Deep learning
1. Introduction

A course @EDHEC
by Romain Tavenard (Prof. @Univ. Rennes 2)

Course details

- 18 hours (3 days)
- Instructor: Romain Tavenard romain.tavenard@univ-rennes2.fr
- Tools (*cf* course page for required Python packages):
  - Google Colab (create an account)
  - or Jupyter Notebooks running on your machine
- Evaluation
  - Group project (50%)
  - Final exam: A few questions + Lab session (50%)

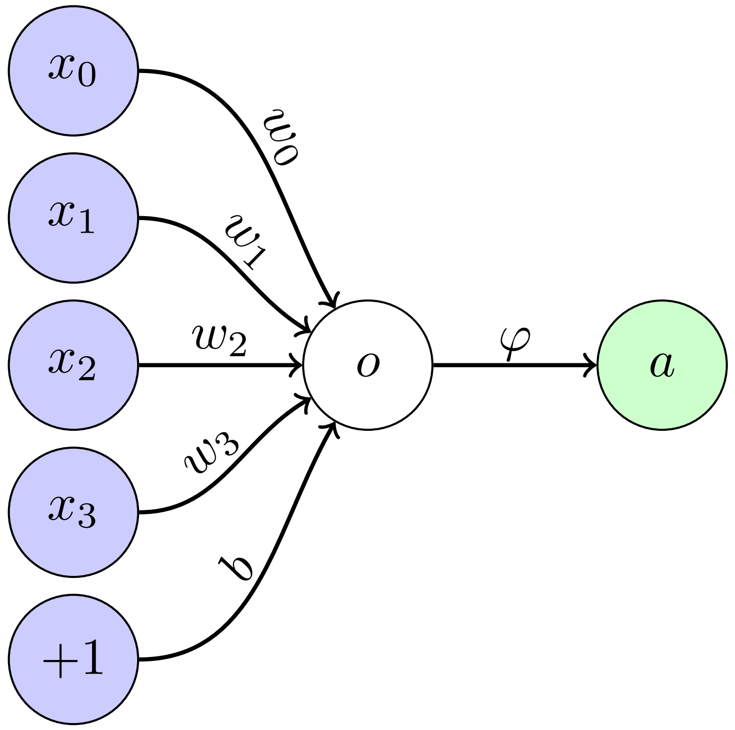
Pre-requisites

- Basics of Python coding
- A (tiny) bit of calculus
    - What's the derivative of a function?
    - Functions of several variables
- Machine Learning topics
    - Empirical risk optimization and its limitations
    - Model evaluation & selection (cross-validation)

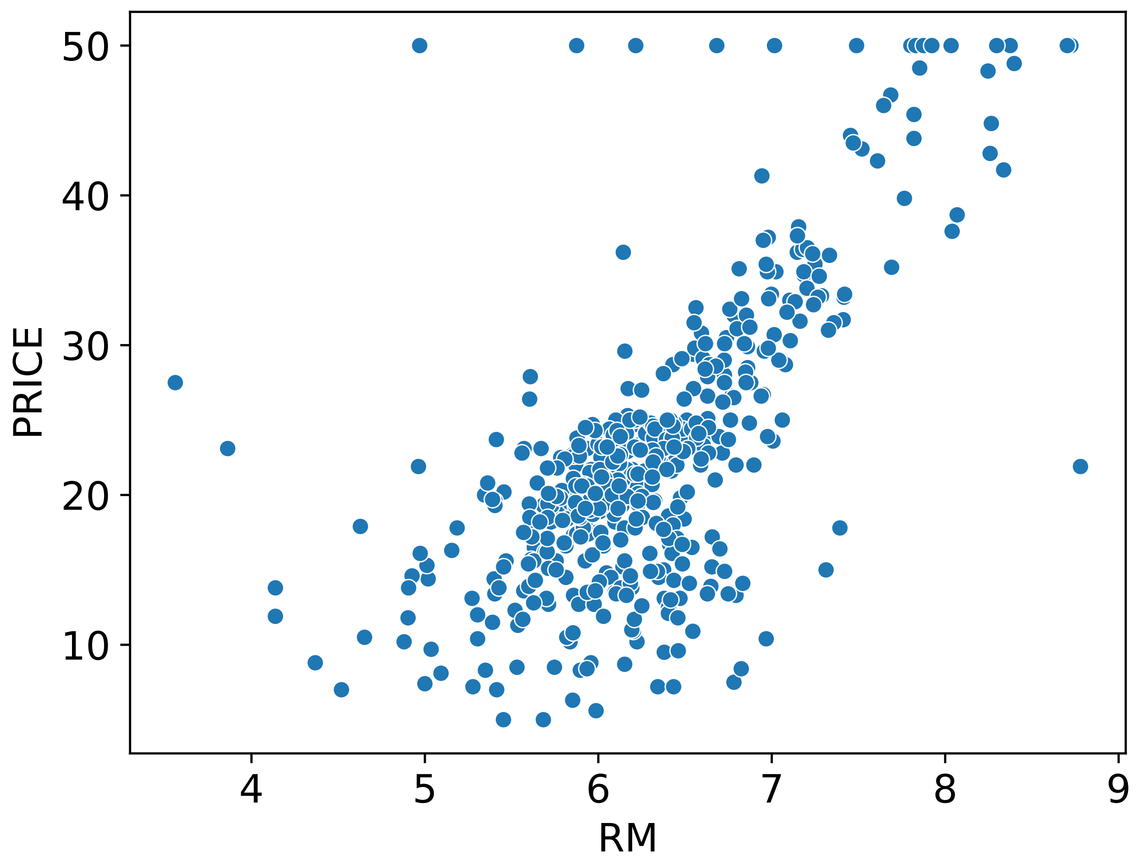
Our first model: the Perceptron

- Input: $x = \{x_0, \ldots, x_D\}$
- Output: a
- Parameters to be optimized: $\{w_0, \ldots, w_D, b\}$
- Activation function (chosen a priori): φ

$$a = \varphi\left(\underbrace{\sum_j w_j x_j + b}_{o}\right)$$



Motivating example



- Dataset: Boston housing prices
- Toy task: predict housing price (PRICE) based on average number of rooms per dwelling (RM)
- Chosen model: linear regression without intercept

$$\text{PREDICTED PRICE} = w_0 \times \text{RM}$$

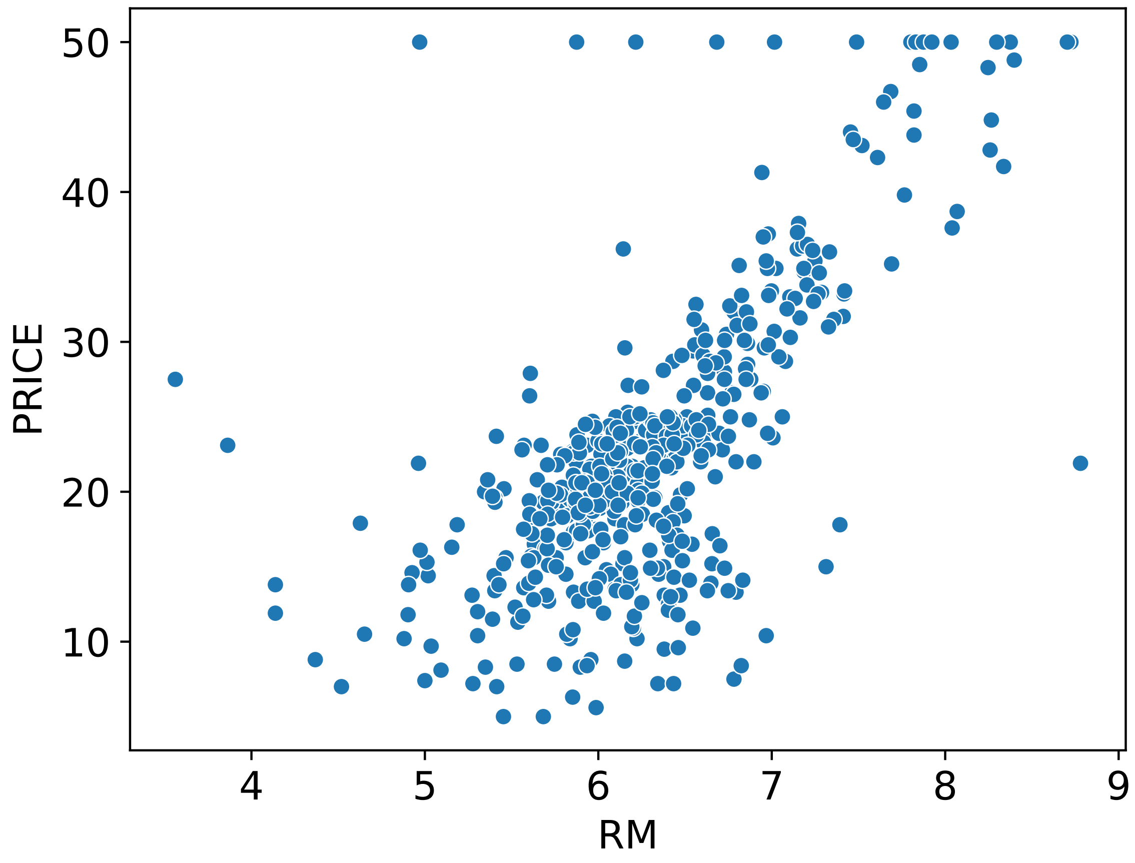
- Cost function (also called loss): Mean Squared Error

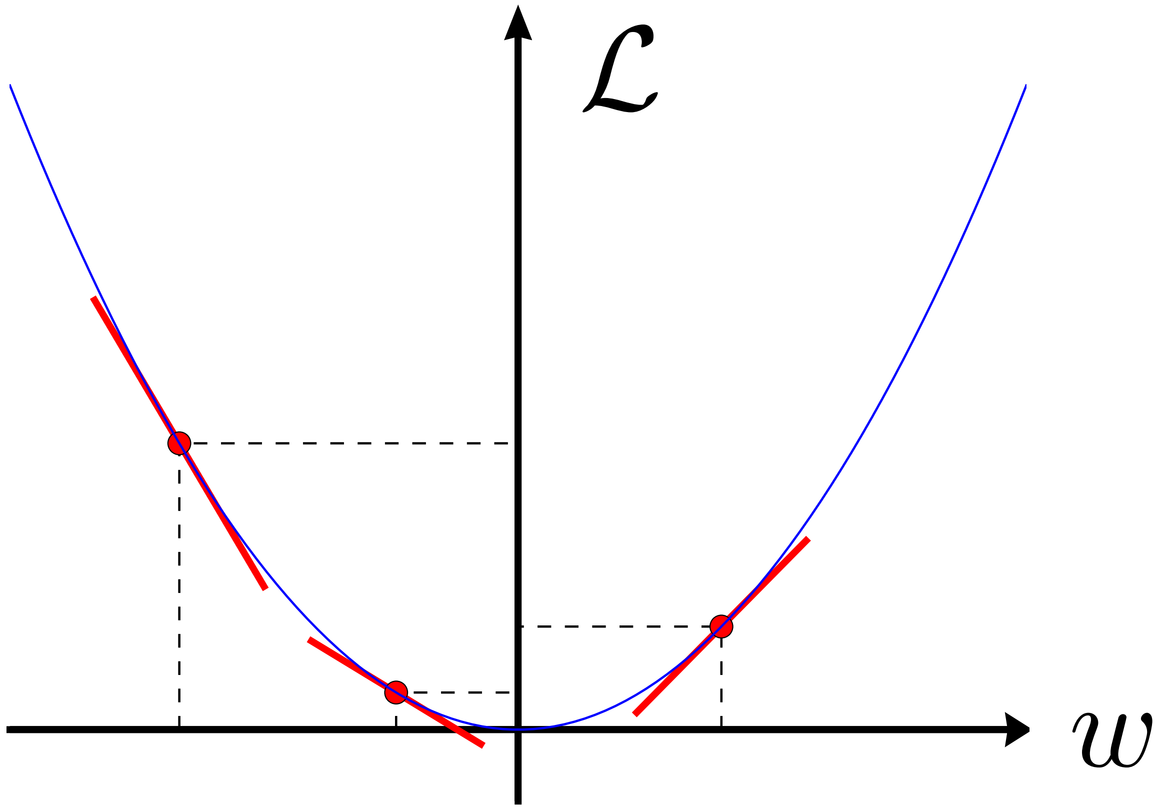
$$\sum_i (\underbrace{\text{PREDICTED PRICE}_{(i)}}_{w_0 \times \text{RM}_{(i)}} - \text{PRICE}_{(i)})^2$$

Motivating example: Optimization

- <u>Goal:</u> Find w_0 that minimizes our loss
- Testing many values at random is not tractable
- <u>Alternative strategy:</u> Gradient descent

$$w^{(t+1)} \leftarrow w^{(t)} - \rho \nabla_w \mathcal{L}(w^{(t)})$$





Course outline

- Model architectures
  - Multi-Layer Perceptrons (for tabular data)
  - Convolutional models (for images)
  - Recurrent models & attention-based models (for time series)
- Different losses for different learning tasks
- Optimization strategies (variants of gradient descent)