FTML practical session 2: 2023/03/09

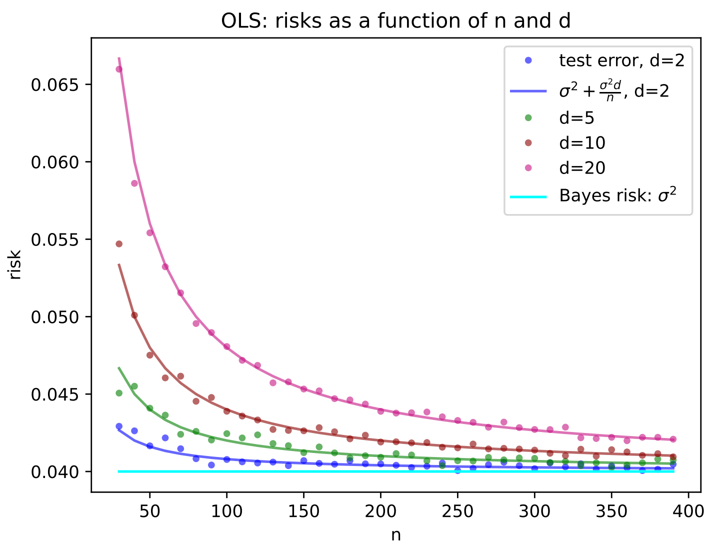

TABLE DES MATIÈRES

INTRODUCTION

The goal of this session is to keep exploring the concepts linked to risks : Bayes risk, empirical risk, overfitting. This time we explore them in a linear regression context.

There are some template files that you can use in the repo :

— **main_ols.py**

— **main_ridge.py**

— **utils_plots.py**

— **utils_algo.py**

— **constants.py**

1 LINEAR REGRESSION

A **linear model**, such as the Ordinary least squares (OLS), can 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.

1.1 A simple example

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 $d = 4$ 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 $x^T = (10, 2, 1, 1)$ 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 study in future classes (feature maps, kernel methods, neural networks, etc).

As we will study the influence of the dimensions of the problem n (number of samples) and d (number of features of each sample), we will work with abstract datasets, instead of a given "physical dataset". However, you can always refer to the previous example in order to interpret linear models that we will use.

1.2 Formalization

Let us abstract the notations a little bit. In the OLS setting,

— $\mathcal{X} = \mathbb{R}^d$ (input space)

— $\mathcal{Y} = \mathbb{R}$ (output space)

The estimator is a **linear mapping** parametrized by $\theta \in \mathbb{R}^d$. The prediction associated with $x \in \mathbb{R}^d$ is $f(x) = \theta^T x = x^T\theta$. If we use the squared-loss, the discrepancy between two real numbers y and y' writes $l(y, y') = (y - y')^2$. Finally, the input dataset is stored in the **design matrix** $X \in \mathbb{R}^{n \times d}$.

$$X = \begin{pmatrix} x_1^T \\ \dots \\ x_i^T \\ \dots \\ x_n^T \end{pmatrix} = \begin{pmatrix} x_{11}, \dots, x_{1j}, \dots x_{1d} \\ \dots \\ x_{i1}, \dots, x_{ij}, \dots x_{id} \\ \dots \\ x_{n1}, \dots, x_{nj}, \dots x_{nd} \end{pmatrix} \tag{2}$$

and the output labels are stored in a vector

$$y = \begin{pmatrix} y_1 \\ \dots \\ y_i \\ \dots \\ y_n \end{pmatrix} \in \mathbb{R}^n \tag{3}$$

1.3 Empirical risk

The **empirical risk** of an estimator θ (when we talk about an estimator, it is here equivalent to refer to θ directly or to the mapping defined by θ)

$$R_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2 \tag{4}$$

How can we write the empirical risk using only matrices, vectors and an euclidean norm ?

By contract, the **generalization error** of an estimator, , does not depend on the dataset. It is a fixed number, defined by (in this context of linear regression, squared loss)

$$R(\theta) = E[(y - \theta^T x)^2] \tag{5}$$

where the expected value is taken over the law of x and y, the random variables representing the input and output respectively. This law is unknown, in general.

2 ORDINARY LEAST SQUARES : EMPIRICAL RISK AND OVERFITTING

2.1 OLS estimator

The **OLS estimator**, noted $\hat{\theta}$, is the value of θ that minimizes $R_n(\theta)$. It is thus the solution to the problem of **empirical risk minimization**, a standard approach in supervised learning. We admit the following proposition :

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

$$\hat{\theta} = (X^T X)^{-1} X^T y \tag{6}$$

With the results of the exercises made today, you can try to prove this result.

2.2 Excess risk the OLS estimator

Given an estimator, we are interested in its **excess risk**. It is the difference between its generalization error and the bayes risk. By definition, this difference is non-negative. We will evaluate the excess risk of the OLS estimator as a function of n and d, for a given statistical setting, in order to observe the results of figure 1.

2.2.1 *Statistical setting*

In order to compute these quantities, it is necessary to make statistical assumptions. We will use the **linear model**, with **fixed design**, a classical framework to analyze OLS. In this setting, we assume that X is given and fixed, and that there exists a vector $\theta^* \in \mathbb{R}^d$, such that $\forall i \in \{1, \dots, \}$,

$$y_i = x_i^T \theta^* + \epsilon_i \tag{7}$$

where for all $i \in \{1, \dots, n\}$, ϵ_i are independent, with expectation $E[\epsilon_i] = 0$ and variance $E[\epsilon_i^2] = \sigma^2$. 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 $\epsilon \in \mathbb{R}^n$, this allows to write

$$y = X^T \theta^* + \epsilon \tag{8}$$

Considering last week's practical session observations, what do we expect to be the Bayes estimator ?

We admit (but you can also prove it) that the corresponding Bayes risk is σ^2.

For a given n and d (with $d \leqslant n$), run a simulation that reproduces the fixed design, linear model statistical setting and verify that the the estimator that you have chosen achieves the Bayes risk. Verify that different estimators have worse (larger) risks. Hence you need to generate first a X and a θ^*.

You will again need to approximate the real risk by the empirical risk, a number of times that is sufficient to observe the convergence of the law of large numbers.

2.3 Impact of the dimensions on the OLS overfitting

We now know that in this statistical setting, the Bayes risk is σ^*. In practical applications, we of course do not have access to θ^* and we must find a relevant θ, based on the data only. We must resort to empirical risk minimization.

We want to compare the risk of the OLS estimator to the Bayes risk, to know how good the OLS estimator compares to the Bayes estimator. Again, remember that in practical applications, we will not have access to the Bayes estimator or the Bayes risk.

Run a simulation that generates an output vector y, computes the corresponding OLS estimator $\hat{\theta}$ and an estimation of its generalization error by sampling more outputs and by computing the empirical risk of $\hat{\theta}$ on these new outputs.

In order to generate the X matrices in various dimensions, you can use uniformly distributed entries. This should ensure that X is injective.

Run the same simulation for various values of n and d in order to reproduce the results of figure 1.

We will prove during the lectures the different results mentioned in this exercice.

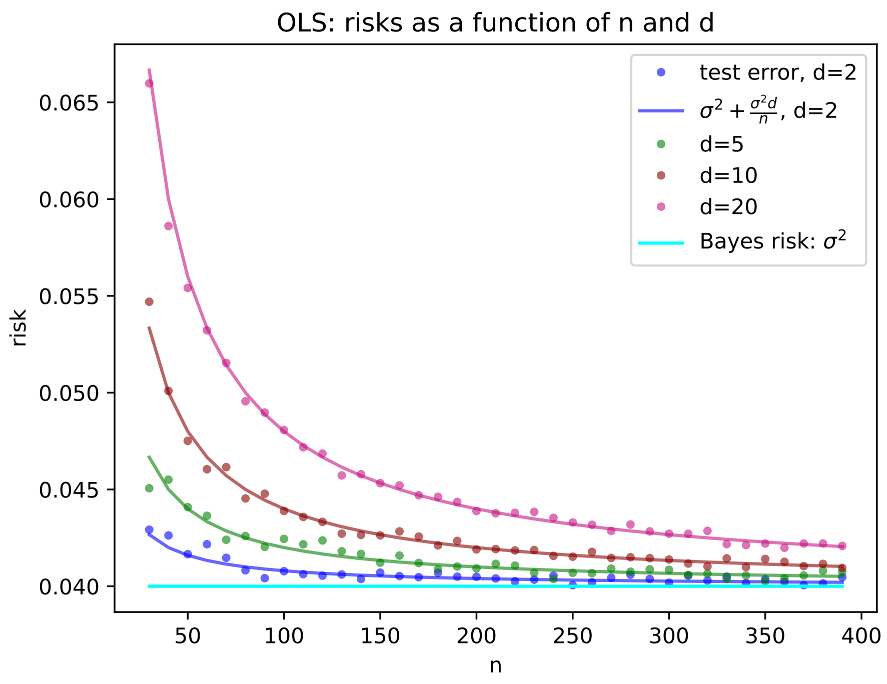

FIGURE 1 – Dependence of the risk (generalization error)

If $d > n$, the $\sigma^2 d/n$ law is not respected !

Why can we still output an OLS estimator when $d > n$?

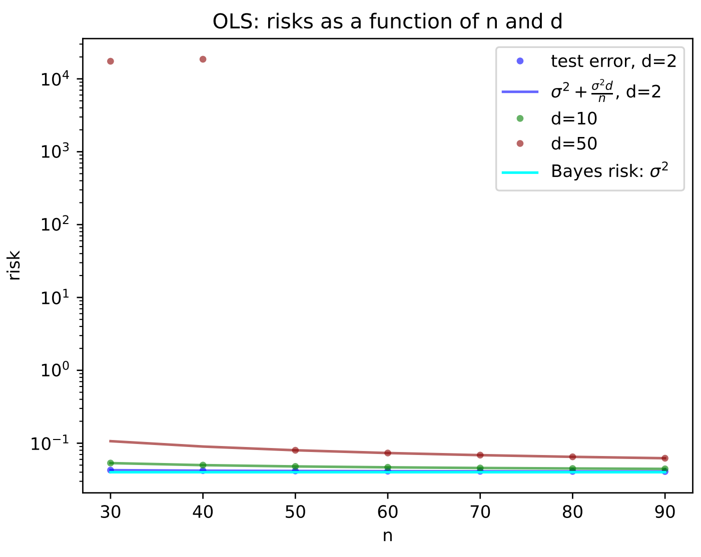

FIGURE 2 – Test errors with cases where $d > n$.

Optional : monitor the stability of the OLS estimator as a function of n and d, for instance by computing the sample standard deviation of the random variable $||\hat{\theta} - \theta^*||/||\theta^*||$ as a function of n and d. When d is closer to n, $\hat{\theta}$ can become very unstable (a small variation of the noise might lead to a large variation of $\hat{\theta}$). This behavior depends on the conditioning of $X^T X$, as we will see later.)

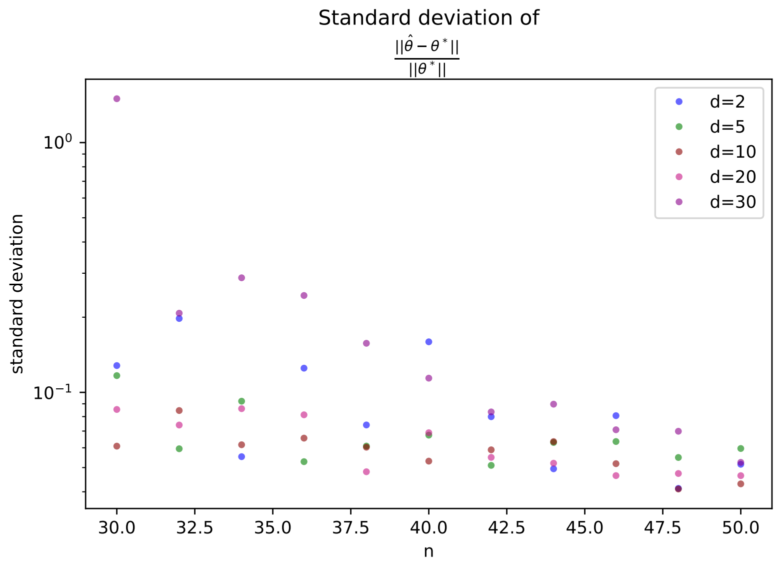

FIGURE 3 – Instability of the OLS estimator when d gets close to n.

3 RIDGE REGRESSION AND HYPERPARAMETER TUNING

The goal of this exercice is to experimentally observe the benefit of using Ridge regression instead of OLS. As we have just experimentally witnessed, the excess risk of the OLS estimator in the linear model, fixed design is $\frac{\sigma^2 d}{n}$.

3.1 Ridge regression estimator

Ridge regression replaces the minimization of the empirical risk by a slightly different objective function.

Definition 1. Ridge regression estimator
It is defined as

$$\hat{\theta}_\lambda = \arg\min_{\theta \in \mathbb{R}^d} \left(\frac{1}{n} ||Y - X\theta||_2^2 + \lambda ||\theta||_2^2 \right) \tag{9}$$

As you can see, there is a parameter λ, that controls the magnitude of the **regularization**. The goal is to penalize estimators with a large amplitude. We admit the following proposition.

Proposition. *The Ridge regression estimator is unique even if $X^T X$ is not inversible and is given by*

$$\hat{\theta}_\lambda = \frac{1}{n}\left(\frac{1}{n}X^T X + \lambda I_d\right)^{-1} X^T y$$

A natural question is then : is the excess risk of the Ridge regressor better (smaller) than that of the OLS estimator, for a good choice of λ ? If yes, how can we tune a good value for λ ?
The answer will be positive for some values of n, d, and σ.

3.2 Excess risk of the ridge regression estimator

Implement a simulation in order to observe situations where the Ridge regression estimator has a better (smaller) generalization error than the OLS estimator for good values of λ. This might happen if one or several of the following conditions are satisfied :

— σ is large

— $d \leqslant n$ or d is close to n

— X has a low rank

Hence, you can tune your simulation in order to satisfy the previous conditions, in order to observe results like in figure 4.

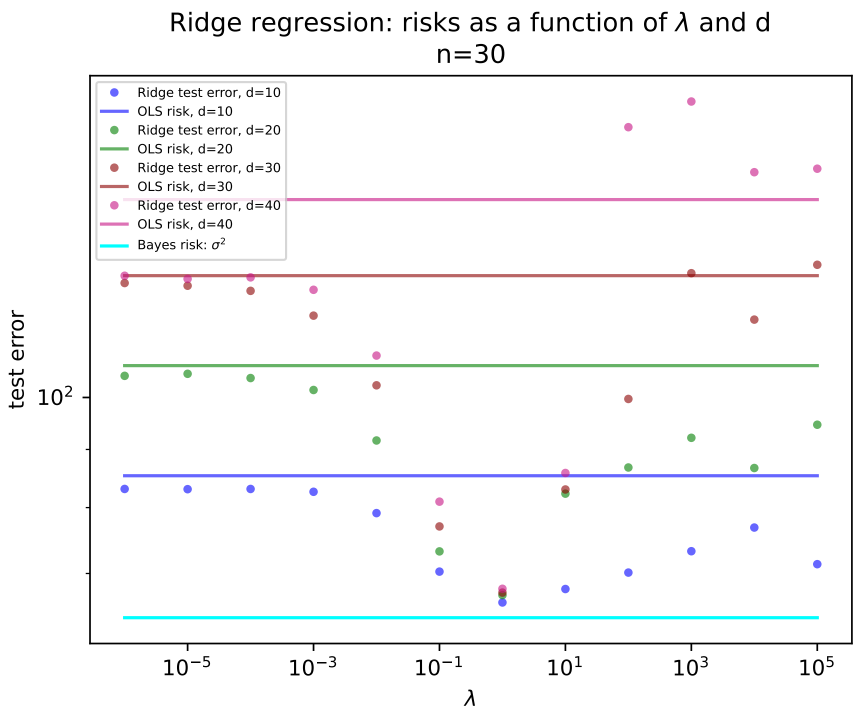

FIGURE 4 – Comparison of OLS and Ridge regression in a specific setting.

λ is called a hyperparameter, a very important notion in machine learning. Most optimization algorithms (hence ML algorithms) have hyperparameters.

4 HYPERPARAMETER SEARCH

There can be several hyperparameters for given a problem : e.g. a regularization constant λ, a solver type, a number of iterations of an iterative algorithm, etc.

In some contexts, there are some theoretical results that guarantee that there exists some hyperparameter values are useful or optimal. However, importantly, we often can not compute these values because they depend on some unknown quantities, typically like our σ. Hence, in order to find good hyperparameters, one must experiment and automate the search, typically using libraries.

4.1 Search method

4.1.1 *Grid search*

The most basic approach to look for hyperparameters is to choose a list of values for each and iterate through a grid containing all possible combinations.

https://scikit-learn.org/stable/modules/grid_search.html#

See also random search.

4.1.2 *Smarter search*

More modern approaches resort to better approaches and many frameworks exists to do so. The **optuna** library is a great tool, with many options.

`https://optuna.org/`.

`https://github.com/optuna/optuna-dashboard`

4.1.3 *Cross validation*

Still linked to the topic of hyperparameter tuning, cross validation is another very important concept that you can start getting familiar with

`https://scikit-learn.org/stable/modules/cross_validation.html`

`https://en.wikipedia.org/wiki/Cross-validation_(statistics)`

In scikit-learn, many estimators have wrappers in order to directly tune hyperparameters with cross validation.

`https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeCV.html`

4.2 Application

Pick a toy dataset from scikit-learn, choose a learning problem and perform a hyperparameter optimization using grid search and then optuna on that problem.

`https://scikit-learn.org/stable/datasets/toy_dataset.html`