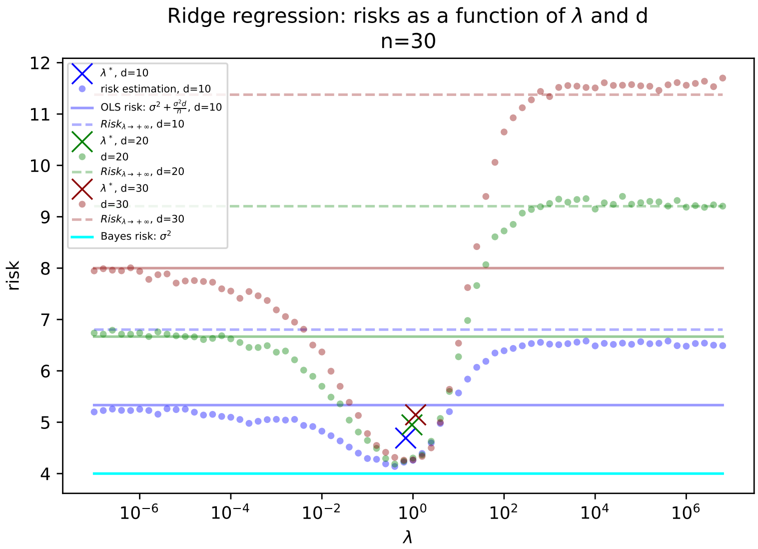
# FTML practical session 3

20 mars 2025




## TABLE DES MATIÈRES

## INTRODUCTION

In this session, we study Ridge regression, which is an extension of OLS, and which leads us to the notion of **hyperparameter** (HP). The choice of HPs is part of the topic of **model selection**. There are two important aspects of HP tuning that we will discuss :

— how to select HP set values

— how to evaluate the quality of a set of HPs

https://scikit-learn.org/stable/model_selection.html

### Template files

You can find some template files in the **exercice_1_ridge/** folder.

## 1 RIDGE REGRESSION AND REGULARIZATION

The goal of this exercice is to experimentally observe the benefit of using Ridge regression instead of OLS. As we have just witnessed in the previous session, the excess risk of the OLS estimator in the linear model, fixed design is $\frac{\sigma^2 d}{n}$, where $n$ is the number of samples and $d$ the number of features. We also keep the same notations for the quantities defined for OLS such as the design matrix $X \in \mathbb{R}^{n,d}$, $\theta \in \mathbb{R}^d$, $y \in \mathbb{R}^n$.

### 1.1 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

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

As you can see, there is a parameter $\lambda$, that controls the magnitude of the **regularization**. The goal is to penalize estimators with a large amplitude. $\lambda$ is called a **hyperparameter**, a very important notion in machine learning. Most optimization algorithms (hence ML algorithms) have hyperparameters and we will study several methods to choose them.

We admit the following proposition for now, but we will prove it during the lectures (you may also try to prove it now).

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

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

What is $\hat{\theta}_\lambda$ when $\lambda = 0$ ?

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 $\lambda$ ? If yes, how can we tune a good value for $\lambda$ ? The answer will depend on the values of $n$, $d$, and $\sigma$ : for some combinations of values, Ridge will indeed perform strictly better than OLS in terms of generalization (like in Figure 1).

### 1.2 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 $\lambda$. This might happen if one or several of the following conditions are satisfied :

— $\sigma$ 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 1. You might use either scikit-learn or a manual implementation of Ridge regression, based on the OLS functions written during the previous session (both for data generation and estimator learning).

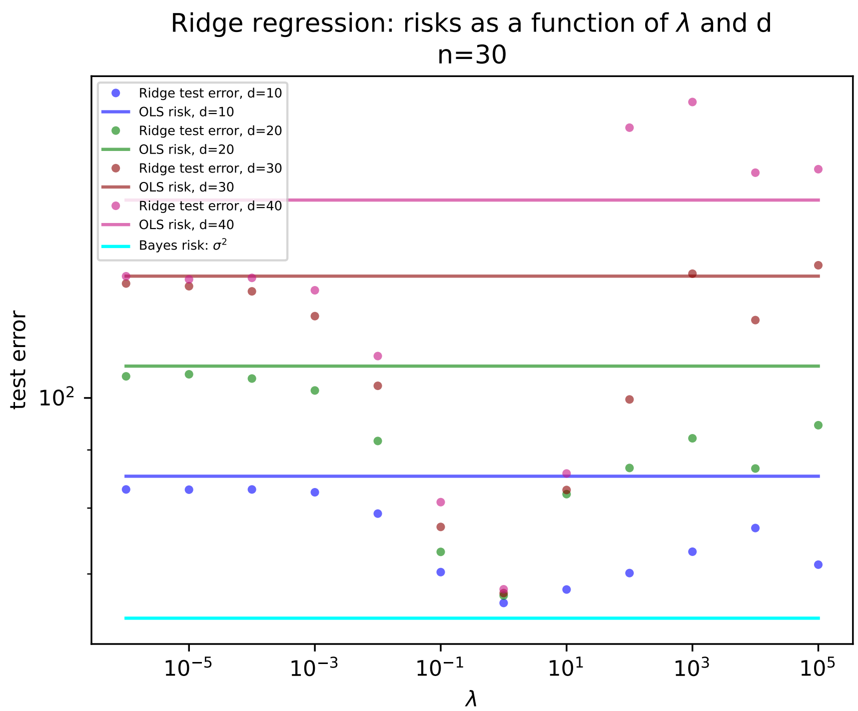



FIGURE 1 – Comparison of OLS and Ridge regression in a specific setting.

## 2 VALIDATION TESTS AND CROSS VALIDATION

### 2.1 Problem

In section 1, we saw that in some situations, some values of the HP exist, that strictly improve the performance. In some cases, these optimal values might be found theoretically, with equations that depend on the statistical properties of the data, like for instance on $\sigma$. However, in practical applications, these quantities are unknown and experimentation is needed to find good HPs.

## 2.2 Validation sets

### 2.2.1 *Definitions*

A basic approach for comparing hyperparameter sets is to separate the dataset into 3 parts :

— a train set : e.g. given a set of HPs, used to solve the optimization problem if we do empirical risk minimization.

— a validation set : used to compute the score of the predictor that was learned with this set of HPs.

— a test set : after trying all sets of HPs, used to compute the score of the predictor that had the best validation score (score on the validation set)

Ideally, the test set must be used **only once** : it gives an unbiased estimation of the generalization error, as opposed to the validation error, that is **not** unbiased.

### 2.2.2 *Issues*

There are several possible issues with this approach, the main one being that it is possible that the HPs "overfit the validation set". This means that it is possible that a given set of HPs lead to a good score on a specific choice of the validation set, but does not generalize as well. As the choice of the validation set is usually arbitraty and random, this might lead to a large variance of the predictor obtained, and a worse generalization error.

### 2.2.3 *Simulation*

The previous issue is not always a big one. Usually, it happens when predictors with large variance are used, and / or with datasets that are too small to contain enough relevant statistical information about the data. We will illustrate this on a classification problem.

Perform a simulation with the train / validation / test split in order to perform a hyperparameter tuning for which the test score is significantly, and robustly worse than the validation score.

Suggestions :

— use for instance the digits dataset or the wine from scikit, and extract only a random subset of this dataset; e.g. 300 points or less.
`https://scikit-learn.org/stable/datasets/toy_dataset.html`

— use **train_test_split()** from scikit to perform the splits.
`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

— use **ParameterGrid()** to build a **grid** that you can iterate over to compare HPs.
`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.ParameterGrid.html`

— use **DecisionTreeClassifier()** or **SVC()** (support vector classifier) from scikit and choose some HPs from these classes to tune the tree learned.
`https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`
`https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier`

You will see that the validation score is not always worse than the test score, depending on the HP grid, and of the size of the dataset!

## 2.3 Cross validation

**Cross-validation (CV)** is another approach to test the quality of a set of HPs, that is usually more robust to statistical noise. The dataset is split only in a train set and a test set, but the train set is used differently. It is itsself splitted multiple times in **folds**, and an **average** of errors on test folds is computed. The main issue with CV is that it is longer : more optimizations have to be performed.

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

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

### 2.3.1 *Simulation*

Run cross validation on the same problem as before in order to obtain a better test score than with the classical train / validation / test approach. You will notice that cross validation itsself has parameters, such as the number of splits in the dataset.

— https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

— https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

— 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