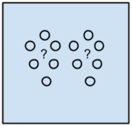
- **Sample (Observation or Data):** item to process (*e.g.*, classify).
*Example: an individual, a document, a picture, a sound, a video, etc.*
- **Features (Input)**: set of distinct traits that can be used to describe each sample in a quantitative manner. Represented as a multi-dimensional vector usually denoted by $\boldsymbol{x}$.
*Example: size, weight, frequency, color, etc.*
- **Training set:** set of data used to discover predictive relationships.
- **Testing set:** set of data used to assess the performance of a model.
- **Label or Prediction (Output):** the class or outcome assigned to a sample. The actual prediction is often denoted by $\hat{\boldsymbol{y}}$ and the ground truth value by $\boldsymbol{y}$.
*Example: cat/dog, temperature, price, etc.*

# Types of learning

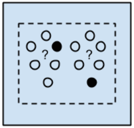
## Learning approaches



Supervised Learning Algorithms



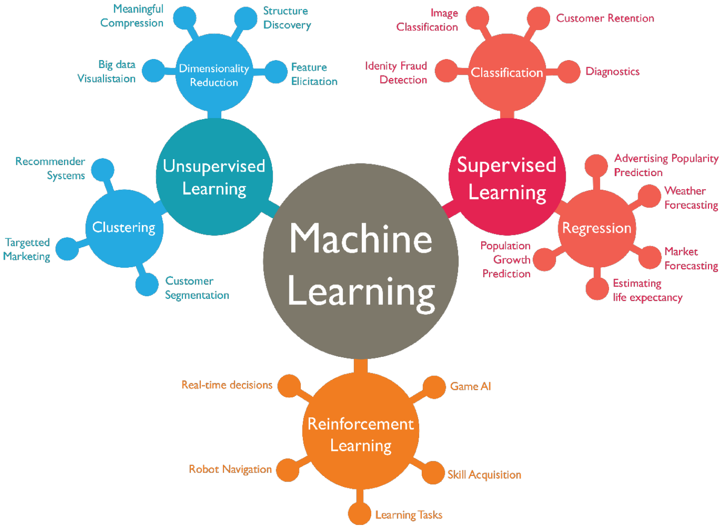
Unsupervised Learning Algorithms



Semi-supervised Learning Algorithms

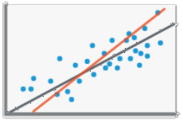
- **Supervised learning**: learn from labeled training data $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$
- **Unsupervised learning**: discover patterns in unlabeled training data $\mathcal{D}_{\text{train}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$
- **Semi-supervised learning**: learn with a small amount of labeled data and a large amount of unlabeled data.
- **Reinforcement learning**: Learning based on feedback or reward $\rightarrow$ `IS320` class.

# Different tasks

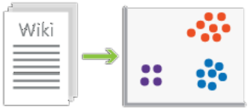


# Problem types

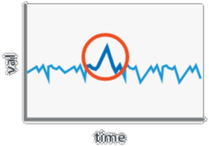

Classification
(supervised – predictive)



Regression
(supervised – predictive)




Clustering
(unsupervised – descriptive)




Anomaly Detection
(unsupervised– descriptive)

## No free lunch theorem

- Generic theorem in optimization theory and machine learning
- It states that, *averaged over all possible problems* and *without any prior knowlege*, **there is no objectively better learning algorithm**
- For example, a very fancy learning algorithm will, on average, perform as well as taking random decisions

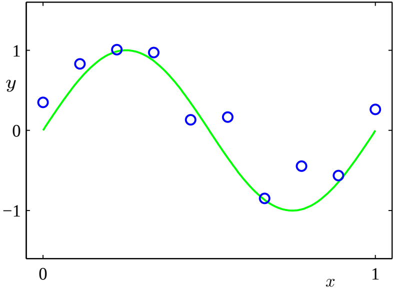

→ In practice, it means that we must design and apply solutions that are **biased towards the problem at hand**

# Schedule

1. Fundamentals and linear regression
2. Probabilistic modeling
3. Linear classifiers
4. Support vector machines
5. Decision trees
6. Combination of models

# Fundamentals and linear regression

# Example: Polynomial linear regression




**Linear regression problem in 1D**

- Input (feature): scalar $x \in \mathbb{R}$
- Output (target): scalar $y \in \mathbb{R}$

**Train set**: $N$ labeled examples (supervised setting)

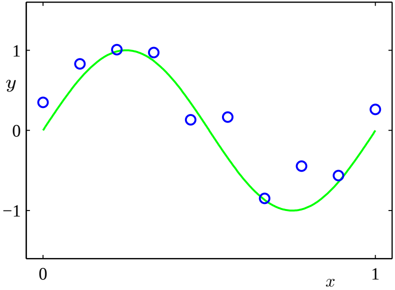
$$\mathcal{D}_{\text{train}} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$$

**Objective**: learn a function $f$ able to make prediction $\hat{y}$ for input $x$

$$\hat{y} = f(x)$$

# Model

**Polynomial linear regression**: we assume that good predictions follow a polynomial form




Our **model** $f$ can be defined as:

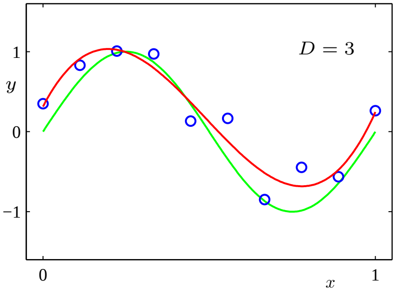
$$f(x; \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_D x^D = \sum_{d=0}^{D} w_d x^d$$

where $\boldsymbol{w} = [w_0, \ldots, w_D]^\mathsf{T} \in \mathbb{R}^{D+1}$ are the **parameters** of the model.

# Loss function

**Learning problem**:
**How to find a "good" $w$?**

$\rightarrow$ Find $\boldsymbol{w}^\star$ minimizing the difference between $(x_i, y_i)$ pairs in $\mathcal{D}_{\text{train}}$




**Mean squared error**: we are looking for $\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$ where

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{N} \sum_{i=1}^{N} \left(f(x_i; \boldsymbol{w}) - y_i\right)^2$$

This is loss function is **convex**, so there is a single global minimum.

# Optimization (1/2)

From the training set $\mathcal{D}_{\text{train}}$ we introduce:

- The data matrix $\boldsymbol{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^D \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^D \end{bmatrix}$
- The target vector $\boldsymbol{y} = \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix}^\mathsf{T}$

Then, the mean squared error can be expressed as:

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{N} \sum_{i=1}^{N} \left(f(x_i; \boldsymbol{w}) - y_i\right)^2 = \frac{1}{N} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2$$

To minimize, because the function is convex, we can set the gradient to $0$:

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) = 0 \iff \nabla_{\boldsymbol{w}} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2 = 0$$

# Optimization (2/2)

We have[1]:

$$\begin{aligned}\nabla_{\boldsymbol{w}} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2 = 0 \;&\Leftrightarrow\; \nabla_{\boldsymbol{w}} \left(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\right)^{\mathsf{T}} \left(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\right) = 0 \\ &\Leftrightarrow\; \nabla_{\boldsymbol{w}} \boldsymbol{w}^{\mathsf{T}} \boldsymbol{X}^{\mathsf{T}} \boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^{\mathsf{T}} X^{\mathsf{T}} \boldsymbol{y} + \boldsymbol{y}^{\mathsf{T}} \boldsymbol{y} = 0 \\ &\Leftrightarrow\; 2\boldsymbol{X}^{\mathsf{T}} \boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X}^{\mathsf{T}} \boldsymbol{y} = 0 \\ &\Leftrightarrow\; \boldsymbol{X}^{\mathsf{T}} \boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^{\mathsf{T}} \boldsymbol{y}\end{aligned}$$

Assuming $\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}$ is invertible[2], we find:

$$\boldsymbol{w} = (\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathsf{T}}\boldsymbol{y}$$

This solution is called the **normal equation**.

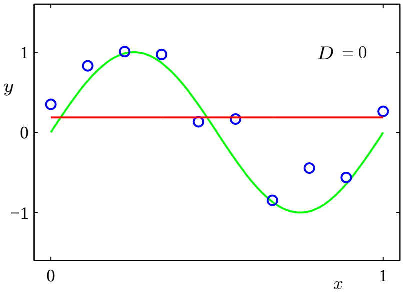
[1] For detailed derivations, see: `https://eli.thegreenplace.net/2014/derivation-of-the-normal-equation-for-linear-regression`

[2] This is the case if we have $N \geqslant D$ distinct $x_i$

# Underfitting and overfitting

**How to choose the value of $D$?**

$$f(x; \boldsymbol{w}) = \sum_{d=0}^{D} w_d x^d$$



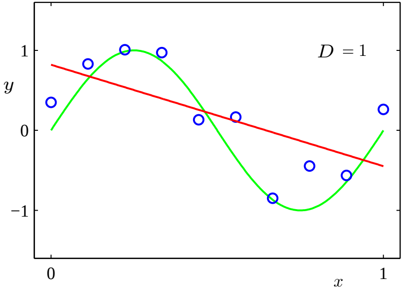

Too small: large error on the train set, too simple representation
$\rightarrow$ **Underfitting**

$D$ is called an **hyperparameter** of the model

# Underfitting and overfitting

**How to choose the value of $D$?**

$$f(x;\boldsymbol{w}) = \sum_{d=0}^{D} w_d x^d$$



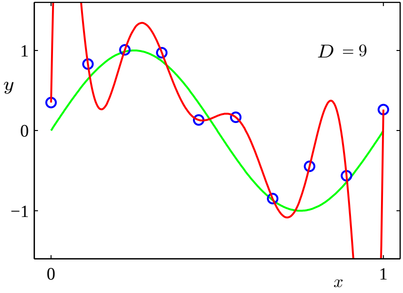

Too small: large error on the train set, too simple representation
$\rightarrow$ **Underfitting**

$D$ is called an **hyperparameter** of the model

# Underfitting and overfitting

**How to choose the value of $D$?**

$$f(x; \boldsymbol{w}) = \sum_{d=0}^{D} w_d x^d$$



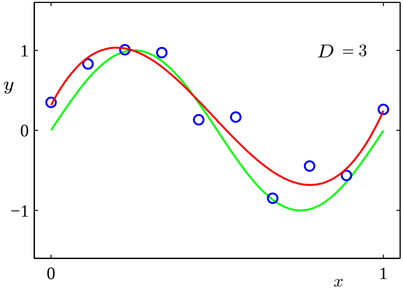

Too large: the model learns "by heart", too complex representation $\rightarrow$ **Overfitting**

$D$ is called an **hyperparameter** of the model

# Underfitting and overfitting

**How to choose the value of $D$?**

$$f(x; \boldsymbol{w}) = \sum_{d=0}^{D} w_d x^d$$



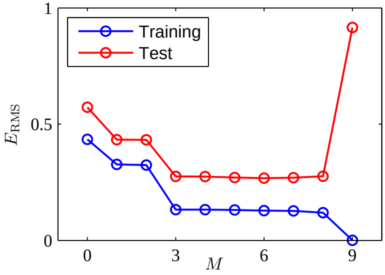

We want a good compromise that finds the general trend of the relationship, but without the noise, to **generalize** on new test data

$D$ is called an **hyperparameter** of the model

# Model capacity

**Model capacity**: ability of a model to learn the training set "by heart"

- Example: the larger $D$, the larger the capacity

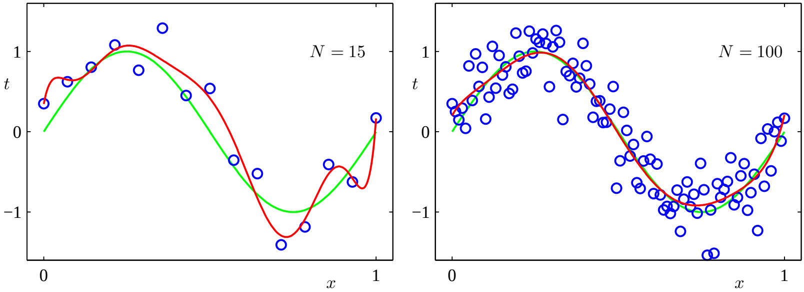



As the capacity increases, the difference between training error and test error is likely to increase

$$E_{RMS} = \sqrt{\mathcal{L}(\boldsymbol{w}^{\star})}$$

# Generalization

As the **amount of training data** increases, the better the trained model will **generalize**.



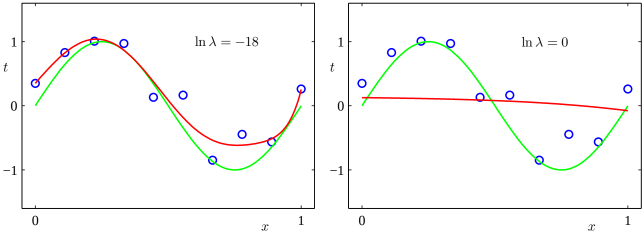
# Regularization

**Regularization:** add a penalty term to the mean squared error

$$\tilde{\mathcal{L}}(\boldsymbol{w}) = \frac{1}{N}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2 \; + \; \lambda\|\boldsymbol{w}\|^2$$

with $\|\boldsymbol{w}\|^2 = \boldsymbol{w}^\mathsf{T}\boldsymbol{w} = w_0^2 + w_1^2 + \ldots + w_N^2$ (*weight decay* penalty).

$\lambda$ is another **hyperparameter**, like $D$, which allows to control the capacity of the linear regression model.




With weight decay, solving for $\boldsymbol{w}$ gives: $\boldsymbol{w} = (\lambda\boldsymbol{I} + \boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\boldsymbol{y}$

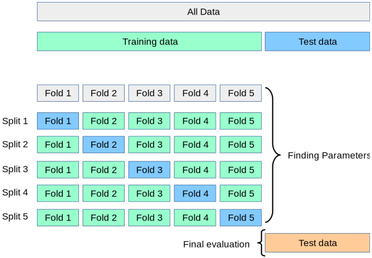
# Model selection

### How to choose the value of hyperparameters?

- Keep some training data aside in a **validation set** $\mathcal{D}_{\mathrm{valid}}$
- For example: split data into 80% for $\mathcal{D}_{\mathrm{train}}$ and 20% for $\mathcal{D}_{\mathrm{valid}}$
- For different values of hyperparameters, choose the model that gives the best performance on $\mathcal{D}_{\mathrm{valid}}$

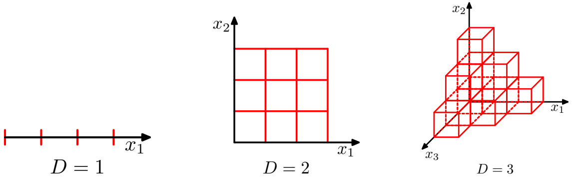
### Cross validation

- Repeat the procedure for different $\mathcal{D}_{\mathrm{train}}$ and $\mathcal{D}_{\mathrm{valid}}$ splits
- Average the results of each split to choose hyperparameters
- Useful when training data is limited



# Curse of dimensionality

- To generalize, we would like to have enough training examples to "fill" the underlying **feature space**
- However, this number grows **exponentially** with the dimension $D$
  - To have 10 samples in each dimension, it would require $10^D$ samples
- In machine learning, $D > 100$ or $D > 1000$ is very common




- But all is not lost! There is usually regularity/structure in the data
  - Dimensionality reduction
  - Feature engineering/learning