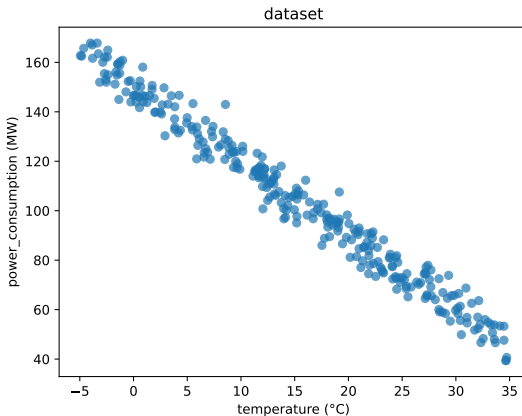


# Machine learning I, supervised learning: linear regression



# Content

Linear regression in one dimension

General linear regression : Ordinary least squares

Overfitting, hyperparameters and ridge regression

# Linear regression

Linear regression is one of the most elementary methods used in ML regression problems. It is useful for many applications, and is often a component of more complex methods.

We will use it to illustrate several classical aspects of ML that are also encountered when using other methods (kernels, trees, neural networks, etc.)

## Linear regression in one dimension

General linear regression : Ordinary least squares

Overfitting, hyperparameters and ridge regression

We want to predict the power that needs to be produced by a power plant in a city, as a function of the temperature only.

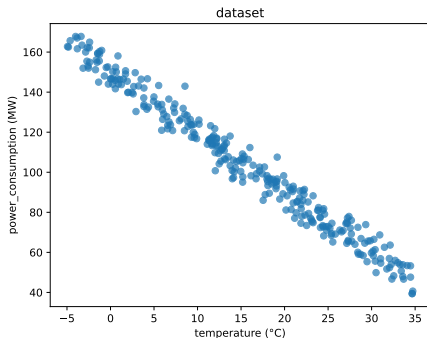


Figure – Dataset

## Exercise 1: Why are the samples not on a straight line?

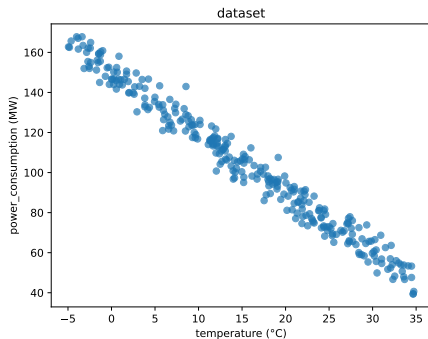


Figure – Dataset

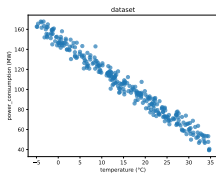


Figure – Dataset

The power consumption does not depend **only** on the temperature, but also on many other variables, that we do not have access to here :

- ▶ time in the day
- ▶ humidity, wind
- ▶ period of the year (holidays or not)
- ▶ other variables

However, our task is to predict the power consumption, only according to the temperature.

This is a **regression** problem, and we need to find a good **estimator** of the power consumed as a function of the temperature.



## Linear regression

Formalization :

- ▶ input space (temperature) :  $\mathcal{X} = \mathbb{R}$
- ▶ output space (power consumption) :  $\mathcal{Y} = \mathbb{R}$
- ▶ dataset :  $D = \{(x_1, y_1), \dots, (x_n, y_n), i \in [1, n]\}$ .

When doing linear regression, our estimator is of the form :

$$h(x) = \theta x + b \tag{1}$$

## Loss function

We will use the squared loss  $l$  :

$$l(y_1, y_2) = (y_1 - y_2)^2 \quad (2)$$

## Empirical risk

With the squared loss, we define the **empirical risk** as :

$$R_n(\theta, b) = \sum_{i=1}^n (\theta x_i + b - y_i)^2 \quad (3)$$

We want to find  $\theta$  and  $b$  such that  $R_n(\theta, b)$  has the **smallest possible value**. (sometimes it is normalized by a division by  $n$ , but this does not change the problem)

# Numpy

Numpy demo.

## Computing empirical risks

### Exercice 2 :

`cd code/linear_regression /1D_linear_regression/` and fix `utils.py` in order to compute the empirical risk.

Launch `main_random_params.py` in order to test several values for  $\theta$  and  $b$  by evaluating their empirical risk.

## Analytic solutions

For some problems, like this one, it is possible to explicitly compute the optimal solution.

For some advanced reasons (convexity and differentiability of  $R_n(\theta)$ ), the points optimizing the empirical risk are obtained by finding  $(\theta^*, b^*)$  such that the gradient cancels (more on that tomorrow).

$$\nabla_{(\theta, b)} R_n(\theta^*, b^*) = 0 \quad (4)$$

## Derivatives

$$\begin{aligned}\frac{\partial R_n}{\partial \theta}(\theta, b) &= \sum_{i=1}^n 2(\theta x_i + b - y_i)x_i \\ &= 2\left[\theta \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i - \sum_{i=1}^n x_i y_i\right]\end{aligned}\tag{5}$$

$$\begin{aligned}\frac{\partial R_n}{\partial b}(\theta, b) &= \sum_{i=1}^n 2(\theta x_i + b - y_i) \\ &= 2\left[\theta \sum_{i=1}^n x_i + nb - \sum_{i=1}^n y_i\right]\end{aligned}\tag{6}$$

Hence we have a system of 2 equations with 2 unknowns (dropping the  $\theta^*$  notation)

$$\theta \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i - \sum_{i=1}^n x_i y_i = 0 \quad (7)$$

$$\theta \sum_{i=1}^n x_i + nb - \sum_{i=1}^n y_i = 0 \quad (8)$$



Which means

$$b = \frac{1}{n} \left( \sum_{i=1}^n y_i - \theta \sum_{i=1}^n x_i \right) \quad (9)$$

$$\theta \sum_{i=1}^n x_i^2 + \frac{1}{n} \left( \sum_{i=1}^n y_i - \theta \sum_{i=1}^n x_i \right) \sum_{i=1}^n x_i - \sum_{i=1}^n x_i y_i = 0 \quad (10)$$

Finally :

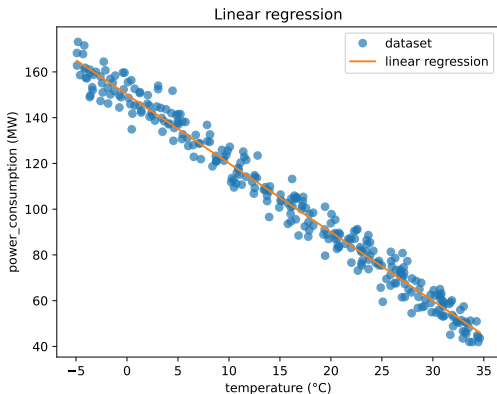
$$\theta \left( \sum_{i=1}^n x_i^2 - \frac{1}{n} \left( \sum_{i=1}^n x_i \right)^2 \right) + \frac{1}{n} \sum_{i=1}^n x_i \sum_{i=1}^n y_i - \sum_{i=1}^n x_i y_i = 0 \quad (11)$$

or

$$\theta^* = \frac{\sum_{i=1}^n x_i y_i - \frac{1}{n} \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{\sum_{i=1}^n x_i^2 - \frac{1}{n} \left[ \sum_{i=1}^n x_i \right]^2} \quad (12)$$

### Exercise 3 :

Fix `main_optimal_params.py` in order to plot the linear regression found with the analytic solution on the same plot as the raw dataset.



Linear regression in one dimension

General linear regression : Ordinary least squares

Overfitting, hyperparameters and ridge regression

## Generalization

Linear regression also works in higher dimensions, when the inputs are multidimensional. For instance in dimension 3,  $x = (x_1, x_2, x_3)$  and :

$$h(x) = \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + b \quad (13)$$

The parameter is now  $(\theta, b) = (\theta_1, \theta_2, \theta_3, b)$ .

Example :  $x$  contains the age, the profession, and the gender.

Now, the input data are stored in a matrix  $X$  with  $n$  lines and  $d$  columns.

The output data are stored in a vector  $y$  with  $n$  lines.

The empirical risk writes (adding back the normalization) :

$$R_n(\theta, b) = \frac{1}{n} \|X\theta - y + b\|^2 \quad (14)$$

(more on the notion of norm later in the course)

## OLS estimator

It is possible to show, with some maths notions, that the  $\theta$  that minimizes the empirical risk is :

$$\hat{\theta} = (X^T X)^{-1} X^T y \quad (15)$$

$T$  is the transposition.

# Scikit

We can use scikit-learn in order to obtain the OLS estimator directly.

<https://scikit-learn.org>



## Scikit in 1D

**main\_scikit.py** computes the OLS for the previous power consumption example.

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LinearRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)

Linear regression in one dimension

General linear regression : Ordinary least squares

Overfitting, hyperparameters and ridge regression

## Overfitting

```
cd ../dD_linear_regression/
```

When  $d$  is of the same order of magnitude or larger than the number of samples  $n$ , it is possible to have a low **train error** and a high **test error**. This is known as **overfitting**.

- ▶ Example with **OLS\_scikit.py**
- ▶ You can experiment with the data used by changing **generate\_data.py**

## Ridge regression

Ridge regression is a variation of OLS. For some advanced reasons, it can reduce overfitting.

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.Ridge.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html)

Example with **Ridge\_scikit.py**.

## Ridge regression

Ridge regression is a variation of OLS. For some advanced reasons, it can reduce overfitting.

`https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.ridge.html`

Example with **`ridge_scikit.py`**.

Importantly, you can see that `Ridge()` has some parameters, called **hyperparameters**. Almost all machine learning algorithms have hyperparameters.

## Choice of the hyperparameters

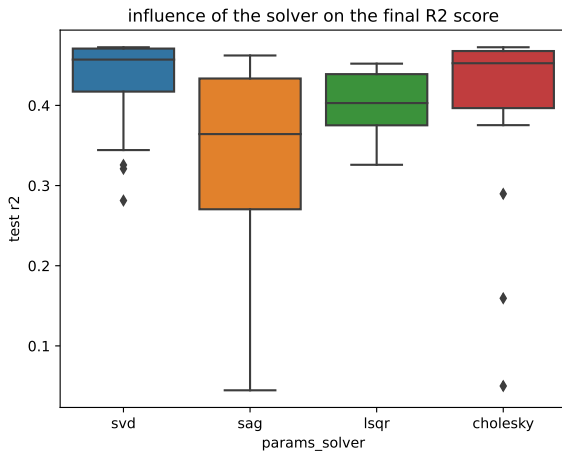
- ▶ The choice of the hyperparameters is very important. Most of the time, it is guided by experimentation and / or theoretical results.
- ▶ Some methods and libraries are helpful to look for good hyperparameters, such as **optuna**  
<https://optuna.org/>
- ▶ other classical methods : gridsearch, random search.

## Using optuna to tune Ridge regression

### Exercise 4 :

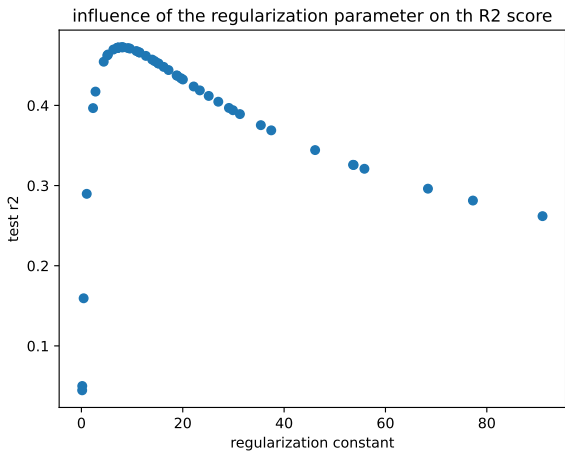
Use `optuna_ridge_scikit.py` in order to choose some good hyperparameters (alpha, solver) for ridge. You will need to study the optuna API and to edit the `objective()` function.

## Analysis of the hyperparameters





## Analysis of the hyperparameters



## Optuna dashboard

<https://github.com/optuna/optuna-dashboard>



## Multi-objective optimization

Optuna can be used to optimize several objectives, e.g. : optimize the score and minimize the computation time (both depend on the hyperparameters).

Notion of Pareto front.