PTML 1: 03/03/2022

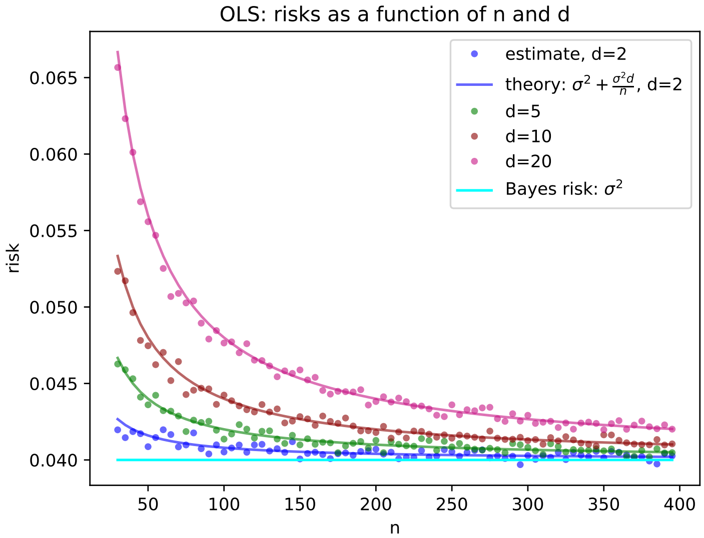

TABLE DES MATIÈRES

1 SOLUTIONS TO EXERCICES 1

See class and FTML/Exercises/Solutions1.pdf.

2 INTRODUCTION TO NUMPY, JUPYTER, PANDAS, MATPLOTLIB

— Verify that the library listed in **PTML/requirements.txt** were correctly imported, by importing them from a python interpreter or by using **pip list.**

— From **TP1/**, run **jupyter-lab** and follow the instruction in **TP_1_numpy_demo.ipynb**.

— Follow the instructions in **TP_1_pandas_demo.ipynb**.

— Run **TP_1_matplotlib_demo.py**.

Time complexity of operations in python :
https://wiki.python.org/moin/TimeComplexity

3 ORDINARY LEAST SQUARES

3.1 Introduction

A **linear model**, such as the OLS, can often be interpreted as predicting an output value (dependent variable) from combining the contributions from the d **features** of the input data (independent variables), in a linear way. This can be useful for classification as well as regression.

For instance, if I want to predict the amount of money that I will spend when buying some clothes, I can use a linear model. If θ contains the price of each type of clothe, and x the number of each type of clothe that I buy, then I have to spend $x^T\theta$. If there exists 4 types of clothes with a price θ_i :

— socks : $\theta_1 = 2$

— T-shirts : $\theta_2 = 25$

— pants : $\theta_3 = 50$

— hats = $\theta_4 = 20$

If I want to buy 10 socks, 2 T-shirts, 1 pants and 1 hat, then $x = (10, 2, 1, 1)$ and I spend

$$\begin{aligned} x^T\theta &= 10 \times 2 + 2 \times 25 + 1 \times 50 + 1 \times 20 \\ &= 140 \end{aligned} \tag{1}$$

Obviously, not all phenomena can be approximated well in a linear way. However, linear regression is a foundation for more advanced modelisation that we will stufy in future classes (feature maps, kernel methods, neural networks, etc).

3.2 Formalization

In this part, we will implement a **linear least-squares, regression**, as presented at the end of lecture 2. The Ordinary least-squares is an important supervised learning problem. In a least squares problem, the loss l writes :

$$l(y, y') = ||y - y'||^2$$

In the Ordinary Least Squares (OLS) setup, $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$, and the estimator is a **linear function** parametrized by $\theta \in \mathbb{R}^d$.

$$F = \{x \mapsto \theta^T x, \theta \in \mathbb{R}^d\}$$

The dataset is stored in the **design matrix** $X \in \mathbb{R}^{n \times d}$.

$$X = \begin{pmatrix} x_1^T \\ \dots \\ x_i^T \\ \dots \\ x_n^T \end{pmatrix} = \begin{pmatrix} x_{11}, \dots, x_{1j}, \dots x_{1d} \\ \dots \\ x_{i1}, \dots, x_{ij}, \dots x_{id} \\ \dots \\ x_{n1}, \dots, x_{nj}, \dots x_{nd} \end{pmatrix}$$

The vector of predictions of the estimator writes $Y = X\theta$. Hence the empirical risk writes

$$R_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)^2$$
$$= \frac{1}{n}||Y - X\theta||_2^2$$

In this TP we want to study the optimal solution of the OLS problem, called the OLS estimator. We assume that X is **injective.** Necessary, $d \leqslant n$.

Proposition. *Closed form solution*
*We X is injective, there exists a unique minimiser of $R_n(\theta)$, called the **OLS estimator**, given by*

$$\hat{\theta} = (X^T X)^{-1} X^T Y \tag{2}$$

The proof can be found in **FTML.pdf** at the OLS section.

3.3 Statistical properties of the OLS estimator

$\hat{\theta}$ depends on the design matrix X and on the output vector Y, it is thus a **random variable.** We are interested in the following questions :

— **What is the excess risk of the OLS estimator ?**

— **what is the stability of the OLS estimator ?**, which means **does a small perturbation on the dataset lead to a large perturbation of the OLS estimator ?** If yes, this means that the OLS estimator is unstable.

To answer these questions, we need a probabilistic framework. We will use the **linear model**, with **fixed design**. This means that we assume that there exists a vector $\theta^* \in \mathbb{R}^d$, such that $\forall i \in \{1, \dots, \}$,

$$y_i = x_i^T \theta^* + \epsilon_i \tag{3}$$

where for all $i \in \{1, \dots, n\}$, ϵ_i are independent, with expectation $E[\epsilon_i] = 0$ and variance $E[\epsilon_i^2] = 0$. The ϵ_i represent a variability in the output, that is due to **noise**, or to the presence of unobserved variables. Put together in a vector ϵ, this allows to write

$$Y = X^T \theta^* + \epsilon \tag{4}$$

1) In this setup, what is the Bayes predictor and the Bayes risk ?

2) What is the expectation of $\hat{\theta}$?

We admit the following properties, that we will show during the lectures :

Proposition. *Distance to optimal parameter, excess risk of OLS*
Still with the same hypotheses (linear model, fixed design)

$$E[R_X(\hat{\theta})] - R_X(\theta^*) = \frac{\sigma^2 d}{n}$$

and, if $\Sigma = X^T X \in \mathbb{R}^{d \times d}$,

$$Var(||\hat{\theta} - \theta^*||^2) = \frac{d\sigma^2}{n} Tr(\Sigma^{-1})$$

We note that both these quantities increase linearly with the dimension.

3.4 Simulations

We would like to experimentally observe the behavior of the OLS estimator, and the variability of $\hat{\theta}$, when the dataset is changed.

3.4.1 *Implementation of OLS*

In the file **TP_1_ols.py** :

— fix **generate_output_data()** in order to generate a dataset according to the linear model, fixed design setting.

— fix **OLS_estimator.py** in order compute the OLS estimator from X and Y.

— fix **error()** in order to compute the mean squared error of a predictor θ on data X with label Y.

Modify **ols_risk.py**, for example by introducing a loop, so that :

— several output data are generated

— and OLS estimator is computed each time

— the test errors and train errors are stored and plotted at the end of the simulation, like for example in 1, 2 , 3 . Assess the influence of n and d by trying different values for each. You can average the test errors to have an estimation of the risk (generalization error) of OLS, and plot the Bayes risk on the graph.

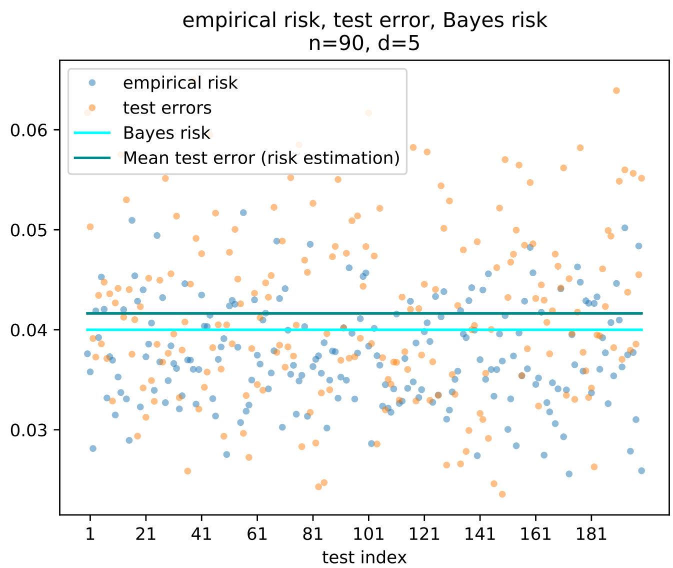

FIGURE 1 – test and train errors

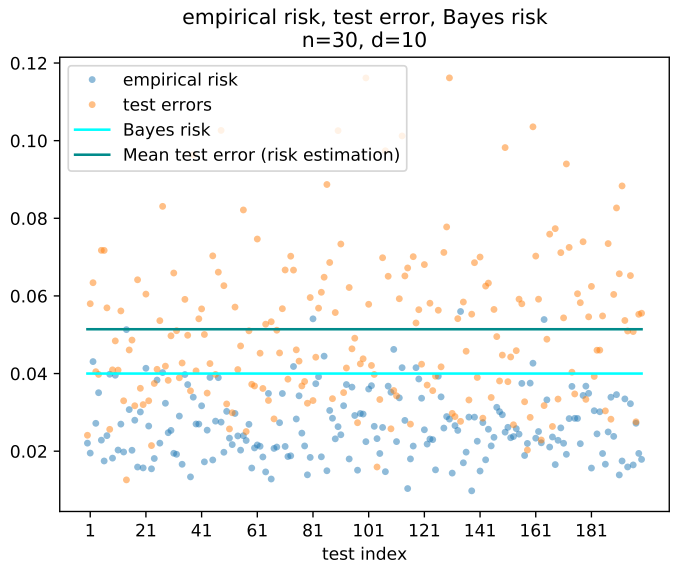

FIGURE 2 – test and train errors, higher $\frac{d}{n}$

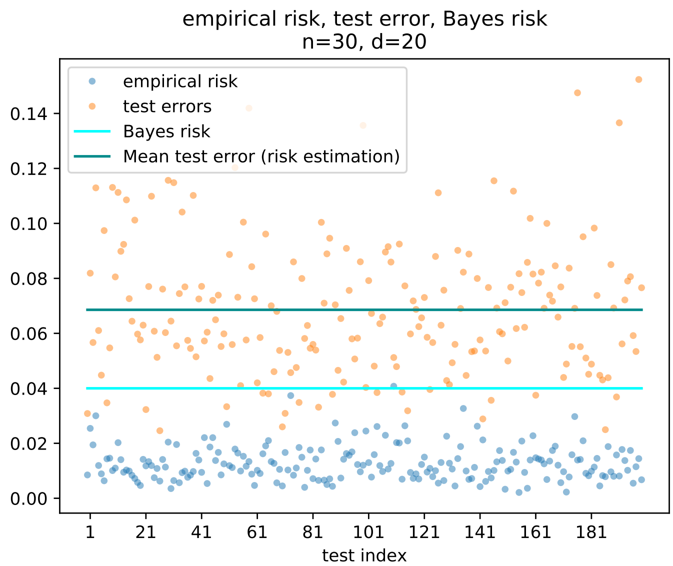

FIGURE 3 – test and train errors, high $\frac{d}{n}$ (overfitting)

3.4.2 *Stability of the OLS estimator*

Plot the relative distance between the OLS estimator $\hat{\theta}$ and the optimal estimator θ^*.

$$\frac{||\hat{\theta} - \theta^*||}{||\theta^*||} \tag{5}$$

and compute the relative distance between the average $< \hat{\theta} >$ and θ^*. This should be small if you run a sufficient number of tests, as $E[\hat{\theta}] = \theta^*$.
See 4.

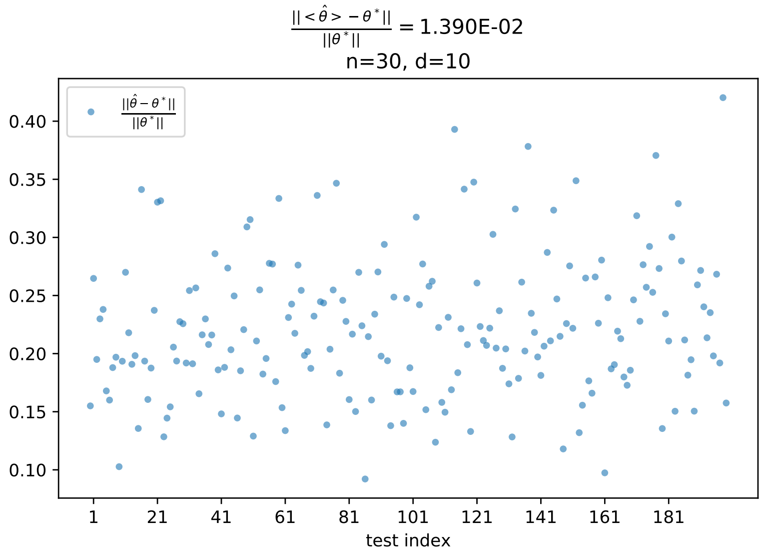

FIGURE 4 – $\hat{\theta}$ is a random variable

What happens is you replace the randomly generated design matrix X by the matrix stored in **"data/design_matrix.npy"**? Why? See 5.

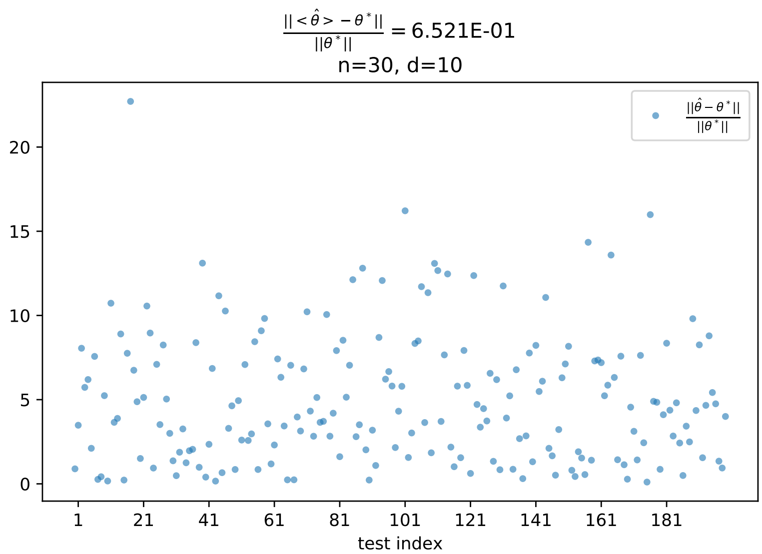

FIGURE 5 – When X is not injective, $\hat{\theta}$ varies more.

3.4.3 *Influence of* d *and* n

Modify the script in order to observe the dependence of the risk as a function of d and n, as stated in 3.3, in order to observe the same behavior as in 6.

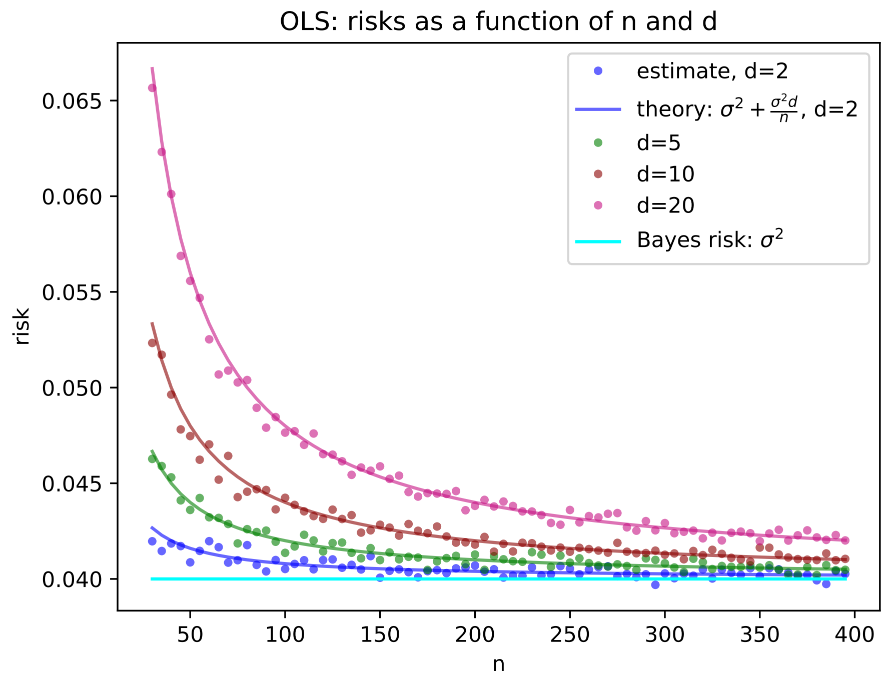

FIGURE 6 – Dependence of the risk (generalization error)