Mathematical Tools for Data Science

Vincent Divol

April 28, 2022

Contents

Additional material

- These lecture notes are partly based on Francis Bach's book **Learning Theory from First Principles** [Bach, 2022].
- We also refer to Stephen Boyd and Lieven Vandenberghe's book **Convex optimization** [Boyd et al., 2004] for Chapter 2.
- Convergence results on gradient descent and Newton's method in Chapter 2 are taken from Yurii Nesterov's **Lectures on convex optimization** [Nesterov, 2018], whereas their stochastic analogues presentend in Chapter 3 are taken from [Shalev-Shwartz and Ben-David, 2014, Chapter 14]
- A nice "cheat sheet" presenting all convergence results for (stochastic) gradient descent has been written by Robert M. Gower [Gower, 2018].
- A great reference to learn about kernels is **Learning with kernels** by Bernhard Schölkopf and Alexander Smola [Smola and Schölkopf, 1998].
- We refer to the **Lectures on the Nearest Neighbor Method** by Gérard Biau and Luc Devroye for k-NN methods presented in Chapter 5 [Biau and Devroye, 2015].
- A great tutorial on spectral clustering (presented in Chapter 6) has been written by Ulrike von Luxburg [von Luxburg, 2007].

Chapter 1

Empirical risk minimization

The problem of **supervised learning** can be expressed in the following way: we are given a family of inputs $(x_1, \dots, x_n)$ on a set $\mathcal{X}$, and associated outputs $(y_1, \dots, y_n)$ on a set $\mathcal{Y}$. Given a new input $x \in \mathcal{X}$, can we predict what the associated output $y \in \mathcal{Y}$ will be? There are two large families of learning problems, the case where $\mathcal{Y}$ is a finite set (**classification task**), and the case where the ouputs y_is take continuous values, typically $\mathcal{Y} = \mathbb{R}$ (**regression task**).

Example 1.0.1.

1. Each input x_i is a picture that represents an animal y_i. In this case, a picture x_i is represented by the RGB value (Red, Green, Blue) of each of its pixels, so that $\mathcal{X} = \mathbb{R}^{3K}$, where K is a number of pixels. The set $\mathcal{Y}$ is the set of animals that are depicted. This is a classification task.

2. Each input x_i is a review of a movie, and y_i is the rating associated with the review. Given a new review x, the goal is to guess if the user who wrote the review liked the movie (y is high) or disliked it (y is low). The set of inputs $\mathcal{X}$ is the set of texts, whereas $y \in \mathcal{Y} = [0, 1]$ represents a grade between 0 and 1. This is a regression task.

3. Each input x_i is a patient that is described by different physiological parameters (including e.g. sex, age, blood pressure, etc.) and a treatment that was given to them, whereas y_i is equal to 1 (the patient is cured) or -1 (the patient is not cured) ($\mathcal{Y} = \{-1, 1\}$). The goal is then

to understand what is the efficiency of different treatments for different profiles of patients.

Before going further, there is an important question to address, concerning the model assumptions made on the inputs $(x_1, \dots, x_n)$ and the outputs $(y_1, \dots, y_n)$. As it is most often the case in machine learning approaches, we will assume that both the inputs and the ouputs are **random**. In particular, it may be possible that for two inputs $x_i = x_j$ that are equal, the corresponding outputs y_i and y_j are different (this for instance makes sense in Example (3), where different patients having the same profile, given the same treatment, may experience different outcomes). We will moreover always assume that the pairs $((x_1, y_1), \dots, (x_n, y_n))$ are **independent and identically distributed** (i.i.d.). This is the simplest assumption, that is reasonable in a large number of cases, although there exist relevant problems where such an assumption is too strong (e.g. if the observations (x_i, y_i) arrive in a sequential manner, it may then be the case that the law of the variable (x_i, y_i) depends on the time i at which it was observed).

To distinguish between deterministic variables and random ones, we will use a **bold font** to refer to the latter, that is $(\mathbf{x}, \mathbf{y})$ is random, whereas (x, y) is deterministic. The law of the observations $(\mathbf{x_i}, \mathbf{y_i})$ will be denoted by P, whereas $P_{\mathbf{x}}$ is the law of the first marginal $\mathbf{x_i}$ and $P_{\mathbf{y}}$ is the law of the second marginal $\mathbf{y_i}$. Remark that P is a probability measure on the space $\mathcal{X} \times \mathcal{Y}$, whereas $P_{\mathbf{x}}$ is a probability measure on $\mathcal{X}$ and $P_{\mathbf{y}}$ is a probability measure on $\mathcal{Y}$. We will write $\mathbb{E}_P[f(\mathbf{x}, \mathbf{y})]$ for the expectation of some function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ with respect to P, whereas the conditional expectation of $f(\mathbf{x}, \mathbf{y})$ given that $\mathbf{x} = x$ is written as $\mathbb{E}_P[f(\mathbf{x}, \mathbf{y})|\mathbf{x} = x]$. We will sometimes only write $\mathbb{E}$ instead of $\mathbb{E}_P$ when it is clear what the underlying law is.

1.1 Risks and losses

How do we measure the quality of a prediction?

The term "predicting" is here rather vague, and the data scientist may want to give it different meanings depending on the context. For instance, in binary classification ($\mathcal{Y} = \{-1, 1\}$), a possible goal is to minimize the number of misclassifications on average. However, the two ouputs -1 and 1 may not play a symmetrical role, and we may want to favorize predictions that make very few mistakes when choosing -1, at the price of making more mistakes

when choosing 1. For example, in medical settings, the output $y = -1$ can represent the fact that the patient x is sick and deserves further treatment, while $y = 1$ means that the patient x is healthy. Then, it is a more serious mistake to predict that $y = 1$ when in fact $y = -1$ than the opposite, and we want to take this into account when assessing the quality of a predictor.

The problem is even more striking when the ouput space $\mathcal{Y}$ is multidimensional, say $\mathcal{Y} = \mathbb{R}^d$. In that case, possible ways of measuring how a prediction $y' = (y'_1, \dots, y'_d)$ is close to the output $y = (y_1, \dots, y_d)$ include:

- the L_∞-norm: $\|y' - y\|_\infty := \max_{j=1,\dots,d} |y'_j - y_j|$,
- the L_p-norm $\|y' - y\|_p := (\sum_{j=1}^d |y'_j - y_j|^p)^{1/p}$,
- the weighted L_p-norm $\|y' - y\|_{p,w} := (\sum_{j=1}^d w_j |y'_j - y_j|^p)^{1/p}$, where $w = (w_1, \dots, w_d)$ is a vector of positive weights,
- the dot product $y' \cdot y = \sum_{j=1}^d y'_j y_j$.

There are no "better" choices of distances among the one listed above, they each represent a different way of measuring how two points in the output space $\mathcal{Y}$ are similar. For instance, choosing the ℓ_1-norm instead of the ℓ_2-norm indicates that we want to penalize less the fact that a huge error was made on one of the entries y'_j of the prediction, which might be a desirable feature in some problems.

More generally, we will work with a general loss function ℓ on the set of outputs.

Definition 1.1.1 (Loss function)**.** *A* ***loss function*** *is a nonnegative function* $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, +\infty)$.

The goal is then to find a function $f : \mathcal{X} \to \mathcal{Y}$ such that $\ell(f(\mathbf{x}), \mathbf{y})$ is small on new samples $((\mathbf{x'_1}, \mathbf{y'_1}), \dots, (\mathbf{x'_{n'}}, \mathbf{y'_{n'}}))$, that we call the testing sample. We will here always assume that the testing sample is also i.i.d. of law P[1]. The goal is then to minimize the average loss on the testing sample, that we call the **expected risk** or the **test error**.

[1]In many practical situations, the law of the testing sample is actually different from the law P on which the predictor was trained. This situation is referred to as **covariate shift** in the literature and requires the development of specific techniques. This issue will never be addressed in these notes.

Definition 1.1.2 (Expected risk)**.** *Let P be a probability measure on $\mathcal{X} \times \mathcal{Y}$ and ℓ be a loss function. Given a function $f : \mathcal{X} \to \mathcal{Y}$, the P-**risk of** f is given by*

$$\mathcal{R}_P(f) := \mathbb{E}_P[\ell(\mathbf{y}, f(\mathbf{x}))]. \tag{1.1}$$

The best prediction $f_P^\star$ is the one that minimizes $\mathcal{R}_P(f)$, and is called the **Bayes predictor**. It turns out that we can give an expression of the Bayes predictor.

Theorem 1.1.3 (Optimality of Bayes predictor)**.** *Let P be a probability measure on $\mathcal{X} \times \mathcal{Y}$. The function $f \mapsto \mathcal{R}_P(f)$ is minimized at $f_P^\star$ that is defined by*

$$f_P^\star(x) \in \arg\min_{z \in \mathcal{Y}} \mathbb{E}_P[\ell(\mathbf{y}, z)|\mathbf{x} = x]. \tag{1.2}$$

Proof. Let $f : \mathcal{X} \to \mathcal{Y}$ be a function. Define the function $\Psi : (x, z) \in \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{E}_P[\ell(\mathbf{y}, z)|\mathbf{x} = x]$. By definition, we have the equality $\Psi(x, f_P^\star(x)) = \min_{z \in \mathcal{Y}} \Psi(x, z)$. By the law of total expectation, we obtain

$$\mathcal{R}_P(f) = \mathbb{E}_P[\ell(\mathbf{y}, f(\mathbf{x}))] = \mathbb{E}_P[\Psi(\mathbf{x}, f(\mathbf{x}))] \geq \mathbb{E}_P[\Psi(\mathbf{x}, f_P^\star(\mathbf{x}))] = \mathcal{R}_P(f_P^\star). \tag{1.3}$$

As this hold for every function f, this implies the conclusion. □

Let us consider concrete examples of losses ℓ and associated Bayes predictors.

Example 1.1.4.

- Consider the problem of binary classification ($\mathcal{Y} = \{-1, 1\}$) with the loss $\ell(y, y') = \mathbf{1}\{y \neq y'\}$. Then, $\mathbb{E}_P[\ell(\mathbf{y}, 1)|\mathbf{x} = x] = P(\mathbf{y} = 1|\mathbf{x} = x)$. We call this quantity the **regression function** $\eta(x)$. We have $\mathbb{E}_P[\ell(\mathbf{y}, -1)|\mathbf{x} = x] = 1 - \eta(x)$. Therefore,

$$f_P^\star(x) = \begin{cases} 1 & \text{if } \eta(x) > 1/2, \\ -1 & \text{otherwise.} \end{cases} \tag{1.4}$$

  In other words, the Bayes predictor follows the following intuitive rule: if the probability of observing the output $\mathbf{y} = 1$ given that $\mathbf{x} = x$ is larger than $1/2$, then we predict 1. Otherwise, we predict -1.

- Let $\mathcal{Y} = \mathbb{R}$ and let $\ell(y, y') = |y - y'|^2$. Remark that the function $z \mapsto \mathbb{E}[(\mathbf{a} - z)^2]$ is minimized at $z = \mathbb{E}[\mathbf{a}]$. This implies that the function $z \mapsto \mathbb{E}_P[\ell(\mathbf{y}, z)|\mathbf{x} = x] = \mathbb{E}_P[|\mathbf{y} - z|^2|\mathbf{x} = x]$ is minimized for $z = \mathbb{E}_P[\mathbf{y}|\mathbf{x} = x]$. Therefore, the Bayes predictor is in this case given by the conditional expectation

$$f_P^\star(x) = \mathbb{E}_P[\mathbf{y}|\mathbf{x} = x]. \tag{1.5}$$

1.2 Empirical risk

If the Bayes predictor is indeed the optimal one, it has a major drawback: computing it requires to know what the law of the sample P is! In practice, we only have access to the training sample $((\mathbf{x_1}, \mathbf{y_1}), \dots, (\mathbf{x_n}, \mathbf{y_n}))$, and P is unknown. We cannot therefore use the Bayes predictor, and our goal will be to design predictors f that can be computed based on the observations, and that will (hopefully) behave almost as well as the Bayes predictor.

A powerful method to do so consists in minimizing the **empirical risk**.

Definition 1.2.1 (Empirical risk)**.** *Let $(\mathbf{x_1}, \mathbf{y_1}), \dots, (\mathbf{x_n}, \mathbf{y_n})$ be a training sample from law P and ℓ be a loss function. The **empirical risk** of the sample is given by*

$$f \mapsto \mathcal{R}_n(f) := \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{y_i}, f(\mathbf{x_i})). \tag{1.6}$$

The law of large number indicates that $\mathcal{R}_n(f) \simeq \mathcal{R}_P(f)$ when n is very large. Therefore, one may expect that minimizing $\mathcal{R}_n$ is a good strategy to build a predictor with small P-risk. There is however a caveat: for most losses ℓ, one can always find **many** functions f such that $\mathcal{R}_n(f) = 0$, some of them being **very irregular**. For instance, in a regression setting with $\mathcal{X} = \mathcal{Y} = \mathbb{R}$ and $\ell(y, y') = |y - y'|$, there are infinitely many functions (continuous or discontinuous) such that $f(x_i) = y_i$, and $\mathcal{R}_n(f) = 0$ for all such functions. Such functions f will then behave badly on new observations $(\mathbf{x}, \mathbf{y})$ sampled according to P: the risk $\mathcal{R}_P(f)$ will be large although $\mathcal{R}_n(f) = 0$.

This minimizing strategy must therefore be improved. An idea consists in minimizing $\mathcal{R}_n$ over a restricted class of functions $\mathcal{F}$, that will encode some regularity that we expect the Bayes predictor to have.

Definition 1.2.2 (Empirical risk minimizer)**.** *Let* $(\mathbf{x_1}, \mathbf{y_1}), \dots, (\mathbf{x_n}, \mathbf{y_n})$ *be a training sample from law* P *and* ℓ *be a loss function. Let* $\mathcal{F}$ *be a class of functions from* $\mathcal{X}$ *to* $\mathcal{Y}$*, that we also call a* ***model****. An* ***empirical risk minimizer*** $\hat{f}_{\mathcal{F}}$ *is any function in* $\mathcal{F}$ *that attains the minimum*

$$\min_{f \in \mathcal{F}} \mathcal{R}_n(f). \tag{1.7}$$

There is a crucial element that needs to be directly mentioned: the computation of the empirical risk minimizer requires the minimization of a potentially complicated functional on an arbitrary set $\mathcal{F}$. This problem is in general untractable, and we will address in Chapters 2 and 3 how to solve it in the case where the loss ℓ is convex. There are however some specific examples where no optimization procedures are required, and an explicit form of the solution exists, as in the following example.

Example 1.2.3 (Linear regression). Consider the regression problem on $\mathbb{R}^d$ with the quadratic loss (that is $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$, and $\ell(y, y') = |y - y'|^2$). In this setting, a popular choice is to consider the class $\mathcal{F}_{\text{lin}}$ of linear predictors of the form $f_\theta : x \mapsto \theta^T x$. The empirical risk is then given by

$$\begin{aligned}\mathcal{R}_n(f_\theta) &= \frac{1}{n}\sum_{i=1}^{n} |\mathbf{y_i} - f_\theta(\mathbf{x_i})|^2 = \frac{1}{n}\sum_{i=1}^{n} |\mathbf{y_i} - \theta^T \mathbf{x_i}|^2 \\ &= \frac{1}{n}\|\mathbf{Y} - \mathbf{X}\theta\|^2,\end{aligned}$$

where

$$\mathbf{Y} = \begin{pmatrix} \mathbf{y_1} \\ \vdots \\ \mathbf{y_n} \end{pmatrix} \text{ and } \mathbf{X} = \begin{pmatrix} \mathbf{x_1} \\ \vdots \\ \mathbf{x_n} \end{pmatrix}.$$

In this case, the empirical risk minimizer $\hat{f}_{\mathcal{F}_{\text{lin}}}$ is given by a linear regression. We may also further restrict the model by considering only vectors θ having a small ℓ_2-norm (ridge regression) or a small ℓ_1-norm (lasso regression).

Decomposition of the empirical risk: underfitting and overfitting

Let us analyze the performance of an empirical risk minimizer. Let $\mathcal{R}_P^\star := \min_f \mathcal{R}_P(f) = \mathcal{R}_P(f_P^\star)$. We want to understand when the risk of $\hat{f}_{\mathcal{F}}$ is not

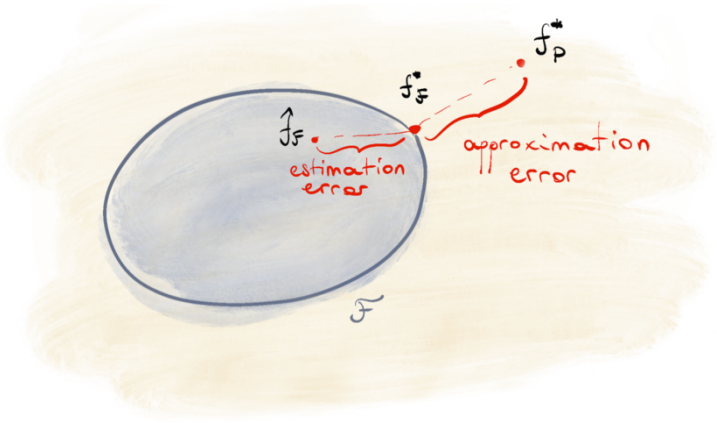


Figure 1.1: Decomposition of the excess risk $\mathcal{R}_P(\hat{f}_P) - \mathcal{R}(f_P^\star)$ into the approximation error $\mathcal{R}_P(f_{\mathcal{F}}^\star) - \mathcal{R}(f_P^\star)$ and the estimation error $\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \mathcal{R}(f_{\mathcal{F}}^\star)$.

much larger than $\mathcal{R}_P^\star$, that is we want to bound the **excess risk**

$$\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \mathcal{R}_P^\star. \tag{1.8}$$

The **excess risk** can be decomposed into

$$\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \mathcal{R}_P^\star = \underbrace{(\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \inf_{f \in \mathcal{F}} \mathcal{R}_P(f))}_{\text{estimation error}} + \underbrace{(\inf_{f \in \mathcal{F}} \mathcal{R}_P(f) - \mathcal{R}_P^*)}_{\text{approximation error}}. \tag{1.9}$$

Remark first that the two error terms are nonnegative.

- The **approximation error** $\inf_{f \in \mathcal{F}} \mathcal{R}_P(f) - \mathcal{R}_P^*$ is a deterministic quantity (it does not depend on the observations) that measures how far the best predictor on $\mathcal{F}$ is from the best predictor (the Bayes predictor). The larger $\mathcal{F}$ is, the smaller this error becomes.

- It is less immediate to understand how the **estimation error** $\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \inf_{f \in \mathcal{F}} \mathcal{R}_P(f)$ behaves. Assume that the infimum $\inf_{f \in \mathcal{F}} \mathcal{R}_P(f)$ is at-

tained at some function $f^\star_{\mathcal{F}}$. We can then write

$$\begin{aligned}
&\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \inf_{f\in\mathcal{F}} \mathcal{R}_P(f) = \mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \mathcal{R}_P(f^\star_{\mathcal{F}}) \\
&= (\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \mathcal{R}_n(\hat{f}_{\mathcal{F}})) + (\mathcal{R}_n(\hat{f}_{\mathcal{F}}) - \mathcal{R}_n(f^\star_{\mathcal{F}})) + (\mathcal{R}_n(f^\star_{\mathcal{F}}) - \mathcal{R}_P(f^\star_{\mathcal{F}})) \\
&\leq (\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \mathcal{R}_n(\hat{f}_{\mathcal{F}})) + 0 + (\mathcal{R}_n(f^\star_{\mathcal{F}}) - \mathcal{R}_P(f^\star_{\mathcal{F}})) \\
&\leq \sup_{f\in\mathcal{F}} (\mathcal{R}_P(f) - \mathcal{R}_n(f)) + (\mathcal{R}_n(f^\star_{\mathcal{F}}) - \mathcal{R}_P(f^\star_{\mathcal{F}})),
\end{aligned} \tag{1.10}$$

where the first inequality follows from the fact that $\hat{f}_{\mathcal{F}}$ minimizes $\mathcal{R}_n$ on $\mathcal{F}$ by definition, so that $\mathcal{R}_n(\hat{f}_{\mathcal{F}}) \leq \mathcal{R}_n(f^\star_{\mathcal{F}})$. If we believe that this inequality is tight (and it is in many cases), then the estimation error is linked to the quantity $\sup_{f\in\mathcal{F}}(\mathcal{R}_P(f) - \mathcal{R}_n(f))$, that is the uniform deviation between the empirical risk $\mathcal{R}_n$ and its expectation $\mathcal{R}_P$ on the class $\mathcal{F}$. This quantity increases with the size of $\mathcal{F}$ and decreases with the number of observations n.

Example 1.2.4 (Polynomial regression). Let $\mathcal{X} = [0,1]$, $\mathcal{Y} = \mathbb{R}$ and ℓ be the square loss. Let $\mathcal{F}_d$ be the set of polynomials of degree d and let $\hat{f}_d := \hat{f}_{\mathcal{F}_d}$. We test the performance of the estimator $\hat{f}_d$ on the *Real estate valuation data set* [Yeh and Hsu, 2018] (taken from the UCI Machine Learning Repository). On this dataset, the goal is to predict the price y of a house based on several features (coordinates, house age, number of nearby convenience stores, etc.). For visualization sake, we consider a single feature x representing the house age and compute the predictors $\hat{f}_d$ for d ranging from 1 to 10 (see Figure 1.2). We observe the two predicted regimes. For $d = 1$, the model is too simple and both the empirical risk $\mathcal{R}_n(\hat{f}_1)$ and the risk $\mathcal{R}_P(\hat{f}_1)$ (that is approximated by the empirical risk on the testing sample) are large: the model is underfitting. For $d = 10$, the empirical risk becomes small, but the risk $\mathcal{R}_P(\hat{f}_{10})$ is really large: our model is too complicated and we are overfitting.

We have discovered a fundamental phenomenon: the excess risk of an empirical risk minimizer is driven by two contrary forces. The first one is the approximation error, that measures how far the model $\mathcal{F}$ is close from "the truth", and will be large if our model is overly simplistic, a regime that we call **underfitting**. The second one is the estimation error, that measures how the set of observations $(\mathbf{y_1}, f(\mathbf{x_1})), \dots, (\mathbf{y_n}, f(\mathbf{x_n}))$ is able to capture the behavior of the expectation $\mathbb{E}_P[\ell(\mathbf{y}, f(\mathbf{x}))]$ over all functions $\mathcal{F}$. If $\mathcal{F}$ is very

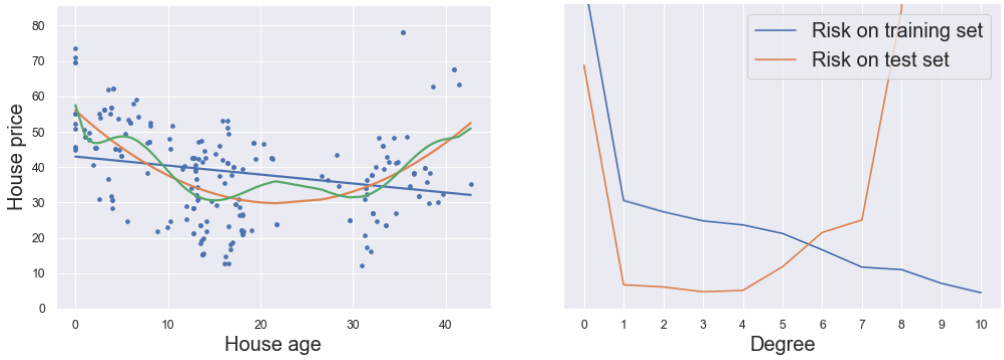


Figure 1.2: Left: linear predictor $\hat{f}_1$ (blue), quadratic predictor $\hat{f}_2$ (orange) and predictor of degree 10 (green). Right: Empirical risk $\mathcal{R}_n(\hat{f}_d)$ as a function of d (in blue) and average risk of $\hat{f}_d$ on the testing sample (in orange). As expected, the empirical risk $\mathcal{R}_n(\hat{f}_d)$ is a nonincreasing function of d.

large, then there will typically be many different functions with very small empirical risk (as in Example 1.2.4), so that the minimizer $\hat{f}_P$ might be very different from $f_P^\star$. We call this regime **overfitting**.

1.3 Bound on the estimation error in binary classification

We focus in this section on the classification task $\mathcal{Y} = \{-1, 1\}$ with the $0-1$ loss $\ell(y, y') = \mathbf{1}\{y \neq y'\}$. Our aim is to understand how the estimation error $\mathcal{R}_P(\hat{f}_\mathcal{F}) - \inf_{f\in\mathcal{F}} \mathcal{R}_P(f)$ scales with $\mathcal{F}$. We first consider the case where the set $\mathcal{F}$ of predictors is finite, and then consider the more delicate case of infinite classes of predictors $\mathcal{F}$ by introducing the concept of VC dimension.

1.3.1 Finite number of predictors

Assume first that the set $\mathcal{F} = \{f_1, \dots, f_k\}$ is finite. The estimation error is bounded using this general inequality.

Theorem 1.3.1 (Maximal inequality)**.** *Let* $\mathbf{z_1}, \dots, \mathbf{z_k}$ *be real valued random variables such that there exists a constant* $\sigma > 0$ *with* $\mathbb{E}[e^{\lambda \mathbf{z_j}}] \leq e^{\lambda^2\sigma^2/2}$ *for*

every $\lambda > 0$. *Then,*

$$\mathbb{E}[\max_{j=1,\dots,k} \mathbf{z_j}] \leq \sigma\sqrt{2\log k}. \tag{1.11}$$

Proof. A first (naive) idea to bound the maximum of a collection of nonnegative numbers (a_j) consists in using that

$$\max_{j=1,\dots,k} a_j \leq \sum_{j=1}^{k} a_j. \tag{1.12}$$

Of course, this bound is often very bad. However, using (1.12) makes sense if the maximum of the a_js (say a_{j_0}) is much larger than the other ones: in that case, the sum is roughly equal to the max.

We will enforce this situation by replacing each a_j by $\exp(\lambda a_j)$ for some parameter $\lambda > 0$. If λ is very large, then indeed $\exp(\lambda a_{j_0})$ is much larger than the other values $\exp(\lambda a_j)$, and therefore the bound

$$\max_{j=1,\dots,k} e^{\lambda a_j} \leq \sum_{j=1}^{k} e^{\lambda a_j} \tag{1.13}$$

becomes a much more reasonable one. Another way of writing this equation is the following:

$$\max_{j=1\dots k} a_j \leq \frac{1}{\lambda}\log\left(\sum_{j=1}^{k} e^{\lambda a_j}\right). \tag{1.14}$$

Let us now fix $a_j = \mathbf{z_j}$. We obtain

$$\mathbb{E}[\max_{j=1\dots k} \mathbf{z_j}] \leq \mathbb{E}\left[\frac{1}{\lambda}\log\left(\sum_{j=1}^{k} e^{\lambda \mathbf{z_j}}\right)\right]. \tag{1.15}$$

We are now in position to use Jensen's inequality.

Lemma 1.3.2 (Jensen's inequality)**.** *Let* $\mathbf{x}$ *be a real valued random variable and* $\varphi : \mathbb{R} \to \mathbb{R}$ *be a convex function. Then,*

$$\varphi(\mathbb{E}[\mathbf{x}]) \leq \mathbb{E}[\varphi(\mathbf{x})]. \tag{1.16}$$

Applying Jensen's inequality to $\varphi = \exp$, we obtain

$$\mathbb{E}[\max_{j=1\dots k} \mathbf{z_j}] \leq \frac{1}{\lambda}\log\left(\mathbb{E}\left[\sum_{j=1}^{k} e^{\lambda \mathbf{z_j}}\right]\right). \tag{1.17}$$

The assumption $\mathbb{E}[e^{\lambda \mathbf{z_j}}] \leq e^{\lambda^2\sigma^2/2}$ yields

$$\mathbb{E}[\max_{j=1\dots k} \mathbf{z_j}] \leq \frac{\log k}{\lambda} + \frac{\lambda\sigma^2}{2}. \tag{1.18}$$

We choose $\lambda > 0$ to minimize this expression: the optimal value is $\lambda = \sqrt{2\log k}/\sigma$ and we obtain the final bound. □

Let us now turn to the quantity $\mathbb{E}[\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \inf_{f\in\mathcal{F}} \mathcal{R}_P(f)]$. According to the inequality (1.10), it is enough to bound

$$\begin{aligned}
&\mathbb{E}\left[\sup_{f\in\mathcal{F}}(\mathcal{R}_P(f) - \mathcal{R}_n(f)) + (\mathcal{R}_n(f^\star_{\mathcal{F}}) - \mathcal{R}_P(f^\star_{\mathcal{F}}))\right] \\
&= \mathbb{E}[\max_{j=1,\dots,k}(\mathcal{R}_P(f_j) - \mathcal{R}_n(f_j))] + \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n \mathbf{1}\{\mathbf{y_i} \neq f^\star_P(\mathbf{x_i})\}\right] - \mathbb{P}(\mathbf{y} \neq f^\star_P(\mathbf{x})) \\
&= \mathbb{E}[\max_{j=1,\dots,k}(\mathcal{R}_P(f_j) - \mathcal{R}_n(f_j))].
\end{aligned}$$

Let us apply the maximal inequality to the random variables $\mathbf{z_j} = \mathcal{R}_P(f_j) - \mathcal{R}_n(f_j)$ for $j = 1, \dots, k$. We have

$$\mathcal{R}_P(f_j) - \mathcal{R}_n(f_j) = \mathbb{P}(f_j(\mathbf{x}) \neq \mathbf{y}) - \frac{1}{n}\sum_{i=1}^n \mathbf{1}\{f_j(\mathbf{x_i}) \neq \mathbf{y_i}\}. \tag{1.19}$$

The independence of the observations $(\mathbf{x_i}, \mathbf{y_i})$ yields

$$\mathbb{E}[e^{\lambda(\mathcal{R}_P(f_j) - \mathcal{R}_n(f_j))}] = \prod_{i=1}^n \mathbb{E}[e^{\frac{\lambda}{n}(\mathbb{P}(f_j(\mathbf{x})\neq\mathbf{y}) - \mathbf{1}\{f_j(\mathbf{x_i})\neq\mathbf{y_i}\})}]. \tag{1.20}$$

Let $p_j = \mathbb{P}(f_j(\mathbf{x_i}) \neq \mathbf{y_i})$. Then,

$$\mathbb{E}[e^{\frac{\lambda}{n}(p_j - \mathbf{1}\{f_j(\mathbf{x_i})\neq\mathbf{y_i}\})}] = p_j e^{-\frac{\lambda}{n}(1-p_j)} + (1-p_j)e^{\frac{\lambda}{n}p_j}. \tag{1.21}$$

The maximum of this quantity is obtained at $p_j = 1/2$, so that

$$\mathbb{E}[e^{\frac{\lambda}{n}(p_j - \mathbf{1}\{f_j(\mathbf{x_i})\neq\mathbf{y_i}\})}] \leq \frac{e^{\frac{\lambda}{n}} + e^{-\frac{\lambda}{n}}}{2} \leq e^{\lambda^2/(2n^2)}, \tag{1.22}$$

where we use the standard inequality $e^\lambda + e^{-\lambda} \leq 2e^{\lambda^2/2}$. Therefore, the random variable $\mathbf{z_j}$s satisfy the condition of Theorem 1.3.1 with $\sigma = 1/\sqrt{n}$. We thus obtain the following result.

Theorem 1.3.3 (Error bound in expectation on the estimation error in binary classification: finite case)**.** *Assume that $\mathcal{F}$ contains k elements and that ℓ is the $0-1$ loss. Then*

$$\mathbb{E}[\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \inf_{f\in\mathcal{F}} \mathcal{R}_P(f)] \leq \mathbb{E}[\sup_{f\in\mathcal{F}}(\mathcal{R}_P(f) - \mathcal{R}_n(f))] \leq \sqrt{\frac{2\log k}{n}}. \quad (1.23)$$

By the central limit theorem, we expect the fluctuations of $\mathcal{R}_n(f)$ around $\mathcal{R}_P(f)$ to be of order $1/\sqrt{n}$. Theorem 1.3.3 asserts that the uniform deviations of $\mathcal{R}_n(f)$ over a family of k functions f are also of order $1/\sqrt{n}$.

1.3.2 Vapnik-Chervonenkis dimension

The left hand side of Theorem 1.3.3 diverges as the size k of the class $\mathcal{F}$ of predictors grow (although at a slow $\sqrt{\log k}$ rate). Does this mean that everything is hopeless when $\mathcal{F}$ is infinite and that we should stick with a finite set $\mathcal{F}$ in practice? Hopefully not! Indeed, in many situations, one chooses $\mathcal{F}$ to be some infinite "well-behaved" family. This is for instance the case for linear regression, where the set of predictors is the (infinite) set $\mathcal{F}_{\text{lin}} = \{x \mapsto x^T\theta,\ \theta \in \mathbb{R}^d\}$. For the classification problem, it turns out the size of the set $\mathcal{F}$ is not a good measure of its complexity. Rather, the "effective" size of a set $\mathcal{F}$ is measured by the number of classifications $(f(\mathbf{x_1}), \ldots, f(\mathbf{x_n}))$ over all $f \in \mathcal{F}$.

Let us first remark that, even if $\mathcal{F}$ is infinite, then the set of possible classifications $\mathcal{C}_{\mathcal{F}}(\mathbf{x_1}, \ldots, \mathbf{x_n}) := \{(f(\mathbf{x_1}), \ldots, f(\mathbf{x_n})),\ f \in \mathcal{F}\}$ is always finite, and of size at most 2^n (each $f(\mathbf{x_i})$ is equal to ± 1). Using a technical tool called *symmetrization*, one can show that one can indeed replace the size of $\mathcal{F}$ in Theorem 1.3.3 by the size of the classification set $\mathcal{C}_{\mathcal{F}}(\mathbf{x_1}, \ldots, \mathbf{x_n})$.

Lemma 1.3.4. *Let $\mathcal{N}_{\mathcal{F}}(\mathbf{x_1}, \ldots, \mathbf{x_n})$ be the size of the set $\mathcal{C}_{\mathcal{F}}(\mathbf{x_1}, \ldots, \mathbf{x_n})$. Then,*

$$\mathbb{E}[\sup_{f\in\mathcal{F}}(\mathcal{R}_P(f) - \mathcal{R}_n(f))] \leq 2\mathbb{E}\left[\sqrt{\frac{2\log \mathcal{N}_{\mathcal{F}}(\mathbf{x_1}, \ldots, \mathbf{x_n})}{n}}\right]. \quad (1.24)$$

For many different examples, the quantity $\mathcal{N}_{\mathcal{F}}(\mathbf{x_1}, \ldots, \mathbf{x_n})$ is much smaller than the maximum possible value 2^n, making inequality (1.24) non trivial. Consider for instance the set $\mathcal{F}_0 = \{x \mapsto \mathbf{1}\{x^{(1)} \geq a\},\ a \in \mathbb{R}\}$. Then, the set $\mathcal{C}_{\mathcal{F}_0}(\mathbf{x_1}, \ldots, \mathbf{x_n})$ contains at most $n+1$ elements (see Figure 1.3). We

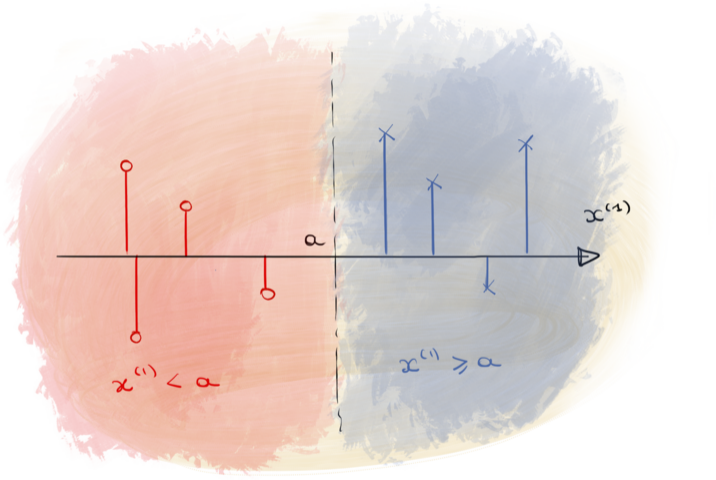


Figure 1.3: Given n points $x_1, \ldots, x_n$ in $\mathbb{R}^d$, the number $\mathcal{N}_{\mathcal{F}_0}(x_1, \ldots, x_n)$ of classifications using a classifier $f \in \mathcal{F}_0$ is at most $n+1$. Indeed, if we order the points such that their first coordinates are in increasing order $x_1^{(1)} \leq x_2^{(1)} \leq \cdots \leq x_n^{(1)}$, then choosing a classifier in $\mathcal{F}_0$ amounts to choosing the largest index which will be classified as -1, and there are $n+1$ such indices.

therefore directly obtain a bound of order $\sqrt{\log n/n}$ in this case, which is comparable to the bound that we obtained in the previous section (Theorem 1.3.3), although $\mathcal{F}_0$ is infinite.

For general sets $\mathcal{F}$, bounding directly $\mathcal{N}_{\mathcal{F}}(\mathbf{x_1}, \dots, \mathbf{x_n})$ is delicate, and there are even certain values of n for which finding a good bound is hopeless. Indeed, assume that for every $x_1, \dots, x_n \in \mathcal{X}$ and every $y_1, \dots, y_n \in \{0, 1\}$, one can find a function $f \in \mathcal{F}$ with $f(x_i) = y_i$ for $i = 1, \dots, n$. Then, the empirical risk $\mathcal{R}_n(\hat{f}_{\mathcal{F}}) = \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{f(\mathbf{x_i}) \neq \mathbf{y_i}\}$ is always equal to 0. We are exactly in the overfitting regime where we expect the estimation error to be large. In that case, we have $\mathcal{N}_{\mathcal{F}}(\mathbf{x_1}, \dots, \mathbf{x_n}) = 2^n$, making inequality (1.24) vacuous.

Definition 1.3.5 (Vapnik-Chervonenkis dimension)**.** *Let $\mathcal{F}$ be a set of functions from $\mathcal{X}$ to $\{-1, 1\}$. The **Vapnik-Chervonenkis dimension** $\mathrm{VC}(\mathcal{F})$ of $\mathcal{F}$ is defined as the largest number n such that there exists a configuration $x_1, \dots, x_n \in \mathcal{X}$ such that for every possible classifications $y_1, \dots, y_n \in \{-1, 1\}$, there exists $f \in \mathcal{F}$ with $f(x_i) = y_i$ for $i = 1, \dots, n$. We set $\mathrm{VC}(\mathcal{F}) = +\infty$ if this condition holds for every $n \in \mathbb{N}$.*

According to the previous discussion, for $n \leq \mathrm{VC}(\mathcal{F})$, the set $\mathcal{F}$ is overfitting and there is no hope in bounding the estimation error. However, should $n \gg \mathrm{VC}(\mathcal{F})$, then the next lemma asserts that $\mathcal{N}_{\mathcal{F}}(x_1, \dots, x_n)$ scales at most polynomially with n. In particular, it is *much* smaller than 2^n!

Lemma 1.3.6 (Sauer's lemma)**.** *Let $\mathcal{F}$ be a set with finite VC dimension. Let $n > 2\mathrm{VC}(\mathcal{F})$. Then, for every $x_1, \dots, x_n \in \mathcal{X}$, we have*

$$\log \mathcal{N}_{\mathcal{F}}(x_1, \dots, x_n) \leq \mathrm{VC}(\mathcal{F}) \log\left(\frac{en}{\mathrm{VC}(\mathcal{F})}\right). \tag{1.25}$$

Putting Lemma 1.3.4 and Lemma 1.3.6 together, we obtain the following result.

Theorem 1.3.7 (Error bound in expectation on the estimation error in binary classification: with VC dimension)**.** *Assume that $\mathcal{F}$ has a finite VC dimension $\mathrm{VC}(\mathcal{F})$ and that ℓ is the $0-1$ loss. Then, for $n \geq 2\mathrm{VC}(\mathcal{F})$,*

$$\begin{aligned} \mathbb{E}[\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \inf_{f \in \mathcal{F}} \mathcal{R}_P(f)] &\leq \mathbb{E}[\sup_{f \in \mathcal{F}} (\mathcal{R}_P(f) - \mathcal{R}_n(f))] \\ &\leq 2\sqrt{\frac{2\mathrm{VC}(\mathcal{F})}{n} \log\left(\frac{en}{\mathrm{VC}(\mathcal{F})}\right)}. \end{aligned} \tag{1.26}$$

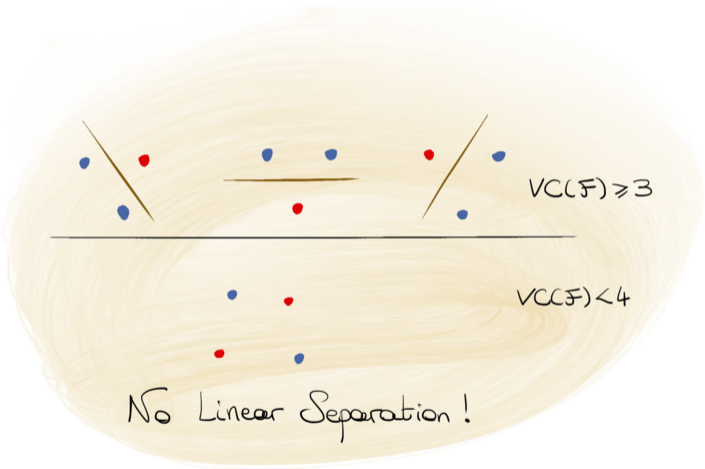


Figure 1.4: Let $\mathcal{F}$ be the set of linear classifier in $\mathbb{R}^2$. There exists a set of three points, such that linear classifiers can output all possible classifications (top). However, for all configurations of four points, there exists a classification that cannot be realized by a linear classifier (bottom). Therefore, $\mathrm{VC}(\mathcal{F}) = 3$.

We conclude by giving some properties of the VC dimension.

Proposition 1.3.8. *Let $\mathcal{F}$ be a set of functions from $\mathcal{X}$ to $\{-1, 1\}$.*

1. *If $\mathcal{F}$ is of size k, then $\mathrm{VC}(\mathcal{F}) \leq \log_2(k)$.*
2. *It $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{F}$ is the set of linear classifiers (that is $f \in \mathcal{F}$ is of the form $f(x) = 1$ if x belongs to some halfspace H and -1 otherwise), then $\mathrm{VC}(\mathcal{F}) = d + 1$.*
3. *Let $s \geq 1$ be an integer and let $\mathcal{F}_s$ be the set of classifiers of the form $\max_{j=1,\dots,s} f_j$ for some functions f_js in $\mathcal{F}$. Then,*

$$\mathrm{VC}(\mathcal{F}_s) \leq \mathrm{VC}(\mathcal{F})(2s \log_2(3s)). \tag{1.27}$$

Remark 1.3.9. So far, we have only given tools to bound the estimation error $\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \inf_{f\in\mathcal{F}} \mathcal{R}_P(f)$. What about the approximation error $\inf_{f\in\mathcal{F}} \mathcal{R}_P(f) - \mathcal{R}_P^*$? This quantity will depend on the regularity of the Bayes predictor $f_P^\star$.

Assume for the sake of simplicity that $\mathbf{x}$ is uniform on the cube $\mathcal{X} = [0,1]^d$ and $\mathbf{y} = f_0(\mathbf{x})$ for some function $f_0 : \mathcal{X} \to \{-1, 1\}$ (that is there is no noise) for $d \geq 2$. Then, the Bayes risk $\mathcal{R}_P^\star$ is equal to 0 and the approximation error is equal to

$$\inf_{f \in \mathcal{F}} \mathbb{P}(f(\mathbf{x}) \neq f_0(\mathbf{x})). \tag{1.28}$$

This probability represents the area of the cube where f and f_0 differ. Let us consider a toy example to get some intuition. Assume that f_0 is equal to 1 on some smooth convex set of volume 1 (see Chapter 2) and -1 outside. Consider the model $\mathcal{F}_s$ consisting of intersections of s halfplanes, that is every $f \in \mathcal{F}_s$ is of the form

$$f(x) = \begin{cases} 1 & \text{if } x \in \bigcap_{j=1}^s H_j \\ -1 & \text{otherwise,} \end{cases} \tag{1.29}$$

where $H_1, \ldots, H_s$ are s halfplanes. It is then known [Bronstein, 2008] that the approximation error is bounded by $c_d s^{-2/(d-1)}$ for some constant c_d. As expected, this quantity decreases as s gets larger. Using Theorem 1.3.7 and Proposition 1.3.8, we obtain the following bound on the excess risk (for n larger than s):

$$\mathbb{E}[\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \mathcal{R}_P^\star] \leq c'_d \sqrt{\frac{s \log(s) \log(n)}{n}} + c_d s^{-2/(d-1)}, \tag{1.30}$$

where c'_d is some positive constant. Letting $s = n^{(d-1)/(d+3)}$, we obtain a bound of order

$$\mathbb{E}[\mathcal{R}_P(\hat{f}_{\mathcal{F}}) - \mathcal{R}_P^\star] \leq c''_d \log(n) n^{-2/(d+3)} \tag{1.31}$$

for some other constant c''_d. Note that this rate of convergence is extremely slow for large d: we refer to this phenomenon as the **curse of dimensionality**.

Appendix

Symmetrization

We provide here a proof of Lemma 1.3.4. This is a delicate proof, that uses a key technical tool used symmetrization. We call a random sign $\mathbf{e}$ that is equal to $+1$ with probability $1/2$ and -1 with probability $1/2$ as a **Rademacher random variable**.

Lemma 1.3.10. *Let T be a set and let $\mathbf{a_1}, \dots, \mathbf{a_n}$ be i.i.d. random variables in $\mathbb{R}^T$: each $\mathbf{a_i}$ is a function from T to $\mathbb{R}$. We assume that for every $t \in T$, $\mathbb{E}[\mathbf{a_i}(t)]$ is finite. Let $\mathbf{e_1}, \dots, \mathbf{e_n}$ be n i.i.d. Rademacher random variables, independent from the $\mathbf{a_i}$s. We have*

$$\mathbb{E}[\sup_{t\in T} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{a_i}(t) - \mathbb{E}[\mathbf{a_i}(t)])] \leq 2 \cdot \mathbb{E}[\sup_{t\in T} \sum_{i=1}^{n} \mathbf{e_i}\mathbf{a_i}(t)]. \tag{1.32}$$

Proof. We introduce $\mathbf{a'_1}, \dots, \mathbf{a'_n}$ an independent copy from $\mathbf{a_1}, \dots, \mathbf{a_n}$. The random vectors $\mathbf{a_i} - \mathbf{a'_i}$ are independent and symmetric, such that $\mathbf{a'_i} - \mathbf{a_i}$ has the same law as $\mathbf{a_i} - \mathbf{a'_i}$. One can check that $\mathbf{a_i} - \mathbf{a'_i}$ has the same law as $\mathbf{e_i}(\mathbf{a_i} - \mathbf{a'_i})$. Therefore,

$$\begin{aligned}\mathbb{E}[\sup_{t\in T} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{a_i}(t) - \mathbb{E}[\mathbf{a_i}(t)])] &= \mathbb{E}[\sup_{t\in T} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{a_i}(t) - \mathbb{E}[\mathbf{a'_i}(t)])] \\ &= \mathbb{E}[\sup_{t\in T} \mathbb{E}[\frac{1}{n}\sum_{i=1}^{n}(\mathbf{a_i}(t) - \mathbf{a'_i}(t))|\mathbf{a_1}, \dots, \mathbf{a_n}])].\end{aligned}$$

The function $z \mapsto \sup_{t\in t} z(t)$ is convex. Therefore, by Jensen's inequality,

$$\begin{aligned}\mathbb{E}[\sup_{t\in T} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{a_i}(t) - \mathbb{E}[\mathbf{a_i}(t)])] &\leq \mathbb{E}[\mathbb{E}[\sup_{t\in T} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{a_i}(t) - \mathbf{a'_i}(t))]] \\ &= \mathbb{E}[\sup_{t\in T} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{a_i}(t) - \mathbf{a'_i}(t))] \\ &= \mathbb{E}[\sup_{t\in T} \frac{1}{n}\sum_{i=1}^{n}\mathbf{e_i}(\mathbf{a_i}(t) - \mathbf{a'_i}(t))] \\ &= \mathbb{E}[\sup_{t\in T} \frac{1}{n}\sum_{i=1}^{n}\mathbf{e_i}\mathbf{a_i}(t)] + \mathbb{E}[\sup_{t\in T} \frac{1}{n}\sum_{i=1}^{n} -\mathbf{e_i}\mathbf{a'_i}(t))] \\ &= 2 \cdot \mathbb{E}[\sup_{t\in T} \frac{1}{n}\sum_{i=1}^{n}\mathbf{e_i}\mathbf{a_i}(t)].\end{aligned}$$

□

We apply this general inequality with $T = \mathcal{F}$ to the random variables

$\mathbf{a_i}(f) = \ell_{01}(f(\mathbf{x_i}), \mathbf{y_i}) = \mathbf{1}\{f(\mathbf{x_i}) \neq \mathbf{y_i}\}$ to obtain

$$\begin{aligned}\mathbb{E}[\sup_{f\in\mathcal{F}}(\mathcal{R}_P(f) - \mathcal{R}_n(f))] &\leq 2\cdot\mathbb{E}[\sup_{f\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^{n}\mathbf{e_i}\mathbf{1}\{f(\mathbf{x_i}) \neq \mathbf{y_i}\}] \\ &= 2\cdot\mathbb{E}[\sup_{u\in\mathcal{C}_\mathcal{F}(\mathbf{x_1},\ldots,\mathbf{x_n})}\frac{1}{n}\sum_{i=1}^{n}\mathbf{e_i}\mathbf{1}\{u_i \neq \mathbf{y_i}\}],\end{aligned} \tag{1.33}$$

where $u = (u_1, \ldots, u_n)$ is any element of $\mathcal{C}_\mathcal{F}(\mathbf{x_1}, \ldots, \mathbf{x_n})$, the set of classifications of $\mathbf{x_1}, \ldots, \mathbf{x_n}$ using a classifier $f \in \mathcal{F}$. Conditionally on the observations $(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})$, the random variables $\frac{1}{n}\sum_{i=1}^{n}\mathbf{e_i}\mathbf{1}\{u_i \neq \mathbf{y_i}\}$ satisfy the assumptions of Theorem 1.3.1 with $\sigma = 1/\sqrt{n}$. Also, there are $\mathcal{N}_\mathcal{F}(\mathbf{x_1}, \ldots, \mathbf{x_n})$ such random variables. Then, by applying Theorem 1.3.1 conditionally on the observations $(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})$, we obtain that

$$\mathbb{E}[\sup_{f\in\mathcal{F}}(\mathcal{R}_P(f) - \mathcal{R}_n(f))] \leq 2\cdot\mathbb{E}\left[\sqrt{\frac{2\log\mathcal{N}_\mathcal{F}(\mathbf{x_1}, \ldots, \mathbf{x_n})}{n}}\right], \tag{1.34}$$

that is exactly Lemma 1.3.4.

To summarize, we have used symmetrization to replace a supremum over an infinite number of random variables (each random variable $\mathcal{R}_P(f) - \mathcal{R}_n(f)$ are different because $\mathcal{R}_P(f)$ is a priori different for every function $f \in \mathcal{F}$) to a supremum over only a finite number of $\mathcal{C}_\mathcal{F}(\mathbf{x_1}, \ldots, \mathbf{x_n})$ random variables.

Chapter 2

Convex optimization

2.1 Convexification of the $0-1$ loss

In the previous chapter, we studied in detail the properties of the empirical risk minimizer $\hat{f}_{\mathcal{F}}$ in binary classification. This estimator is defined as the minimizer of the functional

$$f \in \mathcal{F} \mapsto \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{\mathbf{y_i} \neq f(\mathbf{x_i})\}. \tag{2.1}$$

This functional is highly discontinuous, and is actually piecewise constant on each output $A(z_1, \dots, z_n) := \{f \in \mathcal{F},\ \forall i = 1, \dots, n,\ f(\mathbf{x_i}) = z_i\}$ for $z_1, \dots, z_n \in \{-1, 1\}$. The number of such sets is exactly $\mathcal{N}_{\mathcal{F}}(\mathbf{x_1}, \dots, \mathbf{x_n})$. If we believe that Sauer's lemma is tight (and it is in many cases), then this number is exponential in the VC dimension. Even for very simple sets, such as the class of linear classifiers, this number will be very large even for moderate dimension of the input space $\mathcal{X}$. This computational blow up shows the impossibility of minimizing (2.1) in many situations.

To overcome this problem, we make a simple remark. The set $\{-1, 1\}$ is a subset of $\mathbb{R}$. Therefore, we can consider the classification task as an instance of a regression task, choose a predictor $g : \mathcal{X} \to \mathbb{R}$, and obtain a classifier by letting $f = \operatorname{sgn} \circ g$, where sgn is the sign function (equal to $+1$ on $[0, +\infty)$ and to -1 on $(-\infty, 0)$). Minimizing the $0-1$ risk of the classifier $f = \operatorname{sgn} \circ g$ amounts to minimizing the function

$$g \mapsto \mathbb{E}_P[\mathbf{1}\{\operatorname{sgn}(g(\mathbf{x})) \neq \mathbf{y}\}] = \mathbb{E}_P[\mathbf{1}\{g(\mathbf{x})\mathbf{y} < 0\}]. \tag{2.2}$$

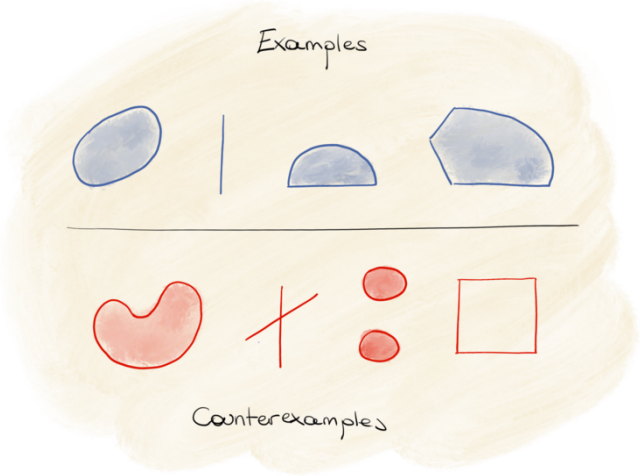


Figure 2.1: Examples and counterexamples of convex sets.

The loss function $\ell(y, y') = \mathbf{1}\{yy' < 0\}$ is still not continuous, making the minimization of the empirical risk as difficult as before. However, this loss can be replaced by other convex losses that are similar to this one. This process is referred to as the convexification of the loss. We will come back to this problem in Section 2.5 after introducing the necessary background on convex optimization.

2.2 Convex functions

We start by recalling some elementary definitions.

Definition 2.2.1 (Convex set)**.** *A set $A \subset \mathbb{R}^d$ is **convex** if, given two points x and y in A, the segment joining x and y is included in A:*

$$\forall x, y \in A,\ \forall t \in [0,1],\ tx + (1-t)y \in A. \tag{2.3}$$

Let $A \subset \mathbb{R}^D$ and let f be a function defined on A. We define the **epigraph** of f as

$$\mathrm{epi}(f) := \{(x,t) \in A \times \mathbb{R} :\ f(x) \geq t\}. \tag{2.4}$$

Definition 2.2.2 (Convex function)**.** *Let A be a convex set and let $f : A \to \mathbb{R}$. We say that f is **convex** if $\mathrm{epi}(f) \subset \mathbb{R}^{d+1}$ is convex. Equivalently, f is*

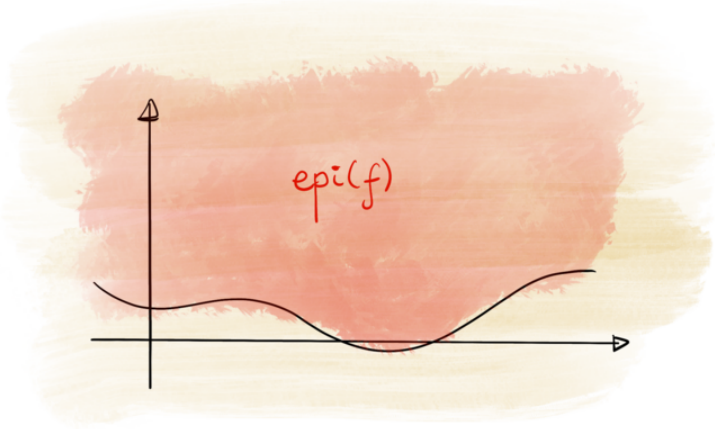


Figure 2.2: The epigraph of a function f (whose graph is displayed in black).

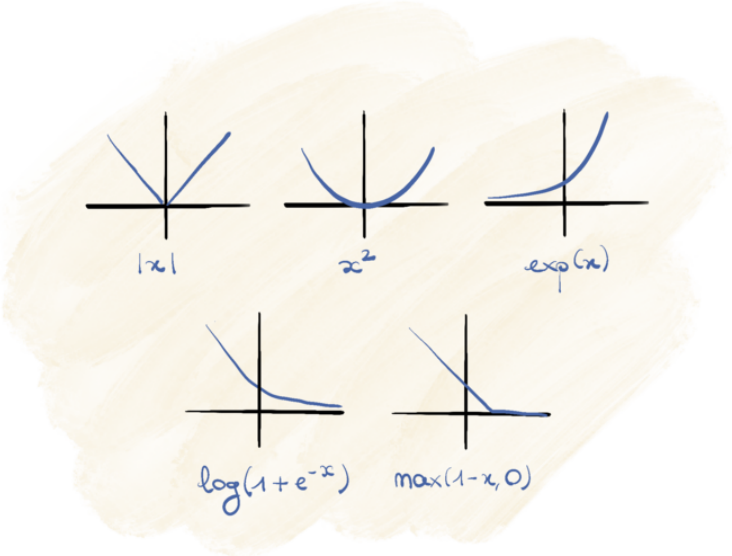


Figure 2.3: Examples of convex functions.

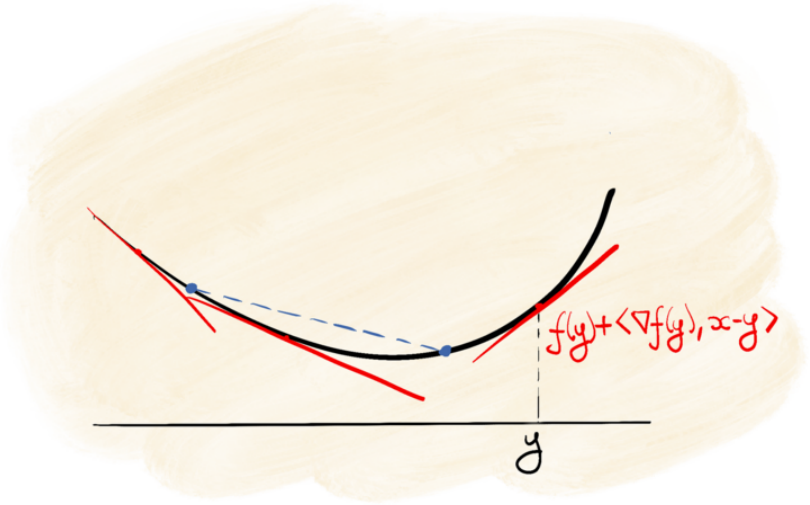


Figure 2.4: Blue: the chord joining two points of the graph of a convex function is above the graph. Red: the graph of a function f stays above its tangent lines.

convex if

$$\forall x, y \in A,\ \forall t \in [0,1],\ f(tx + (1-t)y) \leq tf(x) + (1-t)f(y). \tag{2.5}$$

We call A the domain of f, denoted by $\mathrm{dom}(f)$.

Intuitively speaking, a function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if, when we let a bead rolls from a point on the graph of f, the bead always arrives to the minimum of f (that is the point of lowest altitude). Alternatively, a function f is convex if a chord joining two points on the graph of f is always "above" the graph of f.

A convex function is always continuous on the interior of its domain. However, it may not be differentiable (take $f : x \mapsto |x|$). If we assume that it is differentiable, then the differential has to be monotone.

Proposition 2.2.3. *Let f be a convex differentiable function.*

- *If $d = 1$, then f' is nondecreasing.*
- *If $d \geq 2$, then, for all $x, y \in \mathrm{dom}(f)$, we have*

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq 0 \tag{2.6}$$

and

$$f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle. \tag{2.7}$$

Geometrically, this means that a function is convex if its graph is always above its tangent curves (see Figure 2.2).

What can we say about the second derivative of f (should it exist)? Using the previous proposition, for $d = 1$, we should have $f'' \geq 0$. The analogue of the second derivative for $d \geq 2$ is the Hessian matrix $\nabla^2 f$. The good notion of nonnegativity for symmetric matrices is given by the following condition: for every $x \in \mathrm{dom}(f)$ and $u \in \mathbb{R}^d$, we have $u^\top \nabla^2 f(x) u \geq 0$. We then say that $\nabla^2 f(x)$ is positive semi-definite and we write $\nabla^2 f(x) \succcurlyeq 0$. Equivalently, the eigenvalues of $\nabla^2 f(x)$ are nonnegative. If all the eigenvalues are larger than some value α, then we write $\nabla^2 f(x) \succcurlyeq \alpha \mathrm{Id}$. Likewise, if all the eigenvalues are smaller than some number β, then we write $\nabla^2 f(x) \preccurlyeq \beta \mathrm{Id}$. These are respectively equivalent to having $u^\top \nabla^2 f(x) u \geq \alpha \|u\|^2$ and $u^\top \nabla^2 f(x) u \leq \beta \|u\|^2$ for all $u \in \mathbb{R}^d$.

Proposition 2.2.4. *Assume that f is twice differentiable.*

- *If $d = 1$, then $f'' \geq 0$.*
- *If $d \geq 2$, the Hessian matrix satisfies $\nabla^2 f(x) \succcurlyeq 0$ for every $x \in \mathrm{dom}(f)$.*

Theoretical guarantees for the optimization algorithms presented in the next sections will hold if the convex function f is sufficiently well-behaved. A key ingredient is to assert that f is "really convex" in a quantitative way.

Definition 2.2.5. *A real valued convex function f is said to be α-strongly convex if for all $x, y \in \mathrm{dom}(f)$ and $t \in [0, 1]$,*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) - \frac{\alpha}{2} t(1-t)\|x - y\|^2. \tag{2.8}$$

For example, a linear function $f : x \mapsto \langle a, x \rangle$ is not strongly convex, as we have the identity $f(tx+(1-t)y) = tf(x)+(1-t)f(y)$ for such a function.

Proposition 2.2.6. *Assume that f is twice differentiable. The following conditions are equivalent.*

- *The function f is α-strongly convex.*

- *For all* $x, y \in \mathrm{dom}(f)$,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\alpha}{2}\|x - y\|^2. \tag{2.9}$$

- *The function* $x \mapsto f(x) - \frac{\alpha}{2}\|x\|^2$ *is convex.*
- *We have* $\nabla^2 f(x) \succcurlyeq \alpha \mathrm{Id}$ *for every* $x \in \mathrm{dom}(f)$.

Geometrically speaking, a function is α-strongly convex if its graph is sufficiently curved at every point (more precisely, it has curvature at least $1/\alpha$ in all directions). The prototypical example of α-strongly convex function is the function $x \mapsto \alpha\|x\|^2/2$ (this is clear with the third characterization). Note that strong convexity implies that f has a unique minimizer $x^\star$. Indeed, take x far away from 0 and apply (2.8) to $t = 1/\|x\|$ and $y = 0$. We obtain

$$f(x) \geq \|x\| \left(f(x/\|x\|) + \frac{\alpha}{2}\frac{1}{\|x\|}\left(1 - \frac{1}{\|x\|}\right)\|x\|^2 \right). \tag{2.10}$$

In particular, $f(x)$ goes to infinity as $\|x\|$ goes to infinity. This implies that the infimum of f is attained on some sufficiently large ball, and by continuity of f the infimum is attained at at least one point. Given two minimizers x_1 and x_2, we have, for any $t \in (0, 1)$,

$$\min f \leq f(tx_1 + (1-t)x_2) \leq t \min f + (1-t)\min f - \frac{\alpha}{2}t(1-t)\|x_1 - x_2\|^2. \tag{2.11}$$

This inequality is possible only if $x_1 = x_2$, implying the uniqueness of the minimizer.

Furthermore, α-strongly convex implies the *Polyak–Lojasiewicz condition* (or PL condition). The PL condition asserts that the function f cannot grow too fast near its minimizer $x^\star$: for all $x \in \mathrm{dom}(f)$,

$$f(x) - f(x^\star) \leq \frac{1}{2\alpha}\|\nabla f(x)\|^2. \tag{2.12}$$

A second condition that ensures good convergence properties is the regularity of the gradient of f.

Definition 2.2.7. *Let f be a real-valued function. We say that f is β-smooth if it is differentiable and its gradient ∇f is β-Lipschitz:*

$$\forall x, y \in \mathrm{dom}(f),\ \|\nabla f(x) - \nabla f(y)\| \leq \beta\|x - y\|. \tag{2.13}$$

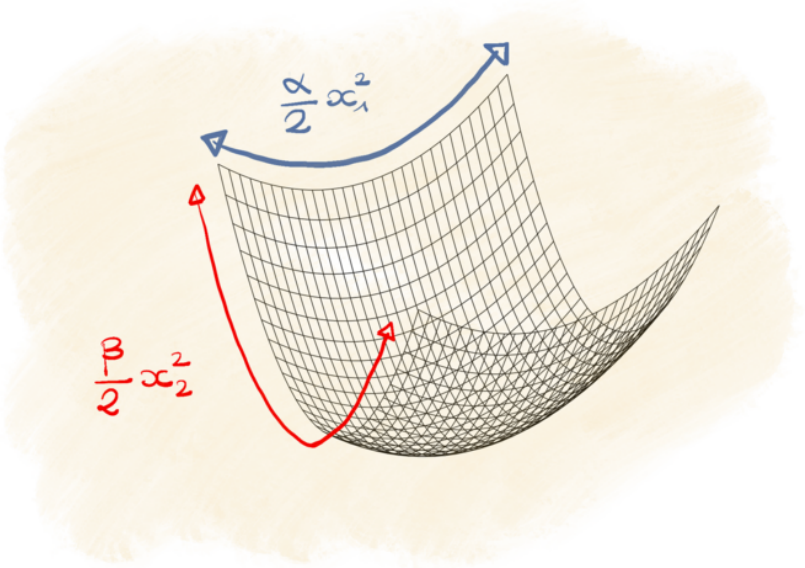


Figure 2.5: The function $f_{\alpha,\beta}$ is α-strongly convex and β-smooth.

Proposition 2.2.8. *Assume that f is twice differentiable. The following conditions are equivalent.*

- *The function f is β-smooth.*
- *For all $x, y \in \mathrm{dom}(f)$,*

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\beta}{2}\|x - y\|^2. \tag{2.14}$$

- *We have $\nabla^2 f(x) \preccurlyeq \beta \mathrm{Id}$ for every $x \in \mathrm{dom}(f)$.*

Geometrically, the β-smoothness condition implies that the graph of the function is not too curved. The prototypical example of a function that is both α-strongly convex and β-smooth is the quadratic function

$$f_{\alpha,\beta} : x \in \mathbb{R}^2 \mapsto \frac{\alpha}{2}x_1^2 + \frac{\beta}{2}x_2^2. \tag{2.15}$$

The graph of this function has an elongated bowl shape, with large width $1/\alpha$ in direction x_1, and small width $1/\beta$ in direction x_2. The Hessian of the

function is given by

$$\nabla^2 f_{\alpha,\beta}(x) = \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix}, \tag{2.16}$$

showing that it is indeed α-strongly convex and β-smooth, see Figure 2.2.

2.3 Gradient descent

We now investigate the problem of finding the minimum of a convex function f. The most important algorithm to find such a minimum is the gradient descent algorithm. The general idea is simple: to find the minimum of a convex function, start at point x, take one step in the direction with largest negative slope, and repeat the procedure.

Definition 2.3.1. *Let f be a real-valued function and let $x_0 \in \mathbb{R}^d$. Let $T \in \mathbb{N}$ be a number of steps and let $(s_t)_{t=0,\dots,T}$ be a sequence of step sizes. The iterates of the gradient descent on f are defined by*

$$x_{t+1} := x_t - s_t \nabla f(x_t) \tag{2.17}$$

for $t \in \{0, \dots, T-1\}$.

We are now in position to state the main result of this chapter: the iterates of a gradient descent on a smooth and strongly-convex function converge towards the minimizer of f at a fast rate.

Proposition 2.3.2 (Convergence of gradient descent for smooth and strongly convex functions)**.** *Let f be a α-strongly convex and β-smooth function defined on $\mathbb{R}^d$. Consider the iterates $(x_t)_{t=0,\dots,T}$ of the gradient descent with constant step-size $s \leq 1/\beta$ and initialization x^0. Let $x^\star$ be the minimizer of f. Then, we have*

$$\begin{aligned} f(x^T) - f(x^\star) &\leq (1 - \alpha s)^T (f(x^0) - f(x^\star)) \\ &\leq \exp(-\alpha s T)(f(x^0) - f(x^\star)). \end{aligned} \tag{2.18}$$

We refer to such a rate of convergence as a **linear** rate of convergence as $\log(f(x^T) - f(x^\star))$ converges at a linear rate to $-\infty$.

Proof. The characterization of β-smoothness (2.14) implies that

$$\begin{aligned} f(x^{t+1}) = f(x^t - s\nabla f(x^t)) &\leq f(x^t) + \langle \nabla f(x^t), -s\nabla f(x^t)\rangle + s^2\frac{\beta}{2}\|\nabla f(x^t)\|^2 \\ &= f(x^t) + s\left(\frac{s\beta}{2} - 1\right)\|\nabla f(x^t)\|^2. \end{aligned} \tag{2.19}$$

Furthermore, by the PL condition (2.12), we have

$$\|\nabla f(x^t)\|^2 \geq 2\alpha(f(x^t) - f(x^\star)). \tag{2.20}$$

Plugging in this inequality in (2.19) yields

$$f(x^{t+1}) - f(x^\star) \leq (f(x^t) - f(x^\star))\left(1 + 2\alpha s\left(\frac{s\beta}{2} - 1\right)\right). \tag{2.21}$$

Having $s \leq 1/\beta$ implies that $2\alpha s\left(\frac{s\beta}{2} - 1\right) \leq -\alpha s$. Iterating this inequality, we obtain the conclusion. □

If we choose the step size s as the largest value allowed in the proposition, that is $s = 1/\beta$, we see that the linear rate of convergence is exactly equal to $\kappa^{-1} = \alpha/\beta$. The quantity $\kappa = \beta/\alpha$ is called the **condition number** of f. This corresponds to an upper bound on the ratio between the largest eigenvalue of the Hessian of f and its smallest. The dependence in κ in Proposition 2.3.2 indicates that functions f with a large condition number κ are harder to minimize.

More often, we want to control the number of iterations of the gradient descent needed to find a point x with $f(x) - f(x^\star) \leq \varepsilon$ for some fixed $\varepsilon > 0$. Proposition 2.3.2 implies that $T = \log(\varepsilon^{-1})/\kappa$ operations are needed. When minimizing the empirical risk, we want to compute a predictor that performs almost as well as the empirical risk minimizer $\hat{f}_{\mathcal{F}}$. Considerations in the previous chapter shows that one can expect the excess risk of $\hat{f}_{\mathcal{F}}$ to be at least of order $1/\sqrt{n}$. Therefore, taking ε of order $1/\sqrt{n}$ will most of the time be enough in our setting. This means that roughly $\log(n)/\kappa$ iterations are needed.

It is possible to relax some assumptions in the previous convergence result. For instance, if f is not strongly convex, one still has convergence of the iterates of the gradient descent, although at a much slower rate.

Proposition 2.3.3 (Convergence of gradient descent for smooth functions)**.** *Let f be a convex β-smooth function defined on $\mathbb{R}^d$. Consider the iterates $(x_t)_{t=0,\dots,T}$ of the gradient descent with constant step-size $s = 1/\beta$ and initialization x^0. Assume that f has a global minimizer $x^\star$. Then, we have*

$$f(x^T) - f(x^\star) \leq \frac{\beta \|x^0 - x^\star\|^2}{2T} \tag{2.22}$$

The dependence is not exponential in T anymore, but only polynomial. In particular, without the strong convexity assumption, $O(\beta/\varepsilon)$ iterations are needed, a number that is polynomial (and not logarithmic) in ε^{-1}. For $\varepsilon = 1/\sqrt{n}$, this leads to $O(\beta\sqrt{n})$ iterations. The proof of Proposition 2.3.3 is more delicate than the previous one and we do not include it. It can be found in [Bansal and Gupta, 2019, Theorem 3.3].

Consider a smooth convex function f that is strongly convex only on a neighborhood of its minimizer $x^\star$. Proposition 2.3.3 implies that gradient descent converges in a small number of steps to a point in this neighborhood. Then, gradient descent will linearly converge to the minimizer. Therefore, gradient descent will naturally adapt to the degree of convexity of the function, without the need to design any complicated procedure to choose the size step: we refer to this phenomenon as gradient descent being adaptive.

2.4 Newton's method

The gradient descent relies on a first order approximation of f: $f(x_0 + h)$ locally looks like $f(x_0) + \langle \nabla f(x_0), h\rangle$, so to decrease f, we should make a step in the direction minimizing the quantity $\langle \nabla f(x_0), h\rangle$ (and this direction is given by $-\nabla f(x_0)$). What if we try to write a second-order approximation of f around x_0? This yields to a second-order method called **Newton's method**. We have

$$f(x_0 + h) \approx P_{f,x_0}(h) = f(x_0) + \langle \nabla f(x_0), h\rangle + \frac{1}{2} h^T \nabla^2 f(x_0) h. \tag{2.23}$$

Let us find the minimum of the quadratic function P_{f,x_0}. Its gradient is equal to $\nabla f(x_0) + \nabla^2 f(x_0)h$. Therefore, should the Hessian be invertible at x_0 (for instance if f is strongly convex), then the minimum of P_{f,x_0} is attained at $h = (\nabla^2 f(x_0))^{-1} \nabla f(x_0)$.

Definition 2.4.1. *Let f be a real-valued function and let $x_0 \in \mathbb{R}^d$. Assume that the Hessian of f is always invertible. Let $T \in \mathbb{N}$ be a number of steps. A step of Newton's method is given by*

$$x^{t+1} := x^t - (\nabla^2 f(x^t))^{-1} \nabla f(x^t), \tag{2.24}$$

for $t \in \{0, \dots, T-1\}$.

To obtain convergence guarantees on Newton's method, we will need Lipschitz continuity of the Hessian matrix $\nabla^2 f$.

Proposition 2.4.2. *Let f be a twice differentiable real-valued convex function defined on $\mathbb{R}^d$ with $\|\nabla^2 f(x)u - \nabla^2 f(y)u\| \leq \gamma \|x - y\| \|u\|$ for every $x, y, u \in \mathbb{R}^d$. Assume further that f is α-strongly convex and β-smooth, with unique minimizer $x^\star$. Consider the iterates $(x_t)_{t=0,\dots,T}$ of the Newton's method with initialization x^0. Then, we have*

$$\|x^{t+1} - x^\star\| \leq \frac{\gamma}{2\alpha} \|x^t - x^\star\|^2. \tag{2.25}$$

In particular, if $\|x^0 - x^\star\| \leq \alpha/\gamma$, then we have

$$f(x^T) - f(x^\star) \leq \frac{\beta\alpha}{\gamma} 2^{-2^{T+1}}. \tag{2.26}$$

Proof. The first step consists in rewriting the update:

$$\begin{aligned}
x^{t+1} - x^\star &= x^t - x^\star - (\nabla^2 f(x^t))^{-1} \nabla f(x^t) \\
&= x^t - x^\star - (\nabla^2 f(x^t))^{-1} \int_0^1 \nabla^2 f(x^\star + \theta(x^t - x^\star))(x^t - x^\star) \mathrm{d}\theta \\
&= (\nabla^2 f(x^t))^{-1} (\nabla^2 f(x^t))(x^t - x^\star) \\
&\qquad - (\nabla^2 f(x^t))^{-1} \int_0^1 \nabla^2 f(x^\star + \theta(x^t - x^\star))(x^t - x^\star) \mathrm{d}\theta \\
&= (\nabla^2 f(x^t))^{-1} G^t (x^t - x^\star),
\end{aligned}$$

where $G^t = \int_0^1 (\nabla^2 f(x^t) - \nabla^2 f(x^\star + \theta(x^t - x^\star)) \mathrm{d}\theta$. The operator norm of G^t is controlled:

$$\begin{aligned}
\left\|G^t\right\|_{\mathrm{op}} &\leq \int_0^1 \left\|\nabla^2 f(x^t) - \nabla^2 f(x^\star + \theta(x^t - x^\star)\right\|_{\mathrm{op}} \mathrm{d}\theta \\
&\leq \gamma \int_0^1 (1 - \theta) \|x^t - x^\star\| \mathrm{d}\theta \leq \frac{\gamma}{2} \|x^t - x^\star\|.
\end{aligned}$$

Therefore, we obtain

$$\|x^{t+1} - x^\star\| \leq \left\|(\nabla^2 f(x^t))^{-1}\right\|_{\text{op}} \frac{\gamma}{2}\|x^t - x^\star\|^2 \leq \frac{\gamma}{2\alpha}\|x^t - x^\star\|^2. \tag{2.27}$$

To iterate this relation, remark that it can be written as

$$c\|x^{t+1} - x^\star\| \leq (c\|x^{t+1} - x^\star\|)^2, \tag{2.28}$$

where $c = \frac{\gamma}{2\alpha}$. This yields

$$c\|x^T - x^\star\| \leq (c\|x^0 - x^\star\|)^{2^T} \leq 2^{-2^T}, \tag{2.29}$$

where we use the condition $\|x^0 - x^\star\| \leq \alpha/\gamma$. Eventually, β-smoothness in $x^\star$ implies that

$$\begin{aligned} f(x^T) - f(x^\star) &\leq \langle \nabla f(x^\star), x^T - x^\star\rangle + \frac{\beta}{2}\|x^T - x^\star\|^2 \\ &= \frac{\beta}{2}\|x^T - x^\star\|^2 \leq \frac{\beta\alpha}{\gamma} 2^{-2^{T+1}}. \end{aligned} \tag{2.30}$$

□

Newton's method converges much faster than gradient descent. We refer to this behavior as a quadratic convergence (because (2.25) states that each iterate is quadratically closer to the minimum than the previous one). Only $O(\frac{\log\log\varepsilon^{-1}}{\beta\alpha/\gamma})$ iterations are needed to obtain an error ε. In the empirical risk minimization context with n observations, this translates to roughly $\log\log n$ iterations. However, each iteration requires to compute the inverse of the Hessian matrix $\nabla^2 f(x)$. In dimension d, this takes $O(d^3)$ operations using Gauss-Jordan elimination. If d is large (and it is in many applications!), then this cost is prohibitive. Note however that methods that are *much* smarter than Gauss-Jordan elimination are used in practice to compute one step of Newton's method. Another drawback of Newton's method is that it will totally break down should f not be strongly convex. On the opposite, even without strong convexity, we still have convergence guarantees (although slower) for gradient descent (Proposition 2.3.3).

2.5 Logistic regression

Recall the setting of Section 2.1. We consider the classification problem, and consider classifiers of the form $x \mapsto \text{sgn}(g(x))$ where $g : \mathcal{X} \to \mathbb{R}$ is some

function. The loss that appears in this context is given by $\ell_{01}(g(x), y) = \mathbf{1}\{g(x)y < 0\}$. Given n observations $(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})$, the empirical risk is equal to

$$g \mapsto \mathcal{R}_n(g) = \frac{1}{n} \sum_{i=1}^{n} \ell_{01}(g(\mathbf{x_i}), \mathbf{y_i}). \tag{2.31}$$

A powerful method to find a good classifier g consists in replacing the ℓ_{01} loss by a convex loss function that, hopefully, will lead to similar behaviors. There are a lot of different choices that can act as a surrogate for the ℓ_{01} loss. A popular one is the logistic loss

$$\ell_{\log}(y, y') = \log(1 + \exp(-yy')) = -\log(\sigma(yy')), \tag{2.32}$$

where σ is the **sigmoid** function

$$\sigma : t \mapsto \frac{1}{1 + \exp(-t)} \in [0, 1], \tag{2.33}$$

see Figure 2.5. Note that σ satisfies $\sigma(-t) = 1 - \sigma(t)$. The corresponding empirical risk is

$$\begin{aligned} g \mapsto \tilde{\mathcal{R}}_n(g) &= \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-\mathbf{y_i} g(\mathbf{x_i}))) \\ &= \frac{1}{n} \sum_{i=1}^{n} \mathbf{z_i} \log(1 + \exp(-g(\mathbf{x_i}))) \\ &\qquad + (1 - \mathbf{z_i}) \log(1 + \exp(g(\mathbf{x_i}))), \end{aligned} \tag{2.34}$$

where $\mathbf{z_i} = \mathbf{1}\{\mathbf{y_i} = 1\}$. Note that this function is convex in g[1].

Link with maximum likelihood estimation

Let $\mathcal{G}$ be a class of real-valued functions defined on $\mathcal{X}$ (for instance $\mathcal{G}$ is the set of linear functions). We consider the following modelization. Assume that there exists $g \in \mathcal{G}$ such that $(\mathbf{x}, \mathbf{y})$ is obtained by sampling $\mathbf{x}$ according to $P_{\mathbf{x}}$, and then letting $\mathbf{y} = 1$ with probability $\sigma(g(\mathbf{x}))$, and $\mathbf{y} = -1$ otherwise.

[1]We have defined what it means to be convex for functions defined on $\mathbb{R}^d$. However, the definition (2.5) can be used as a definition for a convex functionial defined on a space of functions.

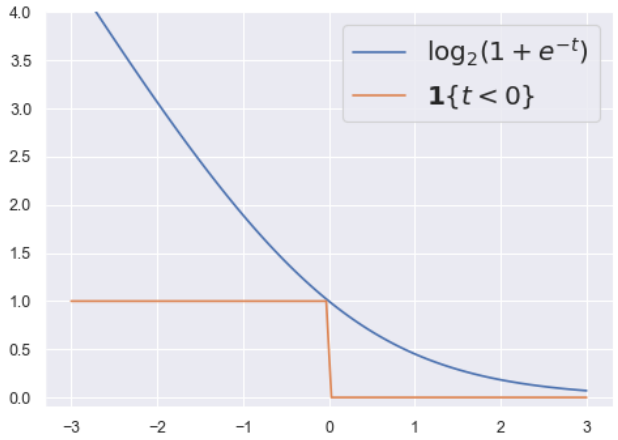


Figure 2.6: The logistic loss is a convexification of the $0-1$ loss.

The likelihood of a set of observations $(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})$ at $g \in \mathcal{G}$ is given by

$$\prod_{i=1}^{n} \sigma(g(\mathbf{x_i}))^{\mathbf{z_i}} (1 - \sigma(g(\mathbf{x_i})))^{1-\mathbf{z_i}}, \tag{2.35}$$

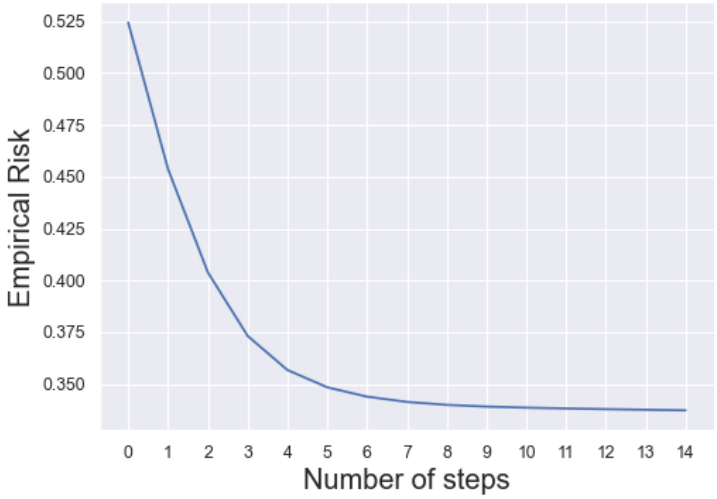
where once again $\mathbf{z_i} = \mathbf{1}\{\mathbf{y_i} = 1\}$. The log-likelihood is equal to

$$\begin{aligned}
&\sum_{i=1}^{n} \mathbf{z_i} \log(\sigma(g(\mathbf{x_i}))) + (1 - \mathbf{z_i}) \log(1 - \sigma(g(\mathbf{x_i}))) \\
&= \sum_{i=1}^{n} \mathbf{z_i} \log(\sigma(g(\mathbf{x_i}))) + (1 - \mathbf{z_i}) \log(\sigma(-g(\mathbf{x_i}))) \\
&= -\sum_{i=1}^{n} \mathbf{z_i} \log(1 + \exp(-g(\mathbf{x_i}))) + (1 - \mathbf{z_i}) \log(1 + \exp(g(\mathbf{x_i}))) \\
&= -\tilde{\mathcal{R}}_n(g).
\end{aligned} \tag{2.36}$$

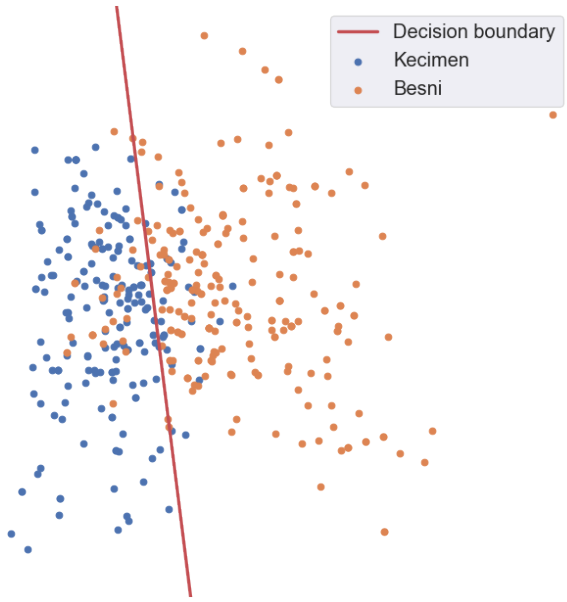
Therefore, maximizing the log-likelihood is equivalent to minimizing the empirical risk $\tilde{\mathcal{R}}_n$. In particular, the empirical risk minimizer $\hat{g}_{\mathcal{G}}$ is a maximum likelihood estimator! Maximum likelihood esitmators are known to satisfy strong theoretical properties (consistency, asymptotic normality, etc.), justifying the use of the logistic loss as a surrogate for the $0-1$ loss.

Example 2.5.1. In [Çinar et al., 2020], the authors use image processing techniques to extract eight relevant geometric features from two different variety

of raisins (Kecimen and Besni). Each grain is then described by a vector in $\mathbb{R}^d$ for $d = 8$ corresponding to those different features (area, perimeter, eccentricity, etc.). We consider the set of affine classifiers $\mathcal{G}_{\text{aff}}$ containing functions g of the form $x \mapsto \theta^\top x + b$ for some $\theta \in \mathbb{R}^d$ and $b \in \mathbb{R}$. Given $n = 450$ grains, we implement a logistic regression to classify the grains into the two varieties. Ten steps of gradient descent are enough to reach the minimizer of the empirical risk $\tilde{R}_n$. We display in Figure 2.7 the classifier that was obtained (in the plane given by a PCA on the dataset). The accuracy of the classifier is tested on a test dataset, and is equal to 85%.



(a)



(b)

Figure 2.7: (a) Empirical risk at different steps of the gradient descent in the logistic regression model with linear classifiers. (b) Decision boundary of the linear classifier, in the plane given by the two principal components of the dataset.

Chapter 3

Stochastic convex optimization

We presented in Chapter 1 the risk minimization paradigm. Let P be a probability distribution on $\mathcal{X} \times \mathcal{Y}$ (where $\mathcal{X}$ is the set of inputs and $\mathcal{Y}$ is the set of outputs). Given a loss function $\ell : \mathcal{Y} \times \mathcal{Y}$, we defined the P-risk of a predictor $f : \mathcal{X} \times \mathcal{Y}$ as the quantity $\mathcal{R}_P(f) = \mathbb{E}_P[\ell(f(\mathbf{x}), \mathbf{y})]$. The Bayes predictor is defined as the predictor $f_P^\star$ minimizing the P-risk, with $\mathcal{R}_P^\star = \mathcal{R}_P(f_P^\star)$. To approach the Bayes predictor, we introduce a class of predictors $\mathcal{F} = \{f_\theta : \mathcal{X} \to \mathcal{Y},\ \theta \in \Theta\}$ indexed by some convex subset $\Theta \subset \mathbb{R}^d$ and consider the minimizer $\theta^\star$ of the function $\theta \in \Theta \mapsto \mathcal{R}_P(f_\theta)$. Note that in practice, we do not have access to the function $\mathcal{R}_P(f_\theta)$ (as P is unknown), so that computing $\theta^\star$ is not an option.

In Chapter 1, we proposed to approximate $\theta^\star$ by computing the minimum of the empirical risk $\mathcal{R}_n$ on the class of predictors $\mathcal{F}$. We here propose a slightly different perspective to achieve this same goal. What if we tried to apply the gradient descent algorithm presented in Chapter 2 to the function $F : \theta \mapsto \mathcal{R}_P(f_\theta)$? To do so, we only need to have access to the gradient of F, which is given at $\theta \in \Theta$ by

$$\nabla F(\theta) = \mathbb{E}_P[\nabla_\theta \ell(f_\theta(\mathbf{x}), \mathbf{y})]. \tag{3.1}$$

Of course, as we do not have access to P, we cannot compute this gradient. However, an unbiased estimator of this gradient is given by $\nabla_\theta \ell(f_\theta(\mathbf{x_i}), \mathbf{y_i})$, where $(\mathbf{x_i}, \mathbf{y_i})$ is an observation with distribution P. Running gradient descent with those approximations of the gradient is referred to as **stochas-**

tic gradient descent. We explore in this chapter the performance of this method, and compare it with gradient descent applied to the empirical risk $\mathcal{R}_n$.

3.1 Stochastic gradient descent

We consider the more general setting where the goal is to minimize a function $F : \Theta \subset \mathbb{R}^d \to \mathbb{R}$. Stochastic gradient descent will iteratively compute a sequence of (random) iterates θ^t, based on (random) approximations of the gradients $\nabla F(\theta^t)$. More precisely, we assume that for every $t \geq 0$, we have access to a random vector $\mathbf{v^t}$ that satisfies $\mathbb{E}[\mathbf{v^t}|\theta^t] = \nabla F(\theta^t)$.

Algorithm 1: Stochastic gradient descent

Initialization: list of step sizes $(s_t)_{t=1,\dots,T-1}$, $\theta^1 \in \Theta$;
for $t = 1, \dots, T-1$ **do**
  draw $\mathbf{v^t}$ such that $\mathbb{E}[\mathbf{v^t}|\theta^t] = \nabla F(\theta^t)$;
  let $\theta^{t+1} = \theta^t - s_t \mathbf{v^t}$;
end
Output: $\overline{\theta} = \frac{1}{T} = \sum_{t=1}^{T} \theta^t$;

This general framework might appear quite abstract, so let us give directly the application we have in mind in this chapter. Let $F(\theta) = \mathbb{E}_P[\ell(f_\theta(\mathbf{x}), \mathbf{y})]$, so that $\nabla F(\theta) = \mathbb{E}_P[\nabla_\theta \ell(f_\theta(\mathbf{x}), \mathbf{y})]$. Assume that we have access to a sample of T i.i.d. samples $(\mathbf{x_1}, \mathbf{y_1}), \dots, (\mathbf{x_{T-1}}, \mathbf{y_{T-1}})$ from the distribution P. In this case, we define $\mathbf{v^t} = \nabla_\theta \ell(f_{\theta^t}(\mathbf{x_t}), \mathbf{y_t})$. As the vector θ^t only depends on the observation $(\mathbf{x_j}, \mathbf{y_j})$ for $j < t$, $\mathbf{v^t}$ is independent from θ^t, ensuring that

$$\mathbb{E}[\mathbf{v^t}|\theta^t] = \mathbb{E}_P[\nabla_{\theta^t} \ell(f_\theta(\mathbf{x}), \mathbf{y})] = \nabla F(\theta^t). \tag{3.2}$$

Example 3.1.1. Another set of examples where stochastic gradient descent can be utilized is when we are looking for the minimum of a function F of the form $\theta \mapsto \frac{1}{n} \sum_{i=1}^{n} F_i(\theta)$, where the functions $F_i : \Theta \to \mathbb{R}$ are arbitrary functions. There is nothing random in the definition of the function F. We may however define a uniform random variable $\mathbf{i}$ on the set of indexes $\{1, \dots, n\}$, and remark that one can express F as

$$F(\theta) = \mathbb{E}[F_{\mathbf{i}}(\theta)]. \tag{3.3}$$

In this situation, one can obtain unbiased estimates of the gradients, by letting $\mathbf{i_1}, \dots, \mathbf{i_{T-1}}$ be T i.i.d. uniform random indexes and then by defining $\mathbf{v^t} = \nabla F_{\mathbf{i_t}}(\theta^t)$.

We are now ready to state our first theorem: stochastic gradient descent converges as long as the function F is convex, Lipschitz continuous, and that the estimates $\mathbf{v^t}$ of the gradients are bounded.

Theorem 3.1.2. *Let $B(0;R)$ be the open ball centered at 0 in $\mathbb{R}^d$. Let $F : \mathbb{R}^d \to \mathbb{R}$ be a convex differentiable function, with minimizer $\theta^\star \in B(0;R)$. Let $T \geq 1$ be an integer and assume that for every $t = 1, \dots, T-1$, it holds that $\mathbb{E}[\|\mathbf{v^t}\|^2] \leq \rho^2$ for some constant $\rho > 0$. Consider stochastic gradient descent with constant step size $s = \sqrt{\frac{4R^2}{\rho^2 T}}$ and initialization $\theta^0 \in B(0;R)$. Then, the output $\overline{\theta}$ of stochastic gradient descent after T steps satisfies*

$$\mathbb{E}[F(\overline{\theta})] - F(\theta^\star) \leq 2\frac{R\rho}{\sqrt{T}}. \tag{3.4}$$

First proof of Theorem 3.1.2: without randomness. We first give a proof of Theorem 3.1.2 in the case where we always have $v^t = \nabla F(\theta^t)$. Note that in this case, the algorithm boils down to classical gradient descent, where we use the average of the iterates as our final approximation. To insist on the non-randomness of the method, we write v^t instead of $\mathbf{v^t}$. We can first apply Jensen's inequality: it holds that

$$F(\overline{\theta}) \leq \frac{1}{T}\sum_{t=1}^{T} F(\theta^t). \tag{3.5}$$

Also, by convexity of F (see Proposition 2.2.3), we have

$$F(\theta^t) - F(\theta^\star) \leq \langle v^t, \theta^t - \theta^\star \rangle \tag{3.6}$$

(remember that $v^t = \nabla F(\theta^t)$ by assumption in this simplified setting). Putting those two equation together yields that

$$F(\overline{\theta}) - F(\theta^\star) \leq \frac{1}{T}\sum_{t=1}^{T} \langle v^t, \theta^t - \theta^\star \rangle. \tag{3.7}$$

To bound the sum in (3.7), we are going to make a telescopic sum appear. To do so, we use the general identity

$$\langle a, b \rangle = \frac{1}{2}(\|a+b\|^2 - \|a\|^2 - \|b\|^2) \tag{3.8}$$

with $a = \theta^\star - \theta^t$, $b = sv^t$. This yields

$$\begin{aligned}\langle v^t, \theta^t - \theta^\star\rangle &= -\frac{1}{s}\langle sv^t, \theta^\star - \theta^t\rangle \\ &= -\frac{1}{2s}(\|\theta^\star - \theta^t + sv^t\|^2 - \|\theta^t - \theta^\star\|^2 - s^2\|v^t\|^2) \\ &= -\frac{1}{2s}(\|\theta^{t+1} - \theta^\star\|^2 - \|\theta^t - \theta^\star\|^2 - s^2\|v^t\|^2) \\ &= \frac{1}{2s}(\|\theta^t - \theta^\star\|^2 - \|\theta^{t+1} - \theta^\star\|^2) + \frac{s}{2}\|v^t\|^2. \end{aligned} \tag{3.9}$$

By summing over t, we obtain from (3.7) that

$$F(\bar{\theta}) - F(\theta^\star) \leq \frac{1}{2Ts}(\|\theta^0 - \theta^\star\|^2 - \|\theta^T - \theta^\star\|^2) + \frac{s}{2T}\sum_{t=1}^{T}\|v^t\|^2. \tag{3.10}$$

To conclude, we use that both θ^0 and $\theta^\star$ belong to R, and that all the gradients v^t have a norm smaller than ρ:

$$F(\bar{\theta}) - F(\theta^\star) \leq \frac{(2R)^2}{2Ts} + \frac{s\rho^2}{2}. \tag{3.11}$$

One obtains the conclusion by plugging in the value $s = \sqrt{\frac{4R^2}{\rho^2 T}}$. □

Second proof of Theorem 3.1.2: general case. In the general case, we adopt the same proof technique, but have to be careful when taking expectations. First, note that as before, by Jensen inequality,

$$\begin{aligned} F(\bar{\theta}) - F(\theta^\star) &\leq \frac{1}{T}\sum_{t=1}^{T}\langle G(\theta^t), \theta^t - \theta^\star\rangle \\ &= \frac{1}{T}\sum_{t=1}^{T}\langle \mathbb{E}[\mathbf{v^t}|\theta^t], \theta^t - \theta^\star\rangle \\ &= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\langle \mathbf{v^t}, \theta^t - \theta^\star\rangle|\theta^t]. \end{aligned} \tag{3.12}$$

Therefore, by using the law of total expectation,

$$\begin{aligned}\mathbb{E}[F(\bar{\theta})] - F(\theta^\star) &\leq \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\langle \mathbf{v^t}, \theta^t - \theta^\star\rangle|\theta^t]\right] \\ &= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\langle \mathbf{v^t}, \theta^t - \theta^\star\rangle] \\ &= \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\langle \mathbf{v^t}, \theta^t - \theta^\star\rangle\right].\end{aligned} \tag{3.13}$$

As before, it holds that

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\langle \mathbf{v^t}, \theta^t - \theta^\star\rangle &= \frac{1}{2Ts}(\|\theta^0 - \theta^\star\|^2 - \|\theta^T - \theta^\star\|^2) + \frac{s}{2T}\sum_{t=1}^{T}\|\mathbf{v^t}\|^2 \\ &\leq \frac{(2R)^2}{2Ts} + \frac{s}{2T}\sum_{t=1}^{T}\|\mathbf{v^t}\|^2.\end{aligned} \tag{3.14}$$

By putting (3.13) and (3.14) together, we obtain that

$$\begin{aligned}\mathbb{E}[F(\bar{\theta})] - F(\theta^\star) &\leq \frac{(2R)^2}{2Ts} + \frac{s}{2T}\sum_{t=1}^{T}\mathbb{E}[\|\mathbf{v^t}\|^2] \\ &\leq \frac{(2R)^2}{2Ts} + \frac{s}{2T}\sum_{t=1}^{T}\rho^2 \\ &\leq \frac{(2R)^2}{2Ts} + \frac{s\rho^2}{2}.\end{aligned}$$

The conclusion is obtained as before by choosing $s = \sqrt{\frac{4R^2}{\rho^2 T}}$. □

Remark 3.1.3.

1. The randomness in (3.4) comes from the randomness in the estimates $\mathbf{v^t}$ of the gradients. In particular, we want to insist on the fact that the function F is not random here.

2. As a particular case of this theorem, we may consider the case where $\mathbf{v^t} = \nabla F(\theta^t)$ (exact gradients). This exactly corresponds to classical gradient descent with the final output being equal to the average of the iterates θ_t.

3. In practice, it is common to discard the first iterates when computing the average $\overline{\theta}$. This allows one to "forget" about the initialization θ^1 (that has no reason to be relevant).

Example 3.1.4 (Toy example). Consider the function $F : \mathbb{R}^2 \to \mathbb{R}$ defined by $F(\theta) = \frac{a_1}{2}\theta_1^2 + \frac{a_2}{2}\theta_2^2$. If we have access to the gradients of F, then iterates of the gradient descent will linearly converge to 0 according to the results from Chapter 2. If, on the opposites, we do not have access to $\nabla F(\theta)$, but to a corrupted version $\mathbf{v} = \nabla F(\theta) + \mathbf{u}$ where $\mathbf{u}$ is a bounded random variable, then we can implement stochastic gradient descent. The iterates of the stochastic gradient descent are displayed in Figure 3.1.

The rate of convergence in this theorem is of order $1/\sqrt{T}$. In the previous chapter (Proposition 2.3.3), we showed that if F is β-smooth (that is the gradient of F is β-Lipschitz), then we can have a rate of convergence of order $1/T$. It is natural to wonder if this faster rate also holds for stochastic gradient descent. It turns out that assuming smoothness does not improve the $1/\sqrt{T}$ rate of convergence here. However, assuming that the function is α-strongly convex is enough to obtain this $1/T$-rate of convergence. To do so, we use a variant of the previous stochastic gradient algorithm where we use an additional projection step to ensure that the different iterates θ^t do not blow up. For $R > 0$, we let

$$\mathrm{proj}_R(\theta) = \begin{cases} \theta & \text{if } \|\theta\| \leq R \\ R\frac{\theta}{\|\theta\|} & \text{otherwise,} \end{cases} \tag{3.15}$$

see also Figure 3.2.

Algorithm 2: Stochastic gradient descent with projection step

Initialization: list of step sizes $(s_t)_{t=1,\dots,T-1}$, $\theta^1 \in \Theta$, radius $R > 0$;
for $t = 1, \dots, T-1$ **do**
  draw $\mathbf{v^t}$ such that $\mathbb{E}[\mathbf{v^t}|\theta^t] = G(\theta^t)$;
  let $\theta^{t+1} = \mathrm{proj}_R(\theta^t - s_t\mathbf{v^t})$;
end
Output: $\overline{\theta} = \frac{1}{T} = \sum_{t=1}^{T} \theta^t$;

Theorem 3.1.5. *Let $F : \mathbb{R}^d \to \mathbb{R}$ be a α-strongly convex differentiable function, with minimizer $\theta^\star \in B(0; R)$. Let $T \geq 1$ be an integer and assume*

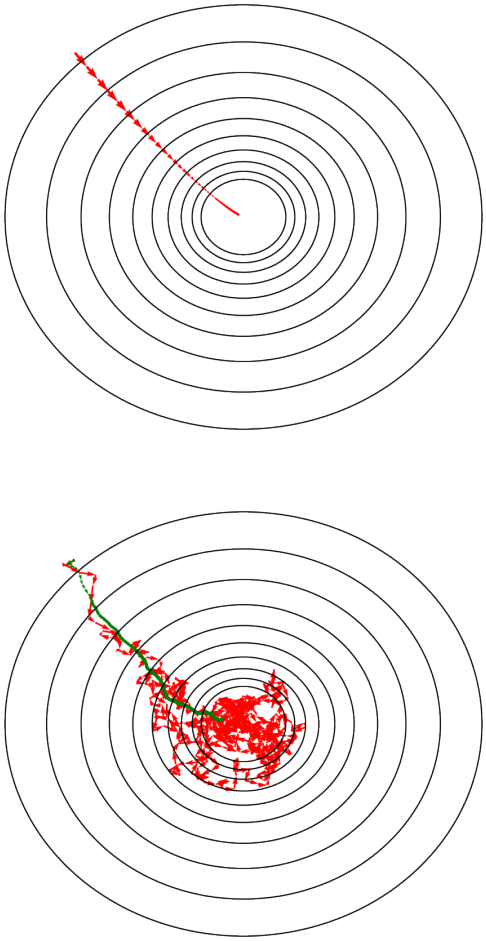


Figure 3.1: Top: iterates of the gradient descent. Bottom: iterates of the stochastic gradient descent (red), and average of the first iterates (green).

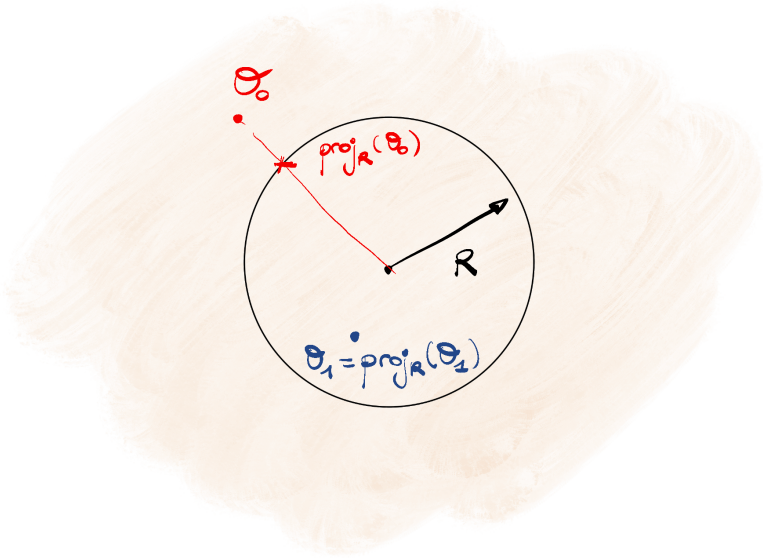


Figure 3.2: The proj_R function.

that for every $t = 1, \ldots, T-1$, *it holds that* $\mathbb{E}[\|\mathbf{v}^t\|^2] \leq \rho^2$ *for some constant* $\rho > 0$. *Consider stochastic gradient descent with projection step, with step size* $s_t = 1/(\alpha t)$ *and initialization* θ^1. *Then, the output* $\overline{\theta}$ *of stochastic gradient descent after* T *steps satisfies*

$$\mathbb{E}[F(\overline{\theta})] - F(\theta^\star) \leq \frac{\rho^2}{2\alpha T}(1 + \log(T)). \tag{3.16}$$

Proof. For sake of simplicity, we prove the result in the case $R = \infty$ (that is without the projection step). Our starting point is once again the identity (3.9), that states that

$$\begin{aligned} \langle \mathbf{v}^t, \theta^t - \theta^\star \rangle &= \frac{1}{2s_t}(\|\theta^t - \theta^\star\|^2 - \|\theta^{t+1} - \theta^\star\|^2) + \frac{s_t}{2}\|\mathbf{v}^t\|^2 \\ &= \frac{\alpha t}{2}(\|\theta^t - \theta^\star\|^2 - \|\theta^{t+1} - \theta^\star\|^2) + \frac{1}{2\alpha t}\|\mathbf{v}^t\|^2. \end{aligned} \tag{3.17}$$

We then use strong convexity (see Proposition 2.9), which implies that for

every $t \geq 1$,

$$\begin{aligned} F(\theta^t) - F(\theta^\star) &\leq \langle \nabla F(\theta^t), \theta^t - \theta^\star \rangle - \frac{\alpha}{2}\|\theta^t - \theta^\star\|^2 \\ &= \langle \mathbb{E}[\mathbf{v^t}|\theta^t], \theta^t - \theta^\star \rangle - \frac{\alpha}{2}\|\theta^t - \theta^\star\|^2 \\ &= \mathbb{E}\Big[\langle \mathbf{v^t}, \theta^t - \theta^\star \rangle - \frac{\alpha}{2}\|\theta^t - \theta^\star\|^2 \Big| \theta^t\Big] \\ &= \mathbb{E}\Big[\frac{\alpha(t-1)}{2}\|\theta^t - \theta^\star\|^2 - \frac{\alpha t}{2}\|\theta^{t+1} - \theta^\star\|^2 + \frac{1}{2\alpha t}\|\mathbf{v^t}\|^2 \Big| \theta^t\Big]. \end{aligned} \tag{3.18}$$

The next steps are now similar to the previous proof. We first apply Jensen's inequality to obtain that $F(\overline{\theta}) - F(\theta^\star) \leq \frac{1}{T}\sum_{t=1}^T (F(\theta^t) - F(\theta^\star))$. Summing the inequality (3.18) for $t = 1, \ldots, T-1$, we obtain that

$$\begin{aligned} \mathbb{E}[F(\overline{\theta}) - F(\theta^\star)] &\leq \frac{1}{T}\sum_{t=1}^T \mathbb{E}[F(\theta^t) - F(\theta^\star)] \\ &\leq \frac{1}{T}\sum_{t=1}^T \mathbb{E}[F(\theta^t) - F(\theta^\star)] \\ &\leq \frac{1}{T}\sum_{t=1}^T \mathbb{E}\Big[\mathbb{E}\Big[\frac{\alpha(t-1)}{2}\|\theta^t - \theta^\star\|^2 - \frac{\alpha t}{2}\|\theta^{t+1} - \theta^\star\|^2 + \frac{1}{2\alpha t}\|\mathbf{v^t}\|^2 \Big| \theta^t\Big]\Big] \\ &\leq \frac{1}{T}\mathbb{E}\Big[\sum_{t=1}^T \frac{\alpha(t-1)}{2}\|\theta^t - \theta^\star\|^2 - \frac{\alpha t}{2}\|\theta^{t+1} - \theta^\star\|^2 + \frac{1}{2\alpha t}\|\mathbf{v^t}\|^2\Big] \\ &\leq \mathbb{E}\Big[-\frac{\alpha T}{2T}\|\theta^T - \theta^\star\|^2 + \frac{1}{T}\sum_{t=1}^T \frac{1}{2\alpha t}\|\mathbf{v^t}\|^2\Big] \\ &\leq \frac{1}{T}\sum_{t=1}^T \frac{\rho^2}{2\alpha t} \leq \frac{\rho^2}{2\alpha T}(1 + \log(T)), \end{aligned}$$

concluding the proof. □

Note that one can actually choose $R = +\infty$ in the previous theorem, so that the same result holds without the projection step. However, it is most of the time delicate to ensure that the expected norm of the gradients $\mathbb{E}[\|\mathbf{v^t}\|^2]$ stay bounded without this projection step. For example, for a quadratic function of the form $F(\theta) = \frac{\alpha}{2}\|\theta\|^2$, the norm of the gradient will blow up if $\|\theta\|$ diverges. An alternative method to ensure that the iterates do not blow

up consists in adding a regularization term to the objective function, that is we minimize $F(\theta) + \lambda\|\theta\|^2$ for some $\lambda > 0$ instead. See Theorem 5.5 in [Bach, 2022] for details.

3.2 Application to risk minimization

We now review how those theorems translate in the setting of risk minimization. Let $F(\theta) = \mathcal{R}_P(f_\theta) = \mathbb{E}_P[\ell(f_\theta(\mathbf{x}), \mathbf{y})]$ be the P-risk of some predictor f_θ indexed by $\theta \in \mathbb{R}^k$, with minimizer $\theta^\star$. Assume that we have access to n i.i.d. samples $(\mathbf{x_1}, \mathbf{y_1}), \dots, (\mathbf{x_n}, \mathbf{y_n})$ from distribution P. We compare two methods:

- (SGD) Let $\hat{\theta}_{\text{SGD}}$ be the output Stochastic gradient descent (with projection) with n steps using the gradient estimates $\nabla_\theta \ell(f_\theta(\mathbf{x_i}), \mathbf{y_i})$.
- (GD-ER) Let $\hat{\theta}_T$ be the output of gradient descent applied for T steps on the empirical risk

$$\theta \mapsto \mathcal{R}_n(f_\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(\mathbf{x_i}), \mathbf{y_i}). \tag{3.19}$$

Note that for any of those predictors $\hat{\theta}$, it holds that the expected excess of risk can be decomposed into

$$\mathbb{E}[\mathcal{R}_P(f_{\hat{\theta}}) - \mathcal{R}_P^\star] = \underbrace{\mathbb{E}[\mathcal{R}_P(f_{\hat{\theta}}) - \mathcal{R}_P(f_{\theta^\star})]}_{\text{optimization error}} + \underbrace{\mathcal{R}_P(f_{\theta^\star}) - \mathcal{R}_P^\star}_{\text{approximation error}}. \tag{3.20}$$

As explained in Chapter 1, the approximation error will depend only the "size" of the set of predictors $\mathcal{F} = \{f_\theta,\ \theta \in \mathbb{R}^k\}$. On the contrary, the optimization error will depend on our method to find the minimum of F. We consider two questions.

- (Q1) What is the minimal number n of samples required to get an optimization error smaller than ε using (SGD) or (GD-ER)?
- (Q2) What is the associated time complexity of the algorithm?

For sake of conciseness, we will only answer (Q1) and (Q2) in the "favorable" case where, for every $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the function $\theta \in \mathbb{R}^k \mapsto \ell(f_\theta(x), y)$ is α-strongly convex and β-smooth. We further assume that the minimizer $\theta^\star$ belong to $B(0; R)$ with R of the form $c/\sqrt{\beta}$. One can try as an exercise to play to the same game by removing for instance the α-strongly convex assumption. We let $\kappa = \beta/\alpha \geq 1$ be the condition number. Also, we use the notation $\tilde{O}(\varepsilon^a)$ to denote a quantity of the form $\varepsilon^a(\log(\varepsilon)^{-1})^b$. This allows us to hide logarithmic factors that are almost constant in practice.

Let us first consider (SGD). In this setting, every gradient $\mathbf{v^t}$ is of the form $\nabla_\theta \ell_{\theta^t}(f_\theta(\mathbf{x_i}), \mathbf{y_i})$. By definition of β-smoothness (Definition 2.2.7), the norm of this gradient is smaller than

$$\beta\|\theta^t - \theta^\star\| \leq 2\beta R = \rho.$$

According to Theorem 3.1.5 with $n = T$, we need $n = \tilde{O}(\beta^2 R^2/(\alpha\varepsilon))$ samples to reach a precision ε. Evaluating the gradient $\nabla\ell(f_\theta(\mathbf{x_i}), \mathbf{y_i})$ requires $O(k)$ operations, so that the time complexity of SGD is $\tilde{O}(k\beta^2 R^2/(\alpha\varepsilon))$. Recalling that we assume that $R^2 \leq c^2/\beta$, we obtain a time complexity of order

$$\tilde{O}(k\kappa/\varepsilon).$$

The analysis of (GD-ER) is slightly more complicated. Let $\hat{\theta}_\infty$ be the minimizer of $\theta \mapsto \mathcal{R}_n(\theta)$ (that is the actual empirical risk minimizer). One can further bound the optimization error:

$$\begin{aligned}
\mathcal{R}_P(f_{\hat{\theta}_T}) - \mathcal{R}_P(f_{\theta^\star}) &\leq \mathcal{R}_P(f_{\hat{\theta}_T}) - \mathcal{R}_n(f_{\hat{\theta}_T}) \\
&\quad + \mathcal{R}_n(f_{\hat{\theta}_T}) - \mathcal{R}_n(f_{\hat{\theta}_\infty}) \\
&\quad + \mathcal{R}_n(f_{\hat{\theta}_\infty}) - \mathcal{R}_n(f_{\theta^\star}) \\
&\quad + \mathcal{R}_n(f_{\theta^\star}) - \mathcal{R}_P(f_{\theta^\star}) \\
&\leq 2 \cdot \sup_\theta |\mathcal{R}_n(f_\theta) - \mathcal{R}_P(f_\theta)| + \mathcal{R}_n(f_{\hat{\theta}_T}) - \mathcal{R}_n(f_{\hat{\theta}_\infty})
\end{aligned}$$

where at the last line we use that $\mathcal{R}_n(f_{\hat{\theta}_\infty}) \leq \mathcal{R}_n(f_{\theta^\star})$ by definition of the empirical risk minimizer. The first term in this last inequality can be shown to be at least of order $1/\sqrt{n}$ (this follows from $\mathcal{R}_n(f_\theta)$ being the average of n i.i.d. random variables). In particular, to reach an optimization error $\mathbb{E}[\mathcal{R}_P(f_{\hat{\theta}_T}) - \mathcal{R}_P(f_{\theta^\star})]$ of order ε, we need at least $n = O(\varepsilon^{-2})$ samples. Propostion 2.3.2 asserts that after T steps of gradient descent we have a

control of the form

$$\begin{aligned}\mathcal{R}_n(\theta_T) - \mathcal{R}_n(\theta_\infty) &\leq \exp(-T/\kappa)(\mathcal{R}_n(\theta_0) - \mathcal{R}_n(\theta_\infty)) \\ &\leq \exp(-T/\kappa)\frac{\beta}{2}\|\theta_0 - \theta_\infty\|^2 \\ &\leq \exp(-T/\kappa)2R^2\beta \leq \exp(-T/\kappa)2c^2,\end{aligned} \tag{3.21}$$

where we also use the definition of β-smoothness and the fact that both $\|\theta_0\|$ and $\|\hat{\theta}_\infty\|$ are smaller than R. Therefore, to make this quantity smaller than ε, a number $T = \tilde{O}(\kappa)$ of steps of gradient descent are required. As computing a single gradient $\nabla_\theta \mathcal{R}_n(f_\theta)$ requires to compute n gradients $\nabla_\theta \ell(f_\theta(\mathbf{x_i}), \mathbf{y_i})$, the final complexity of gradient descent is in this situation

$$\tilde{O}(knT) = \tilde{O}(kn\kappa) = \tilde{O}(k/\varepsilon^2\kappa).$$

We summarize the different results in the following table. **Stochastic gradient descent requires less samples and a smaller time complexity to attain a given accuracy.**

Algo.	Num. of samples	Complexity
SGD	$n = \tilde{O}(\kappa/\varepsilon)$	$\tilde{O}(k\kappa/\varepsilon)$
GD	$n = O(1/\varepsilon^2)$	$\tilde{O}(k\kappa/\varepsilon^2)$

Table 3.1: Summary of the convergence rates in the α-strongly convex and β-smooth case.

Chapter 4

Kernel methods

4.1 Linear regression with feature maps

Consider the dataset $\mathbf{x_1}, \ldots, \mathbf{x_n}$ presented in Figure 4. Points inside the ball $B = \{x \in \mathbb{R}^2,\ \|x\| \leq 1\}$ are assigned to -1, whereas points outside B are assigned to $+1$. No linear classifiers will then be able to obtain a good accuracy. A first possibility is to look for more complicated classifiers (composed of intersections of hyperplanes for instance). Another possibility is to "lift" the dataset to a higher dimensional space by considering the map $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ defined by $\Phi(x) = (x, \|x\|^2)$. The transformed dataset $(\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_n}))$ is displayed in Figure 4. Remark that this transformed dataset, although in higher dimension than the original dataset, is considerably simpler to classify: there now exists a linear classifier in $\mathbb{R}^3$ that will make exactly zero classification errors on the training set. This toy example showcases an important concept: if a dataset has some complex structure, it may be a good idea to look at a transformation of the dataset into a larger space. In the larger space, the transformed dataset has hopefully a simpler structure, and a basic method (for instance a linear regression) will then have a very good performance. We call such a transformation Φ a **feature map**.

Let us consider another, less artificial example of this phenomenon. In polynomial regression, the goal is to fit a polynomial function of degree d to observations $(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})$, with $\mathbf{x_i} \in \mathbb{R}$ and $\mathbf{y_i} \in \mathbb{R}$. Let $\mathcal{F}_d$ be the set of polynomial functions of the form $f_a : x \mapsto \sum_{j=0}^{d} a_j x^j$ where

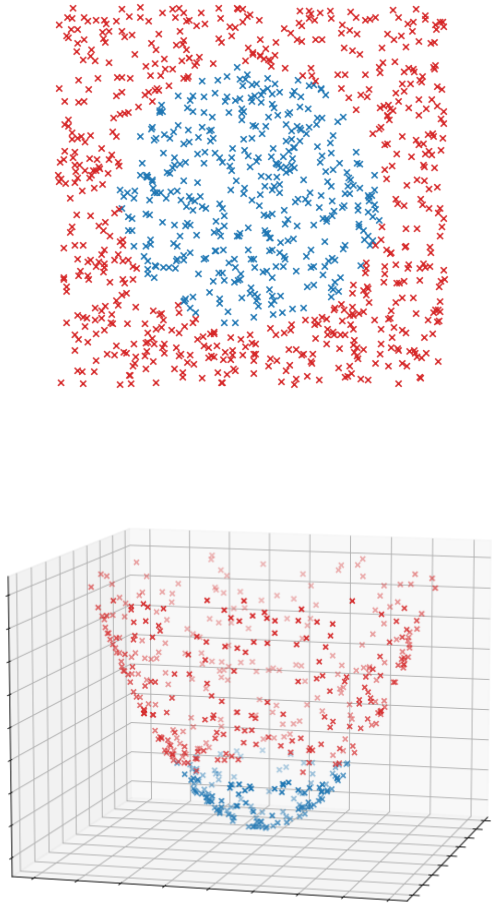


Figure 4.1: Top: We set $\mathbf{y} = -1$ for points $\mathbf{x}$ inside B, and $\mathbf{y} = +1$ otherwise. No linear classifier can attain good accuracy on this dataset. Bottom: In this new representation, there exists a linear classifier with zero risk (given by the equation $z = 1$).

$a = (a_0, \dots, a_d) \in \mathbb{R}^{d+1}$. We are looking for the minimizer of the risk

$$f_a \in \mathcal{F}_d \mapsto \mathcal{R}_n(f_a) := \frac{1}{n} \sum_{i=1}^{n} |\mathbf{y_i} - f_a(\mathbf{x_i})|^2. \tag{4.1}$$

Let $\Phi_d : \mathbb{R} \to \mathbb{R}^{d+1}$ be defined by $\Phi_d(x) = (1, x, x^2, \dots, x^d)$. Then, $f_a(\mathbf{x_i}) = \langle a, \Phi_d(\mathbf{x_i}) \rangle$. It holds that

$$\mathcal{R}_n(f_a) = \frac{1}{n} \sum_{i=1}^{n} |\mathbf{y_i} - \langle a, \Phi_d(\mathbf{x_i}) \rangle|^2 = \frac{1}{n} \|\mathbf{Y} - \tilde{\mathbf{X}} a\|^2,$$

where $\mathbf{Y} = (\mathbf{y_1}, \dots, \mathbf{y_n})^\top \in \mathbb{R}^n$ and $\tilde{\mathbf{X}}$ is the matrix of size $n \times (d+1)$ with ith row given by $\Phi_d(\mathbf{x_i})$. Therefore, minimizing the empirical risk $\mathcal{R}_n$ over $\mathcal{F}_d$ is equivalent to performing a linear regression over the transformed dataset $((\Phi_d(\mathbf{x_1}), \mathbf{y_1}), \dots, (\Phi_d(\mathbf{x_n}), \mathbf{y_n}))$. This is an instance of the same phenomenon: when a linear technique (linear regression) does not perform well, map the dataset in a larger space (here using Φ_d) and apply a linear technique in higher dimension.

Consider now a general feature map $\Phi : \mathcal{X} \to \mathbb{R}^D$, and the associated linear regression problem

$$\frac{1}{n} \sum_{i=1}^{n} |\mathbf{y_i} - \langle a, \Phi(\mathbf{x_i}) \rangle|^2 = \frac{1}{n} \|\mathbf{Y} - \tilde{\mathbf{X}} a\|^2, \tag{4.2}$$

with $\mathbf{Y}$ defined as before and where $\tilde{\mathbf{X}}$ is the matrix of size $n \times D$ with rows given by $\Phi(\mathbf{x_i})$. Assuming that the matrix $\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$ is invertible, the linear regression predictor is given by

$$\hat{a} = (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^\top \mathbf{Y}. \tag{4.3}$$

Computing this solution requires to compute the matrix $(\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}})^{-1}$. In particular, we need to inverse a matrix of size $D \times D$.[1] This is computationally intractable if D is large (naively, it requires $O(D^3)$ operations). Therefore, it might look like we have found a limitation of the "lifting" approach: if we lift our observations in a space of dimension too large, then minimizing becomes a problem too hard to solve. However, there is a trick that allows

[1] If the matrix $\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$ is not invertible, it can be replaced by its so-called pseudo-inverse, so this discussion applies in both regimes where $D \geq n$ and $n \geq D$.

us to bypass this limitation. Indeed, remark that we can always assume that a minimizer $\hat{a}$ of (4.2) is of the form

$$\sum_{l=1}^{n} \hat{b}_l \Phi(\mathbf{x_l}) \tag{4.4}$$

for some coefficients $\hat{b}_l \in \mathbb{R}$. Otherwise, we may project $\hat{a}$ on the vector space spanned by the vectors $\Phi(\mathbf{x_i})$. The projected vector $\tilde{a}$ is such that $\tilde{\mathbf{X}}\hat{a} = \tilde{\mathbf{X}}\tilde{a}$ and is of the form (4.4). Therefore, there are always minimizers of the form (4.4).

This discussion implies that it suffices to look for vectors a of the form $a = \sum_{l=1}^{n} b_l \Phi(\mathbf{x_l})$ when looking for the minimum of (4.2). This is equivalent to minimizing the functional

$$b \in \mathbb{R}^n \mapsto \frac{1}{n} \sum_{i=1}^{n} |\mathbf{y_i} - \sum_{l=1}^{n} b_l \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_l}) \rangle|^2 = \frac{1}{n} \|\mathbf{Y} - \mathbf{G}b\|^2, \tag{4.5}$$

where $\mathbf{G}$ is the Gram matrix of size $n \times n$, defined by $G_{ij} = \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle$. Computing $\mathbf{G}$ requires a number of computations that is only linear in D, and once it is computed, the complexity of the problem will only depend on the number of observations n. Should $\mathbf{G}$ be invertible, the optimal vector $\hat{b}$ is given by

$$\hat{b} = \mathbf{G}^{-1}\mathbf{Y}. \tag{4.6}$$

Let us summarize what we have done so far.

- Instead of looking for a complex predictor on the dataset $(\mathbf{x_1}, \ldots, \mathbf{x_n})$, we lift the observations to a higher dimensional space using a feature map $\Phi : \mathcal{X} \to \mathbb{R}^D$.
- In this higher dimensional space, we then look for a simple prediction (e.g. a linear regression, a ridge regression, etc.).
- If the feature space $\mathbb{R}^D$ is very large, then "naive" computations become intractable. However, it turns out that, once the Gram matrix $\mathbf{G}$ of dot products $\langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle$ is computed, the complexity of the problem only depends on the number of observations n. Such a remark is referred to as the **kernel trick**.

This last remark is particularly important. It shows that we never need to actually compute the vectors $\Phi(\mathbf{x_i})$, but only the dot products $\mathbf{G}_{ij} = \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle$. For some feature maps Φ, the dimension D is very large (or even infinite!) whereas a very simple expression exists for the dot product, making such an observation particularly appealing.

4.2 Reproducing Kernel Hilbert Spaces

The Gram matrix $\mathbf{G}$ that we introduced in the previous section is a matrix whose entries $\mathbf{G}_{ij} = \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle$ measure the proximity between the observations $\mathbf{x_i}$ and $\mathbf{x_j}$. If $\mathbf{G}_{ij}$ is large, then $\Phi(\mathbf{x_i})$ and $\Phi(\mathbf{x_j})$ are aligned, meaning that $\mathbf{x_i}$ and $\mathbf{x_j}$ are similar in some sense. On the contrary, if $\mathbf{G}_{ij} = 0$, then $\Phi(\mathbf{x_i})$ is orthogonal to $\Phi(\mathbf{x_j})$ and $\mathbf{x_i}$ and $\mathbf{x_j}$ are very different (at least if we believe that Φ captures relevant information on the observations). This suggests the following bold idea: what if we replace the dot product $\langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle$ by a "measure of similarity" $k(\mathbf{x_i}, \mathbf{x_j})$ between $\mathbf{x_i}$ and $\mathbf{x_j}$? For instance, a possible notion of similarity is given by the so-called Gaussian (or RBF) kernel

$$k_\sigma(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \tag{4.7}$$

for $x, x' \in \mathbb{R}^d$. Indeed, if x and x' are close, then $k_\sigma(x, x') = 1$ (high similarity), whereas if x and x' are far away, then $k_\sigma(x, x')$ is very small (low similarity).

It turns out that under mild conditions on the function k, we can show that there exists some feature map Φ with $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$. In particular, the discussion of the previous section holds with this feature map Φ. To implement a linear regression, we only need to compute the Gram matrix $\mathbf{G}$, whose entries are given by

$$\mathbf{G}_{ij} = \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle = k(\mathbf{x_i}, \mathbf{x_j}).$$

Therefore, the feature map Φ never needs to be actually computed! Only the values $k(\mathbf{x_i}, \mathbf{x_j})$ have to be computed.

Proving the existence of this feature map Φ requires the introduction of some mathematical machinery. In particular, Φ will in general have outputs living in a space of infinite dimension, that we call a Hilbert space. A Hilbert space $\mathcal{H}$ is a vector space where we define a notion of dot product that behaves like the classical dot product on $\mathbb{R}^D$. This means the following:

- Symmetry: for all $x, y \in \mathcal{H}$, $\langle x, y\rangle_{\mathcal{H}} = \langle y, x\rangle_{\mathcal{H}}$.
- Linearity: for every vectors $x, y, z \in \mathcal{H}$ and $\lambda, \mu \in \mathbb{R}$, we have $\langle x, \lambda y + \mu z\rangle_{\mathcal{H}} = \lambda\langle x, y\rangle_{\mathcal{H}} + \mu\langle x, z\rangle_{\mathcal{H}}$.
- Positive definiteness: for all $x \in \mathcal{H}$, $\langle x, x\rangle_{\mathcal{H}} \geq 0$, with equality if and only if $x = 0$. We write $\|x\|_{\mathcal{H}}$ for $\sqrt{\langle x, x\rangle_{\mathcal{H}}}$. This defines a **norm** on $\mathcal{H}$.
- Completeness: for every continuous linear map $L : \mathcal{H} \to \mathbb{R}$, there exists a vector $h \in \mathcal{H}$ such that $L(x) = \langle h, x\rangle_{\mathcal{H}}$ for every $x \in \mathcal{H}$.

Thise last property might appear mysterious. One can check that it is always satisfied if $\mathcal{H}$ is a Hilbert space of finite dimension (that is $\mathcal{H} = \mathbb{R}^d$). In infinite dimension, completeness ensures that some common mathematical operations (e.g. taking infinite sums or projecting vectors on subspaces) are well-defined.

Example 4.2.1 (For those interested in math theory). Let us show that in a Hilbert space, infinite sums are well-defined. Let $(h_n)_{n\geq 0}$ be vectors in a Hilbert space $\mathcal{H}$, and assume that the infinite sum of real numbers $\sum_{n\geq 0} \|h_n\|_{\mathcal{H}}$ is finite. Let $L : \mathcal{H} \to \mathbb{R}$ be defined by $L(x) = \sum_{n\geq 0}\langle h_n, x\rangle$. One can check by Cauchy-Schwartz inequality that we have

$$|L(x)| \leq \sum_{n\geq 0} \|h_n\|_{\mathcal{H}}\|x\|_{\mathcal{H}} < \infty.$$

Also, by properties of the sum, L is linear. Therefore, L is a continuous linear map. By completeness, there exists a vector $g \in \mathcal{H}$ such that $L(x) = \langle g, x\rangle$ for every x. By definition, we denote this vector by $\sum_{n\geq 0} h_n$, and this is our definition of the infinite sum! One can then check that with this definition, infinite sums in the Hilbert space $\mathcal{H}$ satisfy the usual properties (e.g. linearity).

An example of vector space $\mathcal{H}$ that is not complete is given by the space of continuous bounded functions on $[0, 1]$, endowed with the dot product $\langle f, g\rangle = \int_0^1 f(t)g(t)\mathrm{d}t$. One can check that the linear map defined by $L(f) = \int_0^{1/2} f(t)\mathrm{d}t$ is continuous. However, there does not exist any continuous function h with

$$L(f) = \int_0^{1/2} f(t)h(t)\mathrm{d}t$$

for every continuous bounded function f. Such a function h should be 1 on $[0, 1/2)$ and 0 on $[1/2, 1]$, and therefore would not be a continuous bounded function. However, the space of L_2 functions on $[0, 1]$ is complete.

Let us fix some set $\mathcal{X}$ and some function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Assume that $k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}$ for some Hilbert space $\mathcal{H}$ and some feature map $\Phi : \mathcal{X} \to \mathcal{H}$. Consider some points $x_1, \ldots, x_n \in \mathcal{X}$ and numbers $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$. Then,

$$\begin{aligned} 0 &\leq \| \sum_{i=1}^{n} \lambda_i \Phi(x_i) \|_{\mathcal{H}}^2 \\ &= \sum_{1 \leq i,j \leq n} \lambda_i \lambda_j \langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{H}} \\ &= \sum_{1 \leq i,j \leq n} \lambda_i \lambda_j k(x_i, x_j). \end{aligned}$$

Therefore, to be represented by a feature map, the function k should at least satisfy that

$$\sum_{1 \leq i,j \leq n} \lambda_i \lambda_j k(x_i, x_j) \geq 0. \tag{4.8}$$

It turns out that this condition is also sufficient.

Definition 4.2.2 (Kernel)**.** *Let $\mathcal{X}$ be a set and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a function. We say that k is a (positive semi-definite) kernel if k is symmetric and if for every $n \in \mathbb{N}$, every $x_1, \ldots, x_n \in \mathcal{X}$ and every $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$, condition (4.8) is satisfied.*

Alternatively, if G is the $n \times n$ matrix with entries $k(x_i, x_j)$, then (4.8) asserts that the matrix G is positive semi-definite, that is $G \succcurlyeq 0$. We are now in position to state our main theorem.

Theorem 4.2.3. *Let k be a kernel on a set $\mathcal{X}$. Then, there exists a Hilbert space $\mathcal{H}$ and a feature map $\Phi : \mathcal{X} \to \mathcal{H}$ such that for every $x, x' \in \mathcal{X}$,*

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}. \tag{4.9}$$

Proof. Let $\mathbb{R}^{\mathcal{X}}$ be the set of functions from $\mathcal{X}$ to $\mathbb{R}$ (which is a vector space). Define a function $\Phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}$ by $\Phi(x) = k(x, \cdot)$. Consider the set $\mathcal{H}_0$

obtained as the set of finite sums of the form $\sum_{i=1}^{n} \lambda_i \Phi(x_i)$ for elements $x_i \in \mathcal{X}$ and $\lambda_i \in \mathbb{R}$. The set $\mathcal{H}_0$ is a vector space, and we can define a dot product on it. Let $f = \sum_{i=1}^{n} \lambda_i \Phi(x_i)$ and $g = \sum_{j=1}^{n'} \lambda'_j \Phi(x'_j)$. The dot product is defined as

$$\langle f, g \rangle_{\mathcal{H}_0} := \sum_{i=1}^{n} \sum_{j=1}^{n'} \lambda_i \lambda'_j k(x_i, x'_j). \tag{4.10}$$

One can check that this expression indeed defines a dot product and also does not depend on the choice of expansions of f and g that we have chosen. Furthermore, we have indeed

$$\langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}_0} = k(x, x') \tag{4.11}$$

for any $x, x' \in \mathcal{H}$. However, the vector space $\mathcal{H}_0$ is not necessarily complete. By using a process called completion, we can actually expand it to a larger space $\mathcal{H}$ that is complete, and that will still satisfy (4.9). □

Let us give some techniques to construct kernels.

Proposition 4.2.4. *Let k_1 and k_2 be kernels on $\mathcal{X}$.*

1. *The sum $k_1 + k_2$ is a kernel.*
2. *The product $k_1 \cdot k_2$ is a kernel.*
3. *If $\mathcal{X} \subset \mathbb{R}^d$ and $k(x, x')$ is of the form $k(x - x')$, then k is a kernel if its Fourier transform*

$$\mathcal{F}[k](\xi) = \int \exp(-2\pi i \langle \xi, x \rangle) k(x) \mathrm{d}x \tag{4.12}$$

is nonnegative for every $\xi \in \mathbb{R}^d$.

Proof. Let $x_1, \ldots, x_n \in \mathcal{X}$ and $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$.

1. We have

$$\begin{aligned} &\sum_{1 \le i,j \le n} \lambda_i \lambda_j (k_1(x_i, x_j) + k_2(x_i, x_j)) \\ &\quad = \sum_{1 \le i,j \le n} \lambda_i \lambda_j k_1(x_i, x_j) + \sum_{1 \le i,j \le n} \lambda_i \lambda_j k_2(x_i, x_j) \ge 0. \end{aligned} \tag{4.13}$$

   Therefore, $k_1 + k_2$ is a kernel.

2. Note that the covariance matrix G of a random variable $\mathbf{y} \in \mathbb{R}^n$ is always positive semi-definite (that is we have $G_{ij} = \mathbb{E}[\mathbf{y^{(i)}}\mathbf{y^{(j)}}]$ where $\mathbf{y} = (\mathbf{y^{(1)}}, \dots, \mathbf{y^{(n)}})$). Reciprocally, if G is a positive semi-definite matrix, then there exists a random variable $\mathbf{y} \in \mathbb{R}^n$ with covariance matrix G (take for instance a Gaussian random variable). Consider a random vector $\mathbf{y_1}$ with covariance matrix $G_1 = (k_1(x_i, x_j))_{ij}$ and a random vector $\mathbf{y_2}$ with covariance matrix $G_2 = (k_2(x_i, x_j))_{ij}$, independent from $\mathbf{y_1}$. Then, one can check that the vector $\mathbf{y} = (\mathbf{y_1^{(1)}}\mathbf{y_2^{(1)}}, \dots, \mathbf{y_1^{(n)}}\mathbf{y_2^{(n)}})$ has covariance matrix G with entries $(k_1(x_i, x_j) \cdot k_2(x_i, x_j))_{ij}$. Therefore, G is positive semi-definite and $k_1 \cdot k_2$ is a kernel.

3. We use the inverse Fourier transform formula

$$k(x) = \int \exp(2\pi i \langle \xi, x \rangle) \mathcal{F}[k](\xi) \mathrm{d}\xi.$$

Therefore,

$$\begin{aligned} \sum_{1 \leq i,j \leq n} \lambda_i \lambda_j k(x_i - x_j) &= \sum_{1 \leq i,j \leq n} \lambda_i \lambda_j \int \exp(2\pi i \langle \xi, x_i - x_j \rangle) \mathcal{F}[k](\xi) \mathrm{d}\xi \\ &= \int \sum_{1 \leq i,j \leq n} \lambda_i \lambda_j \exp(2\pi i \langle \xi, x_i - x_j \rangle) \mathcal{F}[k](\xi) \mathrm{d}\xi \\ &= \int \| \sum_{i=1}^{n} \lambda_i \exp(2\pi i \langle \xi, x_i \rangle) \|^2 \mathcal{F}[k](\xi) \mathrm{d}\xi \geq 0. \end{aligned} \tag{4.14}$$

Therefore, k is a kernel.

□

Let us give some examples.

Example 4.2.5. 1. If Φ is a map from the set $\mathcal{X}$ to a Hilbert space $\mathcal{H}$, then $k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}$ defines a kernel.

2. Polynomial kernels: the function $k(x, x') = \langle x, x' \rangle^\alpha$ for $\alpha \in \mathbb{N}$ is a kernel. This follows by induction from Proposition 4.2.4.2.

3. The radial basis function (RBF) kernel, or gaussian kernel: the function $k_\sigma(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$ defines a kernel. Indeed, the Fourier transform of $x \in \mathbb{R}^d \mapsto \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right)$ is given by

$$\xi \mapsto \sqrt{2\pi\sigma^2}\exp(-2\pi\sigma^2\|\xi\|^2), \tag{4.15}$$

which is nonnegative.

4.3 Kernel Ridge Regression

The general kernel approach can be applied to different algorithms, some that we have already encountered (e.g. PCA or logistic regression) and some that we have not mentionned (kernel SVM, kernel k-means, etc.). We will here present two algorithms that can be "kernelized": kernel ridge regression and kernel PCA. Recall the setting of ridge regression. Let $(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})$ be a training sample, with $\mathbf{x_i} \in \mathbb{R}^d$ and $\mathbf{y_n} \in \mathbb{R}$. Ridge regression consists in minimizing the quantity

$$\beta \in \mathbb{R}^d \mapsto \frac{1}{n}\|\mathbf{X}\beta - \mathbf{Y}\|^2 + \lambda\|\beta\|^2, \tag{4.16}$$

where $\mathbf{X}$ is the $n \times d$ matrix with rows given by the $\mathbf{x_i}$s, $\mathbf{Y} \in \mathbb{R}^n$ is the vector with entries $\mathbf{y_i}$, $\|\beta\|$ is the 2-norm of the vector β and $\lambda > 0$ is a penalization parameter. This function has a unique minimizer which is given by

$$\hat{\beta} = \left(\mathbf{X}^\top\mathbf{X} + n\lambda\mathrm{Id}_d\right)^{-1}\mathbf{X}^\top\mathbf{Y}. \tag{4.17}$$

Let us kernelize this algorithm. We now assume that the inputs $\mathbf{x_1}, \ldots, \mathbf{x_n}$ are elements of an arbitrary set $\mathcal{X}$, and that we are given a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Let $\Phi : \mathcal{X} \to \mathcal{H}$ be an associated feature map, such that $\langle\Phi(x), \Phi(x')\rangle_\mathcal{H} = k(x, x')$ for every $x, x' \in \mathcal{H}$. We consider predictions of the form $x \mapsto \langle\Phi(x), \beta\rangle_\mathcal{H}$ and are looking for a minimizer of the functional

$$\beta \in \mathcal{H} \mapsto \frac{1}{n}\|\tilde{\mathbf{X}}\beta - \mathbf{Y}\|^2 + \lambda\|\beta\|^2_\mathcal{H}, \tag{4.18}$$

where $\tilde{\mathbf{X}}$ is a linear operator from $\mathcal{H}$ to $\mathbb{R}^n$, defined by

$$\tilde{\mathbf{X}}\beta = (\langle\Phi(\mathbf{x_1}), \beta\rangle_\mathcal{H}, \ldots, \langle\Phi(\mathbf{x_n}), \beta\rangle_\mathcal{H}).$$

Note that in (4.18) there are two different norms appearing: the euclidean norm, and the Hilbert norm in the penalization term.

Theorem 4.3.1 (Representer theorem)**.** *The minimum of* (4.18) *is attained at a vector* β *of the form* $\beta = \sum_{i=1}^{n} a_i \Phi(\mathbf{x_i})$ *for some real numbers* $a_1, \dots, a_n$.

Proof. Let E be the subspace of $\mathcal{H}$ that is spanned by the vectors $\Phi(\mathbf{x_i})$ and let $E^\perp$ be the orthogonal subspace of E. We can decompose every $\beta \in \mathcal{H}$ into $\beta_1 + \beta_2$, where $\beta_1 \in E$ and $\beta_2 \in E^\perp$. Note that $\langle \Phi(\mathbf{x_i}), \beta_2 \rangle_{\mathcal{H}} = 0$ for every $\mathbf{x_i}$ by definition of orthogonality. This implies that $\langle \Phi(\mathbf{x_i}), \beta \rangle_{\mathcal{H}} = \langle \Phi(\mathbf{x_i}), \beta_1 \rangle_{\mathcal{H}}$. Also, by orthogonality, it holds that $\|\beta\|_{\mathcal{H}}^2 = \|\beta_1\|_{\mathcal{H}}^2 + \|\beta_2\|_{\mathcal{H}}^2$. This yields

$$\begin{aligned}
&\frac{1}{n}\|\tilde{\mathbf{X}}\beta - \mathbf{Y}\|^2 + \lambda\|\beta\|_{\mathcal{H}}^2 \\
&\quad = \frac{1}{n}\|\tilde{\mathbf{X}}\beta_1 - \mathbf{Y}\|^2 + \lambda\|\beta_1\|_{\mathcal{H}}^2 + \lambda\|\beta_2\|_{\mathcal{H}}^2 \\
&\quad \leq \frac{1}{n}\|\tilde{\mathbf{X}}\beta_1 - \mathbf{Y}\|^2 + \lambda\|\beta_1\|_{\mathcal{H}}^2.
\end{aligned}$$

Therefore, the minimum of (4.18) is attained at a point β with $\beta_2 = 0$, that is at a point $\beta \in E$. □

Thus, to find the minimum of (4.18), it suffices to consider vectors β of the form $\sum_{i=1}^{n} a_i \Phi(\mathbf{x_i})$. It is equivalent to minimize over all $a = (a_1, \dots, a_n) \in \mathbb{R}^n$ the quantity

$$\begin{aligned}
&\frac{1}{n}\|\sum_{i=1}^{n} a_i \tilde{\mathbf{X}}\Phi(\mathbf{x_i}) - \mathbf{Y}\|^2 + \lambda\|\sum_{i=1}^{n} a_i \Phi(\mathbf{x_i})\|_{\mathcal{H}}^2 \\
&= \frac{1}{n}\left\| \sum_{i=1}^{n} a_i \begin{pmatrix} \langle \Phi(\mathbf{x_1}), \Phi(\mathbf{x_i}) \rangle \\ \vdots \\ \langle \Phi(\mathbf{x_n}), \Phi(\mathbf{x_i}) \rangle \end{pmatrix} - \mathbf{Y} \right\|^2 + \lambda \sum_{1 \leq i,j \leq n} a_i a_j \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle \\
&= \frac{1}{n}\left\| \sum_{i=1}^{n} a_i \begin{pmatrix} k(\mathbf{x_1}, \mathbf{x_i}) \\ \vdots \\ k(\mathbf{x_n}, \mathbf{x_i}) \end{pmatrix} - \mathbf{Y} \right\|^2 + \lambda \sum_{1 \leq i,j \leq n} a_i a_j k(\mathbf{x_i}, \mathbf{x_j}) \\
&= \frac{1}{n}\|\mathbf{G}a - \mathbf{Y}\|^2 + \lambda a^\top \mathbf{G} a,
\end{aligned} \tag{4.19}$$

where $\mathbf{G}$ is the Gram matrix with entries $\mathbf{G}_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$. This last expression is convex in the parameter $a \in \mathbb{R}^n$ and is minimized at

$$\hat{a} = (\mathbf{G} + n\lambda \mathrm{Id}_n)^{-1} \mathbf{Y}. \tag{4.20}$$

Note that computing $\hat{a}$ requires the inversion of a $n \times n$ matrix, which is an expensive operation if done naively (it requires $O(n^3)$ operations). The Gram matrix of size $n \times n$ also has to be stored. Although some numerical tricks exist to speed up the computations, **kernel methods still are mostly useful when the number n of observations is relatively small (say $n \lesssim 10^4$)**. You should always try to implement a simpler linear method before trying a more complex kernel approach.

Example 4.3.2. We apply kernel ridge regression to a weather forecasting problem (data from *Weather Underground API*). For four years, we are given the weather day $\mathbf{y_i}$ at the ith day of the year $\mathbf{x_i}$. The goal is to predict the weather on a future day. We implement on Python a kernel ridge regression on the data points $(\mathbf{x_i}, \mathbf{y_i})$ using the `sklearn` function `KernelRidge` with $\lambda = 1$, while using the RBF kernel k_σ. In Figure 4.2, we plot the predictions for different values of σ. The importance of the scale parameter σ is showcased. Two points x, x' are considered similar if $\|x - x'\| \lesssim \sigma$ (that is $k_\sigma(x, x')$ is close to 1), whereas if $\|x - x'\| \gtrsim \sigma$, then the points x and x' are considered different (the kernel $k_\sigma(x, x')$ is small). In our example, the value σ reflects the time scale on which we expect to have a significant temperature change. If σ is chosen too small (say $\sigma = 4$ days), then the prediction will vary quickly, as the model thinks that days that are one week apart could have very different temperatures. On the opposite, if σ is too large (say $\sigma = 200$), then we force the model to consider days that are 200 days apart as similar, leading to serious underfitting. In practice, cross validation should be used to tune the parameter σ.

4.4 Kernel Principal Component Analysis

Principal Component Analysis aims at finding the best k-dimensional subspace approximating a dataset of points $\mathbf{x_1}, \dots, \mathbf{x_n}$ in $\mathbb{R}^d$. Let $\mathbf{X}$ be the $n \times d$ matrix whose rows are given by the $\mathbf{x_i}$s. The principal components $v_1, \dots, v_k$ of the dataset are found by computing the eigenvectors corresponding to the k largest eigenvalues of the covariance matrix $\mathbf{X}\mathbf{X}^\top$ (of size $n \times n$). One can then project the data points to the vector space spanned by the k principal components to obtain a representation of the dataset in smaller dimension.

Principal Component Analysis relies on the idea that there exists a linear subspace of low dimension that can explain well the dataset. If this is not the

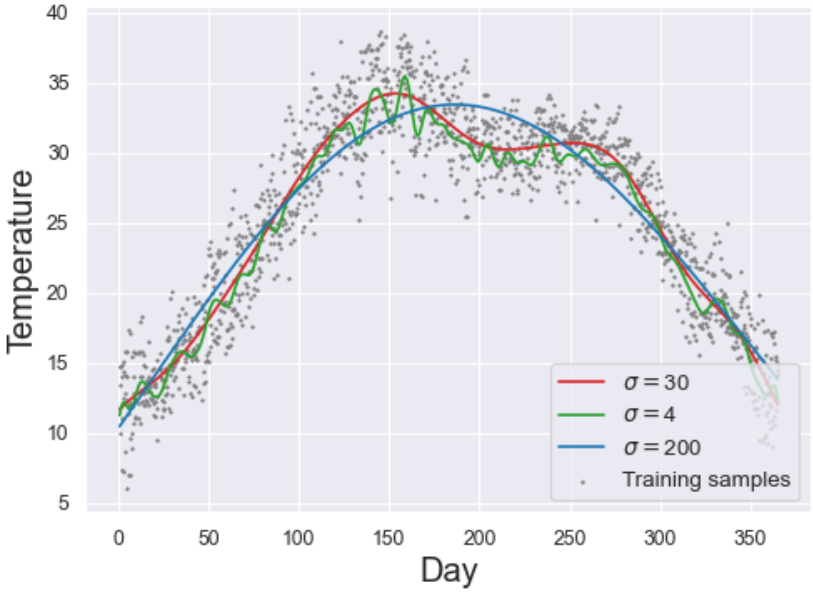


Figure 4.2: Kernel ridge regression on the weather forecast dataset with the RBF kernel k_σ for different values of σ.

case, one can still hope that a linear subspace of low dimension can be a good approximation of a transformation $\Phi(\mathbf{x_1}), \dots, \Phi(\mathbf{x_n})$ of the dataset. Kernel PCA consists in applying PCA to this transformed dataset. As before, we can consider a very general framework, where the observations $\mathbf{x_1}, \dots, \mathbf{x_n}$ lie in a general set $\mathcal{X}$, endowed with a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Let $\Phi : \mathcal{X} \to \mathcal{H}$ be an associated feature map.

Computing the principal components of $\Phi(\mathbf{x_1}), \dots, \Phi(\mathbf{x_n})$ requires to compute the covariance matrix of the corresponding points, which is by definition the matrix with entries $\langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle_{\mathcal{H}} = k(\mathbf{x_i}, \mathbf{x_j})$. Therefore, kernel PCA simply consists in computing the eigenvalues and eigenvectors of the Gram matrix $\mathbf{G}$ with entries $\mathbf{G}_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$.

Example 4.4.1. One of the best features of kernel methods is their versatility: they can be applied not only to vectors, but to inputs living in any set $\mathcal{X}$. As an example, we consider the set of words $\mathcal{X}$. The Word2Vec algorithm provides a framework to learn feature maps from $\mathcal{X}$ to a vector space $\mathbb{R}^d$ [Mikolov et al., 2013]. We use a pretrained model in the `gensim` Python library. This pretrained model gives a ready-to-use feature map $\Phi : \mathcal{X} \to \mathbb{R}^d$ with $d = 25$ that reflects meaningful similarities between different words. To showcase the performance of this feature map, we use a dataset of $n = 785$ words either corresponding to a country (e.g. *bengladesh*, *kenya*, *luxembourg*) or to an emotion (e.g. *awful*, *cruelty*, *displeasure*), see [Sharma, 2017]. Each word $\mathbf{x_i}$ is mapped to the vector $\Phi(\mathbf{x_i})$, and we represent in Figure 4.3 the projection of the vectors $\Phi(\mathbf{x_i})$ on the plane spanned by the two first principal components of the transformed dataset $(\Phi(\mathbf{x_1}), \dots, \Phi(\mathbf{x_n}))$. The points corresponding to emotions and countries are clearly separated, showing that the feature map Φ indeed captures the meaning of the words.

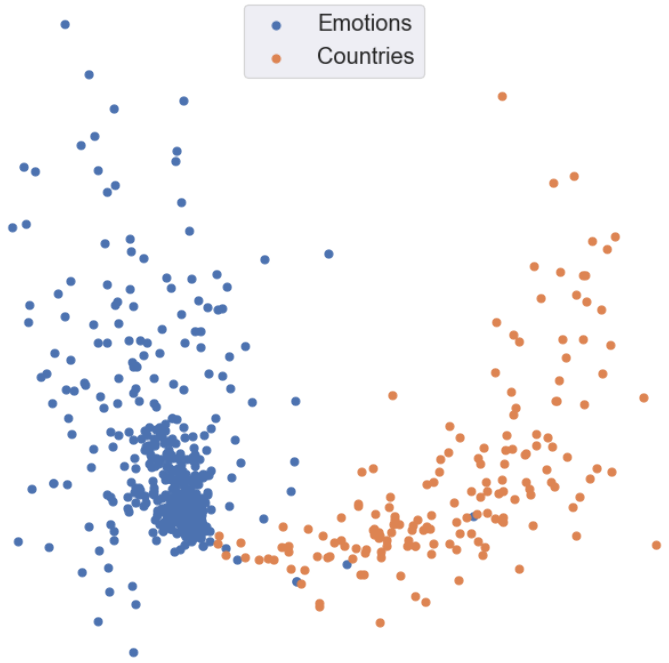


Figure 4.3: PCA (with two principal components displayed) on the dataset $(\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_n}))$. Points corresponding to emotions are in blue, while points coresponding to countries are displayed in orange.

Chapter 5

Local averaging methods

In this chapter, we investigate a class of predictors, called local averaging methods. Those methods are defined by computing a weighted average of the different outputs $\mathbf{y_i}$ from a sample of n observations $(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})$. As such, those methods are simple to compute and to interpret. However, they are best suited to low-dimensional setting as they suffer from the curse of dimensionality.

5.1 The regression problem

Let $(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})$ be a training sample with distribution P. We focus here on regression on the cube $[0,1]^d$: the set of inputs is $\mathcal{X} = [0,1]^d$, the set of outputs is $\mathcal{Y} = \mathbb{R}$, and we use the squared loss $\ell(y, y') = (y - y')^2$. Before studying local averaging methods, let us recall some basic facts on regression. We proved in Chapter 1 that the Bayes predictor for the squared loss is given by $f_P^\star(x) = \mathbb{E}_P[\mathbf{y}|\mathbf{x} = x]$, the conditional expectation of $\mathbf{y}$ given that $\mathbf{x} = x$. We can always write $\mathbf{y}$ as

$$\mathbf{y} = f_P^\star(\mathbf{x}) + \mathbf{e} \tag{5.1}$$

where $\mathbf{e}$ is defined as $\mathbf{e} = \mathbf{y} - f_P^\star(\mathbf{x})$. By construction, $\mathbb{E}[\mathbf{e}|\mathbf{x}] = 0$. We may therefore think of $\mathbf{y}$ as being obtained by corrupting $f_P^\star(\mathbf{x})$ by some random centered noise $\mathbf{e}$. Note however that the distribution of the noise $\mathbf{e}$ may depend on $\mathbf{x}$.

Example 5.1.1. Each input $\mathbf{x}$ represents a street in a city (the city being represented by a square $[0,1]^2$), and $\mathbf{y}$ represents the CO2 concentration at

$\mathbf{x}$. The output $\mathbf{y}$ will vary depending on when the CO2 concentration is measured. In this setting, $f_P^\star(x)$ represents the average CO2 concentration at the street x. The distribution of the noise $\mathbf{e}$ may vary depending on $\mathbf{x}$: for example, some streets x in the city may have higher variations of CO2 concentration than others, so that $\mathbb{E}[\mathbf{e}^2|\mathbf{x} = x]$ will be larger for those streets.

The Bayes risk $\mathcal{R}_P^\star$ is equal to

$$\mathcal{R}_P^\star = \mathbb{E}_P[(f_P^\star(\mathbf{x}) - \mathbf{y})^2] = \mathbb{E}_P[\mathbf{e}^2]. \tag{5.2}$$

Fix a function $f : \mathcal{X} \to \mathbb{R}$. Let us compute $\mathcal{R}_P(f) = \mathbb{E}_P[(f(\mathbf{x}) - \mathbf{y})^2]$. To do so, we first compute $\mathbb{E}_P[(f(\mathbf{x}) - \mathbf{y})^2|\mathbf{x}]$:

$$\begin{aligned}
\mathbb{E}_P[(f(\mathbf{x})-\mathbf{y})^2|\mathbf{x}] &= \mathbb{E}_P[(f(\mathbf{x}) - f_P^\star(\mathbf{x}) - \mathbf{e})^2|\mathbf{x}] \\
&= \mathbb{E}_P[(f(\mathbf{x}) - f_P^\star(\mathbf{x}))^2|\mathbf{x}] + 2\mathbb{E}[(f(\mathbf{x}) - f_P^\star(\mathbf{x}))\mathbf{e}|\mathbf{x}] + \mathbb{E}[\mathbf{e}^2|\mathbf{x}] \\
&= (f(\mathbf{x}) - f_P^\star(\mathbf{x}))^2 + 2(f(\mathbf{x}) - f_P^\star(\mathbf{x}))\mathbb{E}[\mathbf{e}|\mathbf{x}] + \mathbb{E}[\mathbf{e}^2|\mathbf{x}] \\
&= (f(\mathbf{x}) - f_P^\star(\mathbf{x}))^2 + \mathbb{E}[\mathbf{e}^2|\mathbf{x}],
\end{aligned}$$

where we use that $\mathbb{E}[\mathbf{e}|\mathbf{x}] = 0$. By the law of total expectation,

$$\begin{aligned}
\mathcal{R}_P(f) &= \mathbb{E}_P[\mathbb{E}_P[(f(\mathbf{x}) - \mathbf{y})^2|\mathbf{x}]] \\
&= \mathbb{E}[(f(\mathbf{x}) - f_P^\star(\mathbf{x}))^2] + \mathbb{E}[\mathbb{E}[\mathbf{e}^2|\mathbf{x}]] \\
&= \mathbb{E}[(f(\mathbf{x}) - f_P^\star(\mathbf{x}))^2] + \mathcal{R}_P^\star.
\end{aligned}$$

Therefore, the excess of risk of f is equal to

$$\mathcal{R}_P(f) - \mathcal{R}_P^\star = \mathbb{E}_P[(f(\mathbf{x}) - f_P^\star(\mathbf{x}))^2] = \int_{[0,1]^d} (f(x) - f_P^\star(x))^2 \mathrm{d}P_{\mathbf{x}}(x). \tag{5.3}$$

Two information are relevant to understand this model: properties of the noise $\mathbf{e}$ and regularity of the Bayes predictor $f_P^\star$. If $f_P^\star$ is a smooth function (for example Lipschitz continuous) and the noise $\mathbf{e}$ is small, then we expect $f_P^\star(x)$ to be similar to $\mathbf{y_i}$ for $\mathbf{x_i}$ close to x. This yields to the following heuritic.

Heuristic. *Given an input x, the predictor $\hat{f}(x)$ should be similar to the outputs $\mathbf{y_i}$ for $\mathbf{x_i}$ close to x.*

We introduce a large class of simple predictors that satisfy this heuristic. Let $w_1(x), \dots, w_n(x)$ be weights with $\sum_{i=1}^n w_i(x) = 1$ and define

$$\hat{f}_w(x) = \sum_{i=1}^n w_i(x)\mathbf{y_i}. \tag{5.4}$$

The weights $w_i(x)$ depend on the inputs $\mathbf{x_1}, \ldots, \mathbf{x_n}$. According to the heuristic, the weights $w_i(x)$ should be high if x is close to $\mathbf{x_i}$, and low otherwise.

Let us write $\mathbf{e_i} = \mathbf{y_i} - f_p^\star(\mathbf{x_i})$. We make the following assumptions on the model.

(A1) the Bayes predictor $f_P^\star : [0,1]^d \to \mathbb{R}$ is α-Lipschitz continuous, that is, for all $x, x' \in [0,1]^d$,

$$|f_P^\star(x) - f_P^\star(x')| \leq \alpha \|x - x'\|. \tag{5.5}$$

(A2) the Bayes predictor $f_P^\star$ is bounded by $\beta > 0$: for all $x \in [0,1]^d$, $|f_P^\star(x)| \leq \beta$.

(A3) the error $\mathbf{e}$ is bounded: $|\mathbf{e}| \leq \sigma$ for some $\sigma > 0$.

Under this set of assumptions, we can obtain a general decomposition result. Let $x \in [0,1]^d$. We have

$$\begin{aligned}
|\hat{f}_w(x) - f_P^\star(x)| &= \left| \sum_{i=1}^n w_i(x)(f_P^\star(\mathbf{x_i}) + \mathbf{e_i}) - f_P^\star(x) \right| \\
&\leq \left| \sum_{i=1}^n w_i(x)(f_P^\star(\mathbf{x_i}) - f_P^\star(x)) \right| + \left| \sum_{i=1}^n w_i(x)\mathbf{e_i} \right| \\
&\leq \alpha \sum_{i=1}^n |w_i(x)| \|\mathbf{x_i} - x\| + \left| \sum_{i=1}^n w_i(x)\mathbf{e_i} \right|. \qquad (5.6)
\end{aligned}$$

We refer to the first term in this decomposition as the **approximation error** $\mathrm{App}(x)$: it measures how the local average estimator is able to approximate the Bayes predictor at the point x. The second term measures the inherent noise present in the model, and we call it the fluctuation error at x, denoted by $\mathrm{Fluc}(x)$. Using the inequality $(a+b)^2 \leq 2a^2 + 2b^2$, we obtain

$$(\hat{f}_w(x) - f_P^\star(x))^2 \leq 2\mathrm{App}(x)^2 + 2\mathrm{Fluc}(x)^2. \tag{5.7}$$

Let us see how this general decomposition can be used to bound the excess of risk for different weighting schemes.

5.2 Partition estimators

A partition of a set $\mathcal{X}$ is a collection $\mathcal{A} = (A_j)_{j=1,\dots,J}$ of subsets of $\mathcal{X}$ that are pairwise disjoint (that is $A_j \cap A_{j'} = \emptyset$ if $j \neq j'$) and such that $\bigcup_{j=1}^{J} A_j = \mathcal{X}$.

Definition 5.2.1 (Partition estimator)**.** *Consider $(\mathbf{x_1}, \mathbf{y_1}), \dots, (\mathbf{x_n}, \mathbf{y_n})$ a training sample of size n from a distribution P, with inputs $\mathbf{x_i} \in [0,1]^d$ and outputs $\mathbf{y_i} \in \mathbb{R}$. Let $\mathcal{A}$ be a partition of $[0,1]^d$. For $x \in \mathcal{X}$, we let $A(x)$ be the the element A_j of the partition such that $x \in A_j$. We define the weights $w_i : [0,1]^d \to \mathbb{R}$ associated with the partition $\mathcal{A}$ by*

$$w_i(x) := \frac{\mathbf{1}\{\mathbf{x_i} \in A(x)\}}{\sum_{i'=1}^{n} \mathbf{1}\{\mathbf{x_{i'}} \in A(x)\}}. \tag{5.8}$$

If $\sum_{i'=1}^{n} \mathbf{1}\{\mathbf{x_{i'}} \in A(x)\} = 0$, then, by convention, we let $w_i(x) = 0$. The partition estimator $\hat{f}_{\mathcal{A}}$ associated with the partition $\mathcal{A}$ is the local average estimator with weights w_i. The predictor $\hat{f}_{\mathcal{A}}$ is also called a regressogram.

The predictor $\hat{f}_{\mathcal{A}}$ has a very simple structure. For $j = 1, \dots, J$, let I_j be the set of indexes i such that $\mathbf{x_i} \in A_j$, and let $\mathbf{n_j}$ be the size of I_j. If $\mathbf{n_j} = 0$, then $\hat{f}_w(x) = 0$ for $x \in A_j$. Otherwise, if $\mathbf{n_j} > 0$ and $x \in A_j$, the predictor $\hat{f}_{\mathcal{A}}(x)$ is equal to

$$\hat{f}_{\mathcal{A}}(x) = \sum_{i=1}^{n} w_i(x)\mathbf{y_i} = \frac{\sum_{i=1}^{n} \mathbf{1}\{\mathbf{x_i} \in A_j\}\mathbf{y_i}}{\sum_{i'=1}^{n} \mathbf{1}\{\mathbf{x_{i'}} \in A_j\}} = \frac{1}{\mathbf{n_j}} \sum_{i \in I_j} \mathbf{y_i}.$$

To put it otherwise, the prediction $\hat{f}_w$ is constant on each set A_j, equal to the average of the outputs $\mathbf{y_i}$ such that the corresponding input $\mathbf{x_i}$ belongs to A_j.

Example 5.2.2. Let $\mathcal{X} = [0,1]^d$ and let $L > 0$ be an integer. For $1 \leq j_1, \dots, j_d \leq L$, let $\vec{j} = (j_1, \dots, j_d)$ and

$$A_{\vec{j}} = \left[\frac{j_1 - 1}{L}, \frac{j_1}{L}\right) \times \cdot \left[\frac{j_d - 1}{L}, \frac{j_d}{L}\right). \tag{5.9}$$

The cubes $A_{\vec{j}}$ for $1 \leq j_1, \dots, j_d \leq L$ define a partition $\mathcal{A}_L$ of $\mathcal{X}$ into a grid of cubes of side length $1/L$. The predictor $\hat{f}_{\mathcal{A}_L} =: \hat{f}_L$ associated with the cube partition is constant on each of these cubes. For $d = 1$, this is simply a histogram.

The remainder of this section is dedicated to analyzing the cube partition estimator. We denote by $\hat{f}_L$ the partition estimator with partition $\mathcal{A}_L$. Let us first give a short summary of the proof strategy. We know that $\hat{f}_L(x)$ is equal to the average of the outputs $\mathbf{y_i}$ for $\mathbf{x_i}$ being in the same cube as x. As $\mathbf{y_i} = f_P^\star(\mathbf{x_i}) + \mathbf{e_i}$, and as $\mathbf{x_i}$ is at distance $1/L$ from $\mathbf{x}$, the output $\mathbf{y_i}$ is at distance $\alpha/L + |\mathbf{e_i}|$ from $f_P^\star(x)$ (see Figure 5.1). When we average the different outputs $\mathbf{y_i}$, the different error terms $\mathbf{e_i}$ will cancel out on average, so that we get an error of order $\alpha/L + \sigma/\sqrt{\mathbf{n_{\vec{j}}}}$. The conclusion is obtained by controlling $\mathbf{n_{\vec{j}}}$, which follows a binomial random variable.

Let us now turn to the rigorous mathematical analysis. Fix an index $\vec{j}$, and assume for now that $\mathbf{n_{\vec{j}}} > 0$. For $x \in A_{\vec{j}}$, it holds that

$$\begin{aligned} |\mathrm{App}(x)| &\leq \alpha^2 \left(\sum_{i=1}^{n} |w_i(x)| \|\mathbf{x_i} - x\| \right)^2 \\ &\leq \alpha^2 \left(\frac{1}{\mathbf{n_{\vec{j}}}} \sum_{i \in I_{\vec{j}}} \|\mathbf{x_i} - x\| \right)^2 \leq \alpha^2 d L^{-2}, \end{aligned} \tag{5.10}$$

where the last inequality comes from that $\|x - \mathbf{x_i}\| \leq \sqrt{d}/L$ when x and $\mathbf{x_i}$ belong to the same cube $A_{\vec{j}}$.

The fluctuation term is equal to

$$\mathrm{Fluc}(x) = \frac{1}{\mathbf{n_{\vec{j}}}} \sum_{i \in I_{\vec{j}}} \mathbf{e_i}. \tag{5.11}$$

Conditionally on $I_{\vec{j}}$, the random variables $(\mathbf{e_i})_{i \in I_{\vec{j}}}$ are independent and identically distributed. Therefore, the conditional expectation of the fluctuation error with respect to the training sample is equal to

$$\mathbb{E}\left[\mathrm{Fluc}(x)^2 \middle| I_{\vec{j}} \right] = \frac{1}{\mathbf{n_{\vec{j}}}^2} \sum_{i \in I_{\vec{j}}} |\mathbf{e_i}|^2 \leq \frac{\sigma^2}{\mathbf{n_{\vec{j}}}}. \tag{5.12}$$

From (5.7), we obtain

$$\mathbb{E}[(\hat{f}_L(x) - f_P^\star(x))^2 \mathbf{1}\{\mathbf{n_{\vec{j}}} > 0\}] \leq 2\frac{\alpha^2 d}{L^2} + 2\mathbb{E}[\mathbf{1}\{\mathbf{n_{\vec{j}}} > 0\}\frac{\sigma^2}{\mathbf{n_{\vec{j}}}}]. \tag{5.13}$$

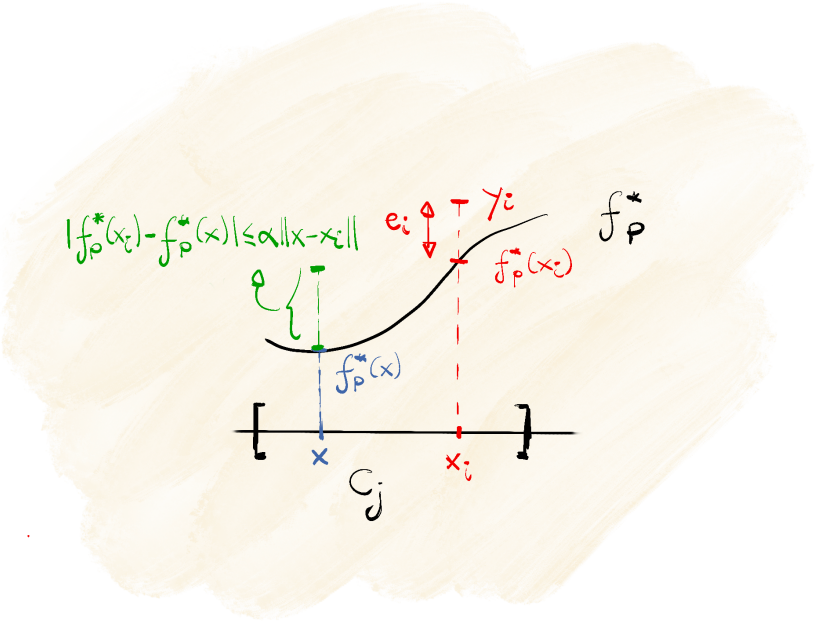


Figure 5.1: Decomposition of the distance between $\mathbf{y_i}$ and $f_P^\star(x)$ into the stochastic error term $\mathbf{e_i}$ and the distance between $f_P^\star(x)$ and $f_P^\star(\mathbf{x_i})$, which is bounded thanks to the Lipschitz property of $f_P^\star$.

It remains to control $\mathbb{E}[\mathbf{1}\{\mathbf{n}_{\tilde{\mathbf{j}}} > 0\}\mathbf{n}_{\tilde{\mathbf{j}}}^{-1}]$. Note that $\mathbf{n}_{\tilde{\mathbf{j}}}$ follows a binomial random variable of parameters n and $p_{\vec{j}} := P(\mathbf{x} \in A_{\vec{j}})$. Indeed, $\mathbf{n}_{\tilde{\mathbf{j}}}$ is the sum over all observations of independent Bernoulli random variables, equal to 1 if the observation is in $A_{\vec{j}}$, and 0 otherwise.

Lemma 5.2.3. *Let* $\mathbf{N}$ *be a binomial random variable of parameter* n *and* p*. Then,*

$$\mathbb{E}[\mathbf{1}\{\mathbf{N} > 0\}\mathbf{N}^{-1}] \leq \frac{2}{pn}. \tag{5.14}$$

Proof. We recall the formula $\frac{1}{k+1}\binom{n}{k} = \frac{1}{n}\binom{n+1}{k+1}$. The formula for the density of a binomial random variable implies that

$$\begin{aligned}
\mathbb{E}[\mathbf{1}\{\mathbf{N} > 0\}\mathbf{N}^{-1}] &= \sum_{k=1}^{n} \binom{n}{k} p^k (1-p)^{n-k} \frac{1}{k} \\
&\leq \sum_{k=1}^{n} \binom{n}{k} p^k (1-p)^{n-k} \frac{2}{k+1} \\
&\leq \frac{2}{n+1} \sum_{k=1}^{n} \binom{n+1}{k+1} p^k (1-p)^{n-k} \\
&\leq \frac{2}{n+1} \sum_{l=2}^{n+1} \binom{n+1}{l} p^{l-1} (1-p)^{n-l+1} \\
&\leq \frac{2(1-p)}{p(n+1)} \sum_{l=2}^{n+1} \binom{n+1}{l} p^{l} (1-p)^{n-l} \\
&\leq \frac{2(1-p)}{p(n+1)} \leq \frac{2}{pn}.
\end{aligned}$$

□

Using the lemma and (5.13) yields

$$\mathbb{E}[(\hat{f}_L(x) - f_P^\star(x))^2 \mathbf{1}\{\mathbf{n}_{\tilde{\mathbf{j}}} > 0\}] \leq 2\frac{\alpha^2 d}{L^2} + \frac{4\sigma^2}{p_{\vec{j}} n}. \tag{5.15}$$

When $\mathbf{n}_{\tilde{\mathbf{j}}} = 0$, then $\hat{f}_L(x) = 0$ by convention. In that case, we obtain

$$\begin{aligned}
\mathbb{E}[(\hat{f}_L(x) - f_P^\star(x))^2 \mathbf{1}\{\mathbf{n}_{\tilde{\mathbf{j}}} = 0\}] = f_P^\star(x)^2 \mathbb{P}(\mathbf{n}_{\tilde{\mathbf{j}}} = 0) &= f_P^\star(x)^2 (1 - p_{\vec{j}})^n \\
&\leq \beta^2 \exp(-n p_{\vec{j}}),
\end{aligned} \tag{5.16}$$

where we use Assumption (A2) and the formula for the probability of a binomial random variable being equal to 0. Putting the two estimates together yields

$$\mathbb{E}[(\hat{f}_L(x) - f_P^\star(x))^2] \leq 2\frac{\alpha^2 d}{L^2} + \frac{4\sigma^2}{p_{\vec{j}} n} + \beta^2 \exp(-n p_{\vec{j}}). \tag{5.17}$$

Recall from (5.3) that the excess of risk of $\hat{f}_L$ is equal to

$$\mathcal{R}_P(\hat{f}_L) - \mathcal{R}_P(f_P^\star) = \int_{[0,1]^d} (\hat{f}_L(x) - f_P^\star(x))^2 \mathrm{d}P_{\mathbf{x}}(x).$$

We obtain the following bound on the expected excess of risk (where expectation represents expectation with respect to the training sample):

$$\begin{aligned}
\mathbb{E}[\mathcal{R}_P(\hat{f}_L) - \mathcal{R}_P(f_P^\star)] &= \int_{[0,1]^d} \mathbb{E}[f_L(x) - f_P^\star(x))^2] \mathrm{d}P_{\mathbf{x}}(x) \\
&= \sum_{\vec{j}} \int_{A_{\vec{j}}} \mathbb{E}[f_L(x) - f_P^\star(x))^2] \mathrm{d}P_{\mathbf{x}}(x) \\
&\leq \sum_{\vec{j}} \int_{A_{\vec{j}}} (2\frac{\alpha^2 d}{L^2} + \frac{4\sigma^2}{p_{\vec{j}} n} + \beta^2 \exp(-n p_{\vec{j}})) \mathrm{d}P_{\mathbf{x}}(x) \\
&\leq \sum_{\vec{j}} p_{\vec{j}} (2\frac{\alpha^2 d}{L^2} + \frac{4\sigma^2}{p_{\vec{j}} n} + \beta^2 \exp(-n p_{\vec{j}})) \\
&\leq 2\frac{\alpha^2 d}{L^2} + \frac{4\sigma^2 L^d}{n} + \beta^2 \sum_{\vec{j}} p_{\vec{j}} \exp(-n p_{\vec{j}})),
\end{aligned}$$

where we use at the last line that there are exactly L^d indexes $\vec{j}$. To conclude, we need to bound the last term in the above equation. One can check that this sum is maximized in the case where all the probabilities $p_{\vec{j}}$ are equal: this sum is therefore smaller than $\exp(-nL^{-d})$.

Theorem 5.2.4 (Excess of risk of the cube partition estimator)**.** *Assume that conditions (A1)-(A3) hold. Then, the cube partition estimator $\hat{f}_L$ with side length $1/L$ satisfies*

$$\mathbb{E}[\mathcal{R}_P(\hat{f}_L) - \mathcal{R}_P(f_P^\star)] \leq 2\frac{\alpha^2 d}{L^2} + \frac{4\sigma^2 L^d}{n} + \beta^2 \exp(-nL^{-d}). \tag{5.18}$$

In particular, if $L = cn^{1/(d+2)}$ *for some constant* c*, we obtain a bound of the form*

$$\mathbb{E}[\mathcal{R}_P(\hat{f}_L) - \mathcal{R}_P(f_P^\star)] \leq Cn^{-2/(d+2)} \tag{5.19}$$

for some other constant C*.*

What should we take away from the above theorem? First, a good news: the partition estimator is consistent, as the excess of risk converges to 0. However, the rate of convergence gets increasingly slow when the number of features d increases. We say that **partition estimators suffer from the curse of dimensionality**. For example, for $d = 18$, the rate of convergence is equal to $n^{-0.1}$, which is only equal to 0.1 even for a number of observations equal to $n = 10^{10}$. This suggests that partition estimators should only be used in low-dimensional settings.

Example 5.2.5. In this example, we are exploring whether there is a relation between the oil price and the volume of oil sold at a given day at the Brent Complex, a physically and financially traded oil market based around the North Sea of Northwest Europe. The pairs $(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_1}, \mathbf{y_1})$ represent an oil price ($\mathbf{x}$ value) and a volume sold ($\mathbf{y}$ value). The dataset was downloaded from Kaggle[1]. In this example $d = 1$ and there are $n = 2859$ observations. Theorem 5.2.4 suggests that we should choose L of order $n^{1/3} \simeq 15$ when designing a partition estimator. This is what is done in Figure 5.2. We also plot the test error (obtained by randomly splitting the dataset in a training set and a testing set) as a function of L. We see that the minimum of the test error is obtained for L roughly of order 50: the theorem only gives an order of magnitude of what should be a good value of L, and nothing more precise. Moreover, we encounter once again two well-known phenomena: underfitting for L too small, and overfitting for L too large. In practice, L should be selected through cross-validation.

5.3 Nadaraya-Watson estimators

The partition estimator of the previous section can be summarized in one sentence: the prediction $\hat{f}_L(x)$ is equal to the average of the outputs $\mathbf{y_i}$ corresponding to the inputs $\mathbf{x_i}$ being in the same cube as x. In this section,

[1]See https://www.kaggle.com/datasets/psycon/historical-brent-oil-price-from-2000-to-202204.

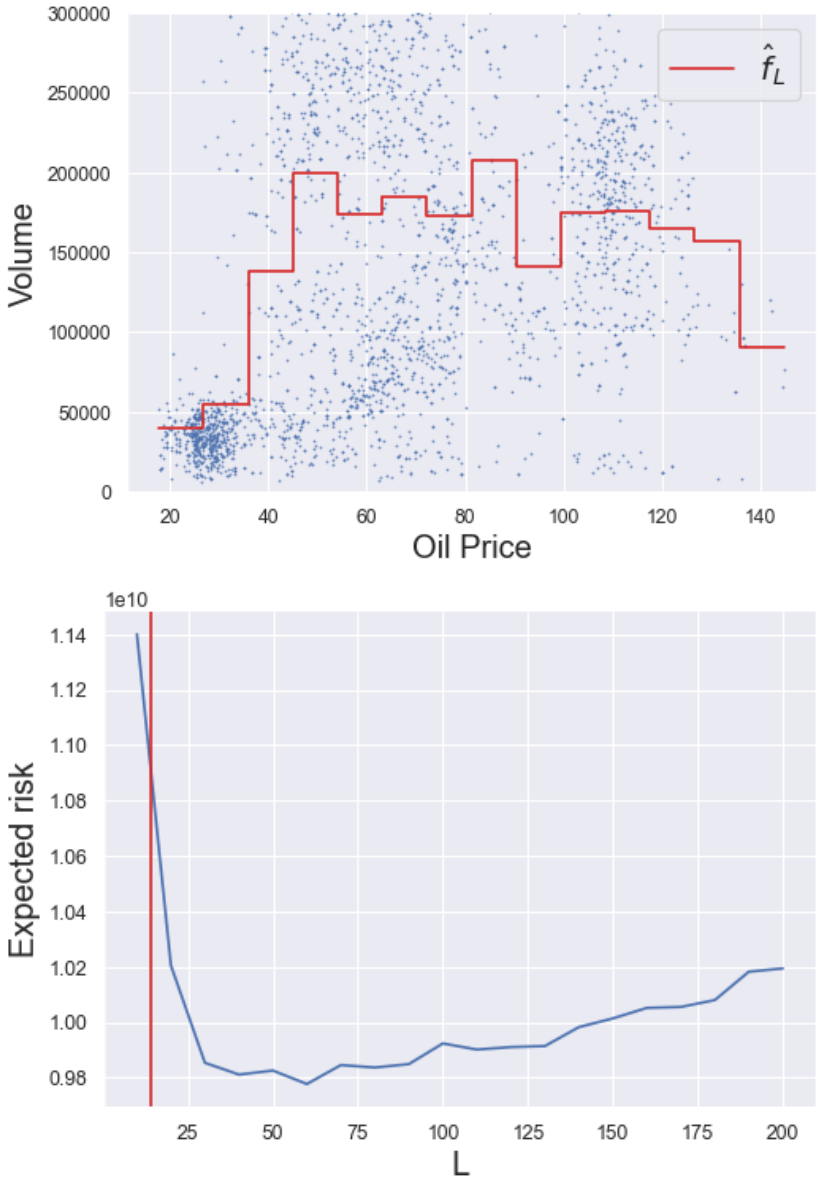


Figure 5.2: Top: prediction $\hat{f}_{L_0}$ for $L_0 = n^{1/3}$. Bottom: Expected risk for different values of L. The vertical line indicates L_0. The minimum excess of risk is attained for L roughly equal to $3L_0$.

we investigate a variation on this same idea. We choose as a prediction at the point x the average of the outputs $\mathbf{y_i}$ such that $\mathbf{x_i}$ is at distance less than h from x, where $h > 0$ is a fixed parameter. This is equivalent to defining a local averaging estimator with weights

$$w_i(x) = \frac{\mathbf{1}\{\|x - \mathbf{x_i}\| \leq h\}}{\sum_{i'=1}^{n} \mathbf{1}\{\|x - \mathbf{x_{i'}}\| \leq h\}}.$$

This can be generalized to other weighting schemes.

Definition 5.3.1. *Consider $(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})$ a training sample of size n from a distribution P, with inputs $\mathbf{x_i} \in [0,1]^d$ and outputs $\mathbf{y_i} \in \mathbb{R}$. Let $K : \mathbb{R}^d \to \mathbb{R}$ be a function with $\int K = 1$ and let $h > 0$. Let K_h be the function defined by $K_h(x) = h^{-d}K(x/h)$ for $x \in \mathbb{R}^d$. The Nadaraya-Watson estimator $\hat{f}_h^{\mathrm{NW}}$ with kernel K_h is defined as the local averaging estimator with weights at $x \in [0,1]^d$ equal to*

$$w_i(x) := \frac{K_h(x - \mathbf{x_i})}{\sum_{i'=1}^{n} K_h(x - \mathbf{x_{i'}})}. \tag{5.20}$$

The word "kernel" in the above definition is the one that is commonly used by statisticians. Note however that the local averaging method is **not** a kernel method and that the two should not be confused.

The analysis of the Nadaraya-Watson estimator is more complex than the one of the partition estimator, and we refer the interested reader to [Tsybakov, 2008, Chapter 1.5]. Let us here only mention that under assumptions similar to assumptions (A1)-(A3), it is possible to show that the Nadaraya-Watson estimator $\hat{f}_h$ satisfies

$$\mathbb{E}[\mathcal{R}_P(\hat{f}_h^{\mathrm{NW}}) - \mathcal{R}_P(f_P^\star)] \leq Cn^{-2/(d+2)}, \tag{5.21}$$

where h is of order $n^{-1/(d+2)}$ and C is a constant depending on the parameters of the model. Therefore, the Nadaraya-Watson estimator attains the same rate of convergence as the partition estimator and also suffers from the curse of dimensionality. This rate can be improved should the Bayes predictor $f_P^\star$ be k-times differentiable. In this case, one can build a Nadaraya-Watson estimator attaining a rate of convergence of order $n^{-2k/(d+2k)}$.

Example 5.3.2. A simple choice of kernel is given by the gaussian kernel defined by $K(u) = 1/(2\pi)^{d/2} \exp(-\|u\|^2/2)$ for $u \in \mathbb{R}^d$. We implement the

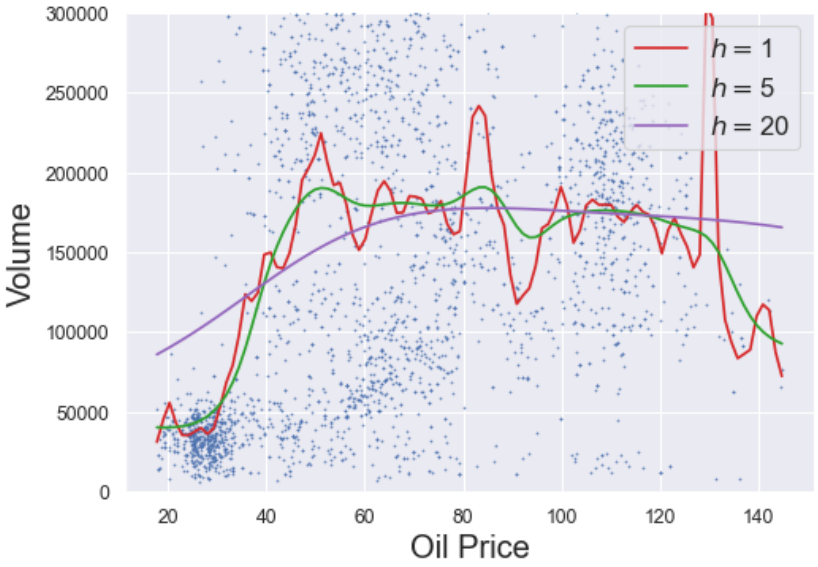


Figure 5.3: Nadaraya-Watson predictor $\hat{f}_h^{\mathrm{NW}}$ for different values of h on the oil dataset.

Nadaraya-Watson estimator on the same dataset as in Example 5.2.5, for the gaussian kernel with different choices of bandwidths h. Once again, the performance of the estimator will crucially depend on h (see Figure 5.3), a parameter which should be selected thanks to cross-validation to avoid both underfitting and overfitting.

5.4 Nearest-neighbor methods

Here is a very simple idea to make a prediction $\hat{f}(x)$ at $x \in [0,1]^d$: look at the point $\mathbf{x_i}$ the closest to x, and choose $\hat{f}(x) = \mathbf{y_i}$. Such a prediction is called the 1-nearest-neighbor estimator. A variation of this scheme is the k-nearest-neighbor (or k-NN) estimator, which is defined by averaging the outputs $\mathbf{y_i}$ corresponding the k inputs that are the closest from x.

Definition 5.4.1. *Consider $(\mathbf{x_1}, \mathbf{y_1}), \dots, (\mathbf{x_n}, \mathbf{y_n})$ a training sample of size n from a distribution P, with inputs $\mathbf{x_i} \in [0,1]^d$ and outputs $\mathbf{y_i} \in \mathbb{R}$. Let $k \geq 1$ be an integer. For $x \in [0,1]^d$, we order the inputs $\mathbf{x_i}$ according to their*

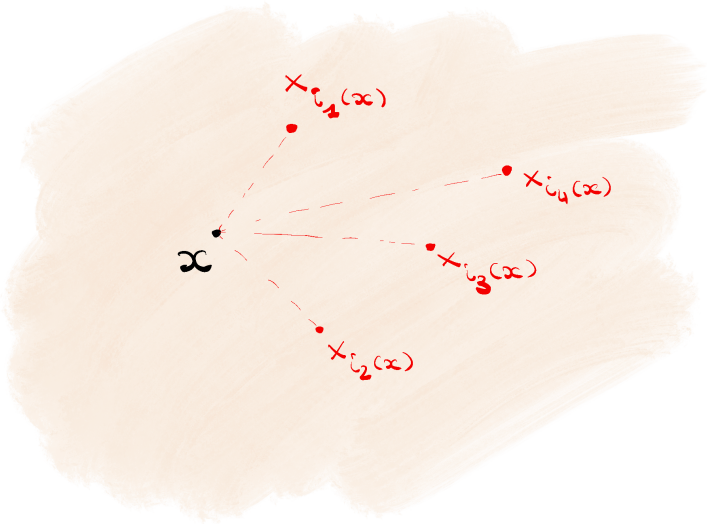


Figure 5.4: Definition of the indexes $\mathbf{i_1}(x), \ldots, \mathbf{i_4}(x)$.

distance to x:

$$\|x - \mathbf{x_{i_1}}(x)\| \leq \|x - \mathbf{x_{i_2}}(x)\| \leq \cdots \leq \|x - \mathbf{x_{i_n}}(x)\|. \tag{5.22}$$

We let $I_k(x) = \{\mathbf{i_1}(x), \ldots, \mathbf{i_k}(x)\}$ and define the weights

$$w_i(x) = \begin{cases} \frac{1}{k} & \text{if } i \in I_k(x) \\ 0 & \text{otherwise.} \end{cases} \tag{5.23}$$

The k-NN estimator $\hat{f}_k^{\mathrm{NN}}$ is the local averaging estimator associated with the weights w_i.

The k-NN estimator at a point x is equal to

$$\hat{f}_k^{\mathrm{NN}}(x) = \frac{1}{k} \sum_{i \in I_k(x)} \mathbf{y_i}, \tag{5.24}$$

that is we average the outputs of the k nearest inputs from x. The approximation error is equal to

$$\mathrm{App}(x) := \alpha \sum_{i=1}^{n} |w_i(x)| \|\mathbf{x_i} - x\| = \frac{\alpha}{k} \sum_{i \in I_k(x)} \|\mathbf{x_i} - x\|, \tag{5.25}$$

that is the average distance between x and its k-nearest neighbors. The fluctuation error is given by

$$\mathrm{Fluc}(x) := \sum_{i=1}^{n} w_i(x)\mathbf{e_i} = \frac{1}{k}\sum_{i\in I_k(x)} \mathbf{e_i}. \tag{5.26}$$

Conditionally on $I_k(x)$, this is a sum of i.i.d. random variables bounded by σ^2. We thus obtain as in Section 5.2 that

$$\mathbb{E}[\mathrm{Fluc}(x)^2] \leq \frac{\sigma^2}{k}. \tag{5.27}$$

The main part of the analysis of the k-NN estimator consists in controlling the distance $\|x - \mathbf{x_{i_k(x)}}\|$ between a point x and its kth nearest neighbor, allowing us to bound the approximation error $\mathrm{App}(x)$. Let us first consider the case $k = 1$. To make our life easier, we will assume that the distribution $P_{\mathbf{x}}$ of the inputs $\mathbf{x_i}$ has a lower bounded density on the cube.

(A4) The distribution $P_{\mathbf{x}}$ has a density p on $[0,1]^d$. Furthermore, there exists a constant $p_{\min} > 0$ such that $p(x) \geq p_{\min}$ for every $x \in [0,1]^d$.

Condition (A4) ensures that the inputs $\mathbf{x_i}$s cover all regions of the cube, and that none is missed out (which would be the case if the density p is zero on that region).

Lemma 5.4.2. *Assume that condition (A4) holds and let $x \in [0,1]^d$. Let ω_d be the volume of the unit ball in $\mathbb{R}^d$. Then, for every $t \geq 0$,*

$$\mathbb{P}(\|x - \mathbf{x_{i_1(x)}}\| \geq t) \leq \exp(-\omega_d 2^{-d} p_{\min} n t^d). \tag{5.28}$$

Proof. The condition $\|x - \mathbf{x_{i_1(x)}}\| \geq t$ is satisfied if and only if the ball $B(x,t)$ centered at x of radius t does not intersect $\{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$. The number $\mathbf{N}$ of inputs $\mathbf{x_i}$ that fall in the ball $B(x,t)$ follows a binomial random variable of parameter n and $P(B(x,t))$. Therefore,

$$\mathbb{P}(\|x - \mathbf{x_{i_1(x)}}\| \geq t) = (1 - P(B(x,t))^n \leq \exp(-nP(B(x,t))). \tag{5.29}$$

The probability $P(B(x,t))$ is lower bounded by

$$\int_{[0,1]^d} \mathbf{1}\{u \in B(x,t)\} p(u) \mathrm{d}u \geq p_{\min} \int_{[0,1]^d} \mathbf{1}\{u \in B(x,t)\} \mathrm{d}u \geq p_{\min} \frac{\omega_d}{2^d} t^d.$$

Indeed, at least a fraction of $1/2^d$ of the ball $B(x,t)$ intersects the cube $[0,1]^d$ (the worst case being attained for x being a corner of the cube). □

Going from a bound on the tail probability to a bound on the second moment is possible thanks to the next lemma.

Lemma 5.4.3. *Let* $\mathbf{z}$ *be a nonnegative random variable. Then*

$$\mathbb{E}[\mathbf{z}^2] = 2\int_0^{+\infty} u\mathbb{P}(\mathbf{z} \geq u)\mathrm{d}u. \tag{5.30}$$

Proof. We have

$$\mathbb{E}[\mathbf{z}^2] = \mathbb{E}[\int_0^{+\infty} \mathbf{1}\{\mathbf{z}^2 \geq t\}\mathrm{d}t] = \int_0^{+\infty} \mathbb{P}(\mathbf{z}^2 \geq t)\mathrm{d}t.$$

The change of variable $t = u^2$ gives the result. □

Applying this lemma yields that It holds that

$$\begin{aligned}\mathbb{E}[\|x - \mathbf{x}_{\mathbf{i_1(x)}}\|^2] &= 2\int_0^{+\infty} u\mathbb{P}(\|x - \mathbf{x}_{\mathbf{i_1(x)}}\| \geq u)\mathrm{d}u \\ &\leq 2\int_0^{+\infty} u\exp(-\omega_d 2^{-d} p_{\min} n u^d)\mathrm{d}u.\end{aligned}$$

This last integral can be computed through the change of variables $v = \omega_d 2^{-d} p_{\min} n u^d$ and by recognizing the expression of the Gamma function[2].

Lemma 5.4.4. *Assume that condition (A4) holds and let* $x \in [0,1]^d$*. Then, it holds that*

$$\mathbb{E}[\|x - \mathbf{x}_{\mathbf{i_1(x)}}\|^2] \leq \frac{\gamma}{n^{2/d}}, \tag{5.31}$$

where $\gamma = \frac{8\Gamma(2/d)}{d(\omega_d p_{\min})^{2/d}}$.

We consider now the case $k > 1$. In this case, the approximation error satisfies

$$\begin{aligned}\mathbb{E}[\mathrm{App}(x)^2] &\leq \alpha^2 \mathbb{E}\left[\left(\frac{1}{k}\sum_{i\in I_k} \|\mathbf{x_i} - x\|\right)^2\right] \\ &\leq \frac{\alpha^2}{k}\mathbb{E}\left[\sum_{i\in I_k} \|\mathbf{x_i} - x\|^2\right] \text{ by Jensen inequality.}\end{aligned} \tag{5.32}$$

[2]See https://en.wikipedia.org/wiki/Gamma_function.

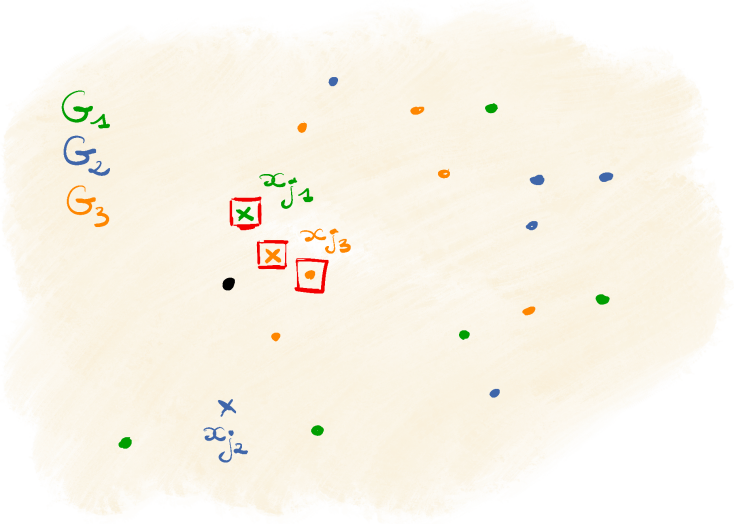


Figure 5.5: The red squares indicate the 3 nearest neighbors from the black dot x. Each color represents a group G_l of observations, whereas the crossed point is the nearest neighbor $\mathbf{x_{j_l}}$ to x in that group. The set of points $\{\mathbf{x_{j_1}}, \ldots, \mathbf{x_{j_k}}\}$ is always farther from x on average than the set of k-nearest neighbors.

The sum of squared distances is bounded thanks to an elementary (but elegant) idea: for any set J of k indexes, we have

$$\sum_{i \in I_k} \|\mathbf{x_i} - x\|^2 \leq \sum_{j \in J} \|\mathbf{x_j} - x\|^2. \tag{5.33}$$

Indeed, if we pick some index j_0 not in I_k in our set J, then the sum of the squared distances over indexes in J can always be decreased by replacing j_0 by one of the indexes of I_k that is not in J. The set J is built by splitting the set of observations $\mathbf{x_1}, \ldots, \mathbf{x_n}$ in k different groups of size roughly n/k. For sake of simplicity, we will assume that n/k is an integer and let $G_l = \{\mathbf{x_{n(l-1)/k+1}}, \ldots, \mathbf{x_{nl/k}}\}$ for $l = 1, \ldots, k$, that is G_1 contains the first n/k observations, G_2 the next n/k observations, and so on. We let j_l be the index of the nearest neighbor of x in the set G_l. See also Figure 5.5. Then, $\|x - \mathbf{x_{j_l}}\|^2$ is the squared distance between a point x and its nearest neighbor from a sample of n/k observations with distribution $P_{\mathbf{x}}$. According to Lemma 5.4.4, we have

$$\mathbb{E}[\|x - \mathbf{x_{j_l}}\|^2] \leq \frac{\gamma}{(n/k)^{2/d}}.$$

We define $J = \{j_1, \ldots, j_k\}$. Equation (5.33) then yields

$$\begin{aligned} \mathbb{E}[\sum_{i \in I_k} \|\mathbf{x_i} - x\|^2] &\leq \mathbb{E}[\sum_{j \in J} \|\mathbf{x_j} - x\|^2] \\ &\leq \sum_{l=1}^{k} \mathbb{E}[\|x - \mathbf{x_{j_l}}\|^2] \leq k\gamma \left(\frac{k}{n}\right)^{2/d}. \end{aligned} \tag{5.34}$$

Putting together (5.27), (5.32) and this last equation yields the following theorem.

Theorem 5.4.5 (Excess of risk of the k-nearest neighbor estimator)**.** *Assume that conditions (A1), (A2) and (A4) hold. Then, the k-nearest neighbor estimator $\hat{f}_k^{\mathrm{NN}}$ satisfies*

$$\mathbb{E}[\mathcal{R}_P(\hat{f}_k^{\mathrm{NN}}) - \mathcal{R}_P(f_P^\star)] \leq 2\alpha^2\gamma \left(\frac{k}{n}\right)^{2/d} + 2\frac{\sigma^2}{k}. \tag{5.35}$$

In particular, if $k = cn^{2/(d+2)}$ for some constant c, we obtain a bound of the form

$$\mathbb{E}[\mathcal{R}_P(\hat{f}_k^{\mathrm{NN}}) - \mathcal{R}_P(f_P^\star)] \leq Cn^{-2/(d+2)} \tag{5.36}$$

for some larger constant C.

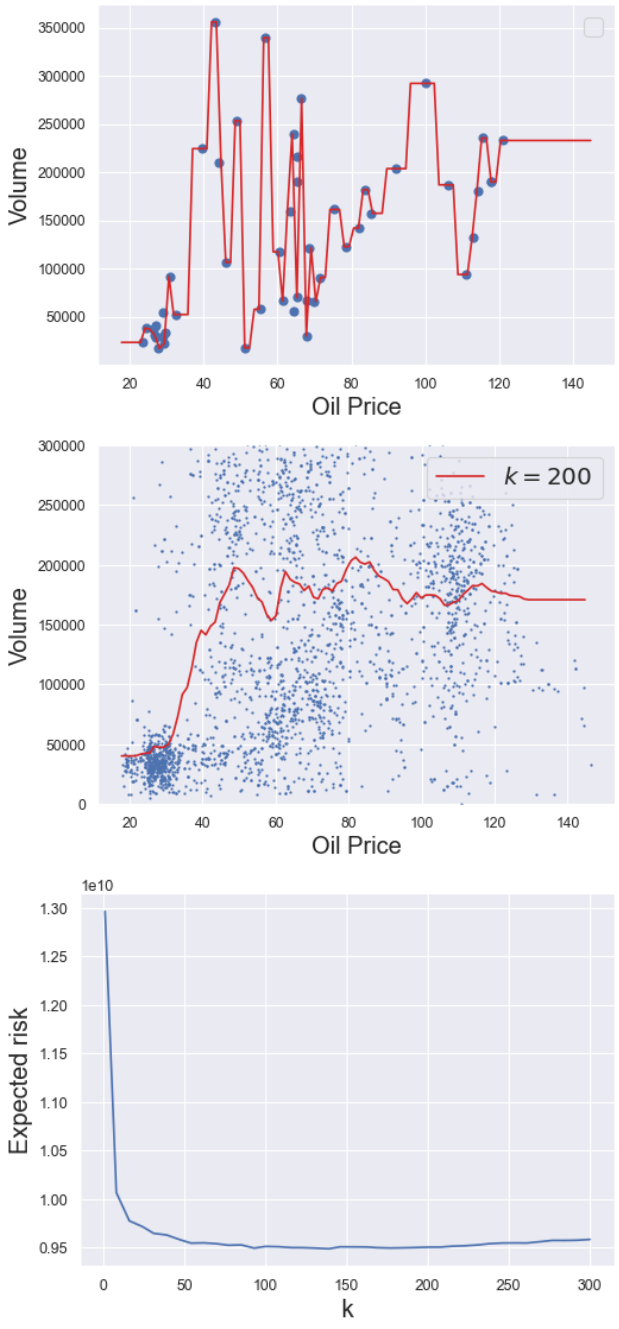


Figure 5.6: Top: the 1-NN estimator on a subsample of size $n = 50$. Middle: the k-NN estimator on the full dataset for the theoretical value $k = n^{2/3} \simeq 200$. Bottom: Expected risk for different values of k.

For an optimal choice of k, the excess of risk of the k-NN estimator is of the same order $n^{-2/(d+2)}$ as the excess of risk of the partition estimator of Section 5.2. In particular, the k-NN estimator also suffers from the curse of dimensionality. One can actually prove that, in a certain sense, the curse of dimensionality is unavoidable if we only make assumptions (A1)-(A4) on the Bayes estimator $f_P^\star$. More structural assumptions on the function $f_P^\star$ are needed to obtain better rates of convergence in high dimension $d \gg 1$.

Example 5.4.6. Eventually, we apply the k-NN estimator to the oil dataset. First, for visualization purposes, we plot the k-NN estimator for $k = 1$ on a subset of $n = 50$ observations, see Figure 5.6. Theorem 5.4.5 predicts that a choice of k of order $n^{2/3}$ is optimal for such a problem: in our example, this gives a value of $k \simeq 200$, and the corresponding k-NN estimator is displayed in Figure 5.6. We then split the set of observations into a train set and a test set, while recording the excess of risk on the test set of $\hat{f}_k^{\mathrm{NN}}$ for different values of k. It appears that $k = 50$ is enough to obtain a small excess of risk. The theorem only gives a rough order of magnitude of what k should be and not a precise value. Cross-validation should be implemented to select the parameter k in practice.

Chapter 6

Clustering methods

In previous chapters, we focused on the problem of **supervised learning**. For instance, in the image classification problem, we are given n training images $\mathbf{x_1}, \ldots, \mathbf{x_n}$ that represent a car ($\mathbf{y_i} = +1$) or a plane ($\mathbf{y_i} = -1$). The goal is then to use these examples to predict whether a new image represents a car or a plane. We focus in this chapter on a different setting, called **unsupervised learning**. In this setting, we only have access to the inputs $\mathbf{x_1}, \ldots, \mathbf{x_n}$, and not to the outputs. The goal is to create groups of inputs (called clusters) such that the inputs are similar to each other in each cluster, whereas the inputs in different clusters are dissimilar. The process of creating those different clusters is called **clustering**. In the object identification example, a clustering method aims at identifying two groups having different features in the set of observations $\mathbf{x_1}, \ldots, \mathbf{x_n}$ (corresponding to cars and planes) without having access to any label. We present two different clustering methods: the k-means method, that can be applied to observations in $\mathbb{R}^d$, and spectral clustering, that take as an input a graph of similarities between data points.

6.1 The k-means problem

Let $\mathbb{X} = (x_1, \ldots, x_n)$ be a collection of n points in $\mathbb{R}^d$. Assume that we want to summarize this set of n points with just one point x^*. What should be this point? A natural idea is to choose

$$x^* = \frac{x_1 + \cdots + x_n}{n}, \tag{6.1}$$

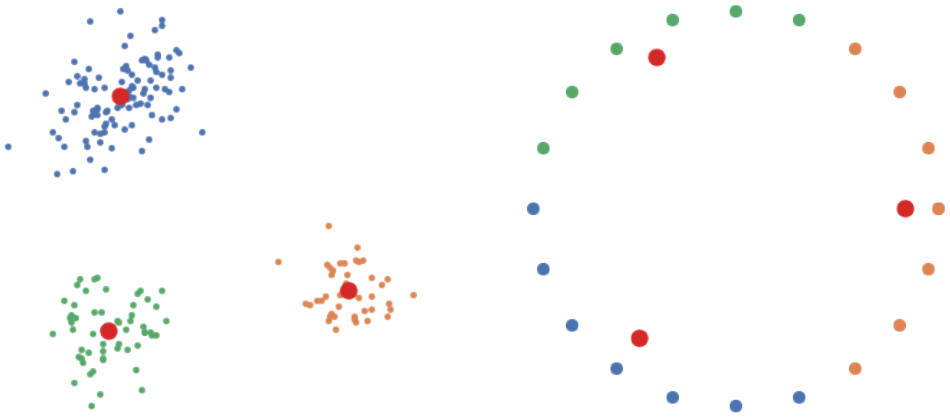


Figure 6.1: The k-means of two sets of points, for $k = 3$. For the second set of points, there is no uniqueness of the k-means.

the average of the n points. The average x^* is actually the minimizer of the function

$$y \in \mathbb{R}^d \mapsto \frac{1}{n} \sum_{i=1}^{n} \|y - x_i\|^2. \tag{6.2}$$

This can for instance be seen by computing for the gradient of the above function, which is zero for $y = x^*$. Assume now that we want to summarize the set using k points. A generalization of (6.2) consists in minimizing

$$F_{k,\mathbb{X}} : (y_1, \ldots, y_k) \in (\mathbb{R}^d)^k \mapsto \frac{1}{n} \sum_{i=1}^{n} \min_{l=1,\ldots,k} \|y_l - x_i\|^2, \tag{6.3}$$

that is we are looking for k representatives (called centroids) such that the sum of the squared distances between each x_i and its closest centroid is minimal. One call the minimizer $F_{k,\mathbb{X}}$ the set of k-means of $\mathbb{X}$, that we denote by $(y_1^*, \ldots, y_k^*)$. Note that the function $F_{k,\mathbb{X}}$ is non-convex in general (see Figure 6.2). Therefore, there might be several minimizers of $F_{k,\mathbb{X}}$, so we should in theory say 'a' set of k-means rather than 'the' set of k-means. We give in Figure 6.1 an example of data points $x_1, \ldots, x_n$ where several k-means exist.

The k-means $(y_1^*, \ldots, y_k^*)$ of the set of points $\mathbb{X}$ divide $\mathbb{X}$ into k clusters, by assigning each x_i to the lth cluster if y_l^* is centroid the closest to x_i. The

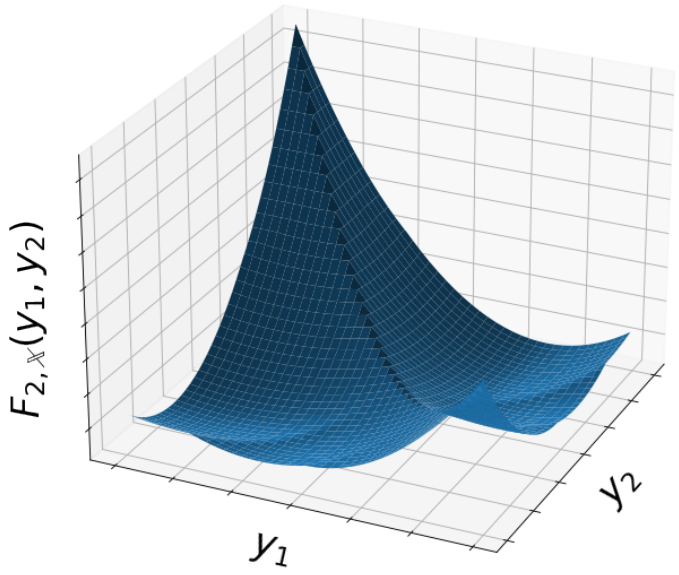


Figure 6.2: Graph of the function $F_{2,\mathbb{X}} : (\mathbb{R}^d)^2 \to \mathbb{R}$ for some set of points $\mathbb{X} \subset \mathbb{R}^d$, with $d = 1$. The function is not convex.

simplest algorithm to compute the set of k-means is called Lloyd's algorithm, presented in Algorithm 3.

Algorithm 3: Lloyd's algorithm

Initialization: centroids $y_1^0, \dots, y_k^0 \in \mathbb{R}^d$;
for $t = 0, \dots, T-1$ **do**
  for $l = 1, \dots, k$ **do**
    let $I_l^t = \{i \in \{1, \dots, n\},\ y_l^t \text{ is the centroid the closest to } x_i\}$;
    let n_l^t be the number of elements in I_l^t;
  end
  for $l = 1, \dots, k$ **do**
    let $y_l^{t+1} = \frac{1}{n_l^t} \sum_{i \in I_l^t} x_i$;
  end
end
Output: $y_1^T, \dots, y_k^T \in \mathbb{R}^d$;

Each step of Lloyd's algorithm is made of two substeps. First, we assign every point x_i to a cluster: the cluster l is chosen if x_i is the closest to y_l^t. Second, we update the centroid by defining y_l^{t+1} as the average of the points x_i of the cluster l.

Proposition 6.1.1. *Lloyd's algorithm coincides with Newton's method applied to the function $F_{k,\mathbb{X}}$.*

Proof. The function $F_{k,\mathbb{X}}$ is twice differentiable at $(y_1, \dots, y_k)$ if there are no points of the set $\mathbb{X}$ that are equidistant to some centroid y_l. We will assume that this condition is always satisfied. In this case, defining the sets I_ls and the numbers n_ls as in Algorithm 3, we can compute $F_{k,\mathbb{X}}$:

$$\begin{aligned} F_{k,\mathbb{X}}(y_1, \dots, y_k) &= \frac{1}{n} \sum_{i=1}^{n} \min_{l=1,\dots,k} \|y_l - x_i\|^2 \\ &= \frac{1}{n} \sum_{l=1}^{k} \sum_{i \in I_l} \|x_i - y_l\|^2. \end{aligned} \tag{6.4}$$

The gradient of $\nabla F_{k,\mathbb{X}}(y_1, \dots, y_k)$ is a vector of size dn that we write as

$$\begin{pmatrix} \nabla_{y_1} F_{k,\mathbb{X}}(y_1, \dots, y_k) \\ \vdots \\ \nabla_{y_k} F_{k,\mathbb{X}}(y_1, \dots, y_k) \end{pmatrix},$$

where $\nabla_{y_l} F_{k,\mathbb{X}}(y_1, \dots, y_k)$ is the partial gradient of $F_{k,\mathbb{X}}$ with respect to y_l (that is a vector in $\mathbb{R}^d$). Let us compute $\nabla_{y_1} F_{k,\mathbb{X}}(y_1, \dots, y_k)$. In (6.4), only the first term of the sum depends on y_1. Therefore,

$$\nabla_{y_1} F_{k,\mathbb{X}}(y_1, \dots, y_k) = \frac{2}{n} \sum_{i \in I_1} (y_l - x_i) = \frac{2n_1}{n} \left(y_1 - \frac{1}{n_1} \sum_{i \in I_1} x_i \right). \tag{6.5}$$

Therefore,

$$\nabla F_{k,\mathbb{X}}(y_1, \dots, y_k) = \frac{2}{n} \begin{pmatrix} n_1 \left(y_1 - \frac{1}{n_1} \sum_{i \in I_1} x_i \right) \\ \vdots \\ n_k \left(y_k - \frac{1}{n_k} \sum_{i \in I_k} x_i \right) \end{pmatrix}.$$

The gradient $\nabla F_{k,\mathbb{X}}(y_1, \dots, y_k)$ is decomposed into k blocks, where the lth block depends linearly on y_l. The Hessian $\nabla^2 F_{k,\mathbb{X}}(y_1, \dots, y_k)$ is therefore a diagonal matrix equal to

$$\frac{2}{n} \begin{pmatrix} n_1 \mathrm{Id}_d & & \\ & \ddots & \\ & & n_k \mathrm{Id}_k \end{pmatrix} \tag{6.6}$$

By definition, an iterate of Newton's method is given by

$$\begin{aligned} \begin{pmatrix} y_1' \\ \vdots \\ y_k' \end{pmatrix} &= \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} - \nabla^2 F_{k,\mathbb{X}}(y_1, \dots, y_k)^{-1} \nabla F_{k,\mathbb{X}}(y_1, \dots, y_k) \\ &= \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} - \begin{pmatrix} \frac{1}{n_1} \mathrm{Id}_d & & \\ & \ddots & \\ & & \frac{1}{n_k} \mathrm{Id}_k \end{pmatrix} \begin{pmatrix} n_1 \left(y_1 - \frac{1}{n_1} \sum_{i \in I_1} x_i \right) \\ \vdots \\ n_k \left(y_k - \frac{1}{n_k} \sum_{i \in I_k} x_i \right) \end{pmatrix} \\ &= \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} - \begin{pmatrix} y_1 - \frac{1}{n_1} \sum_{i \in I_1} x_i \\ \vdots \\ y_k - \frac{1}{n_k} \sum_{i \in I_k} x_i \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{n_1} \sum_{i \in I_1} x_i \\ \vdots \\ \frac{1}{n_k} \sum_{i \in I_k} x_i \end{pmatrix}. \end{aligned}$$

Those iterates are exactly the one given by Lloyd's algorithm. □

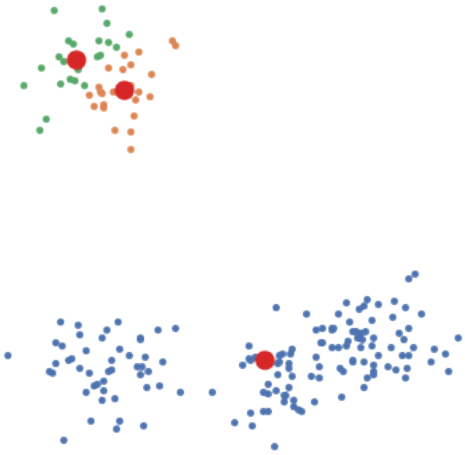


Figure 6.3: With bad initialization, Lloyd's algorithm may converge to a (very) bad configuration. The three centroids computed by Lloyd's algorithm with a bad initialization are displayed in red, and the associated clusters in respectively green, orange, and blue.

When using Lloyd's algorithm, we are applying Newton's method on the function $F_{k,\mathbb{X}}$, that is in general not convex. It is therefore not surprising that the performance of Lloyd's algorithm will crucially depend on the initialization. If the initialization $(y_1^0, \ldots, y_k^0)$ is close enough to the minimizers $(y_1^*, \ldots, y_k^*)$, then Lloyd's algorithm will converge very quickly to the minimizer (as expected for Newton's method). However, with bad initialization, the iterates of Lloyd's algorithm will remain stuck in local minima that correspond to configurations far from being optimal, see Figure 6.3. The `k-mean++` algorithm gives a procedure to find a good initialization for Lloyd's algorithm [Arthur and Vassilvitskii, 2006]. It is the default initialization method in the Python `scikit-learn` library, and shows good performance in practice.

Example 6.1.2. Add example.

6.2 Spectral clustering

Spectral clustering is in many ways an improvement upon k-means. Unlike k-means, that can only be applied to observations in $\mathbb{R}^d$, spectral clustering

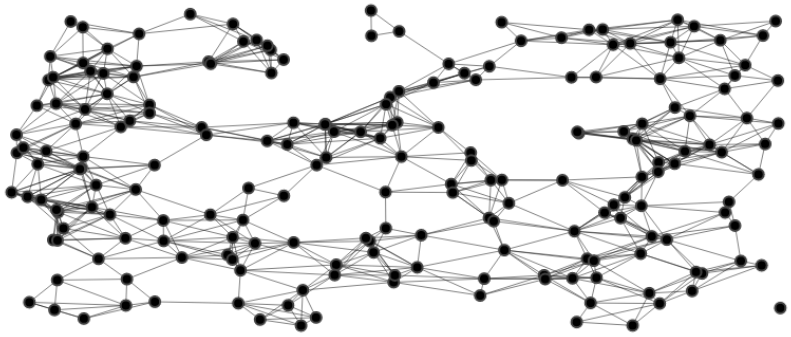


Figure 6.4: The ε-neighborhood graph of a set of points in $\mathbb{R}^2$.

can be applied to any set of observations, as long as a notion of similarity between the observations is defined.

Definition 6.2.1. *A weighted graph $\mathcal{G} := \mathcal{G}(W)$ with n vertices is described by a $n \times n$ matrix of weights $W = (W_{ij})_{1 \leq i,j \leq n}$, where the weights W_{ij} are nonnegative and symmetric (that is $W_{ij} = W_{ji}$).*

Some examples of weighted graphs include:

1. The weight matrix W only contains 0 and 1. In this case, we think of $\mathcal{G}$ as representing a non-weighted graph: if $W_{ij} = 1$, then there is an edge between the vertex i and the vertix j, and if $W_{ij} = 0$ then there is no edge.

2. A particular example of non-weighted graph is the ε-neighborhood graph. Let $x_1, \ldots, x_n \in \mathbb{R}^d$ and let $\varepsilon > 0$. We define $W_{ij} = 1$ if $\|x_i - x_j\| \leq \varepsilon$, and 0 otherwise: we connect two points by an edge if and only if they are at distance less than ε.

3. A variation of this construction is the gaussian graph, where the weights W_{ij} are given by $\exp(-\|x_i - x_j\|^2/(2\sigma^2))$ for some parameter $\sigma > 0$. Two nearby points are assigned a large weight, whereas if two points x_i and x_j are far away, then the weight W_{ij} is small.

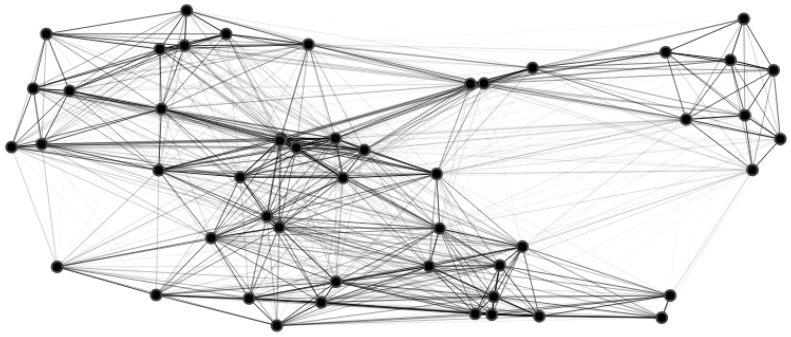


Figure 6.5: The σ-gaussian graph of a set of points in $\mathbb{R}^2$. Thicker edges indicate higher weights.

Spectral clustering allows one to detect the presence of clusters in a weighted graph. Informally, a cluster is a set of vertices such that the similarity between two points of the cluster is high, whereas the similarity between a point of a cluster and a point outside the cluster is small. In the graph setting, the similarity between vertices is given by the weight matrix W.

Definition 6.2.2. *Let $\mathcal{G}$ be a weighted graph with weight matrix W. Let $i \in \{1, \ldots, n\}$.*

1. *The **neighbors** of i are the vertices j such that $W_{ij} > 0$. We then write $i \sim_{\mathcal{G}} j$.*

2. *The **degree** D_i of a vertex i is defined as $D_i = \sum_{j=1}^{n} W_{ij}$. We let D be the $n \times n$ diagonal matrix with entries D_i on the diagonal. We call D the **degree matrix**.*

3. *We say that two vertices i and j are **connected** if there exists a path $i = i_1, i_2, \ldots, i_{k-1}, i_k = j$ such that $i_l \sim_{\mathcal{G}} i_{l+1}$ for every $1 \leq l \leq k-1$.*

4. *A **connected component** in $\mathcal{G}$ is a set $\mathcal{C}$ of vertices in $\mathcal{G}$ such that all pairs of vertices in $\mathcal{C}$ are connected, but no vertices in $\mathcal{C}$ is connected to a vertex not in $\mathcal{C}$.*

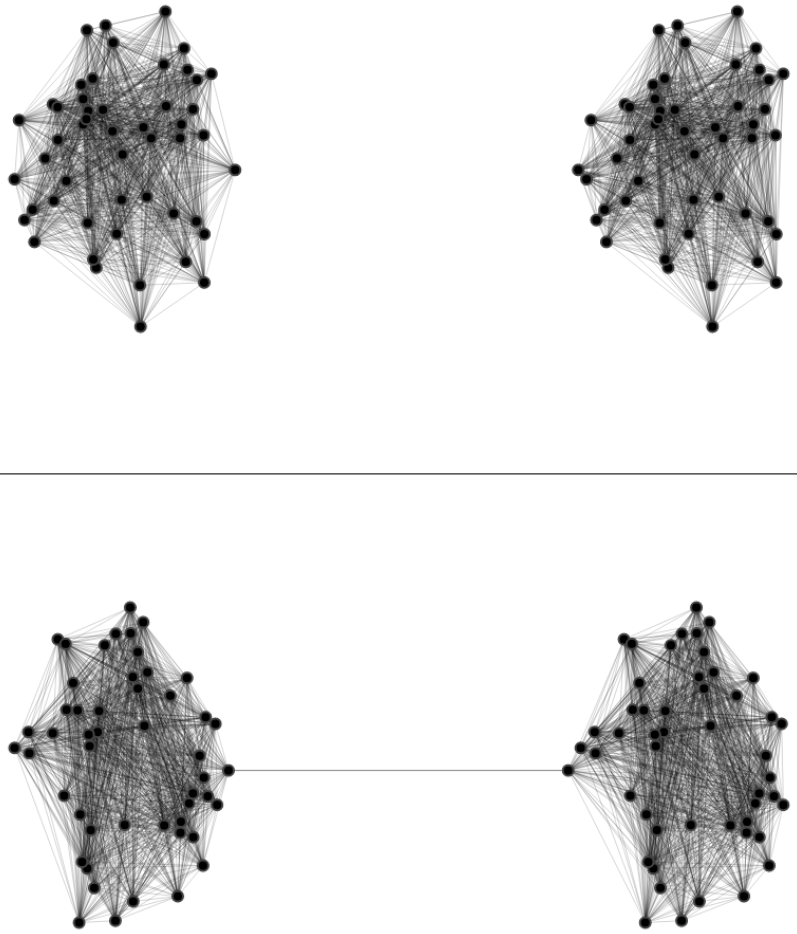


Figure 6.6: Top: a graph with two connected components. Bottom: a graph with a single connected component, but two clusters.

A first natural idea is to define the clusters in our graph as the connected components. By construction, similarity is positive between points in a connected component, but zero between two points in different clusters. However, this notion is not robust to noise, see Figure 6.6. Indeed, the top graph in Figure 6.6 has two connected component, but it suffices to add one edge (bottom graph) to create a graph with a single connected component. We still want to think about the bottom graph as containing two clusters.

To built upon this first idea, we need to make the transform the notion of "being connected" into a quantitative notion, that is we need to make sense of the sentence "How connected are two vertices?". Here is a very elegant way to do so. Let i and j be two vertices. Consider a particle p starting at i and that will randomly walk around the graph: at step t, if the particle p is at vertex k, then it will move to one of the neighbors l of k with probability equal to $Q_{kl} = W_{kl}/D_k$. Informally, if the particle p takes a lot of time to go from vertex i to vertex j, then it means that the two vertices are not well-connected. On the opposite, the random particle p goes on average from i to j in a short amount of time, then i and j can be considered closed.

Definition 6.2.3. *We define the probability transition matrix Q as $D^{-1}W$, its entries are given by $Q_{ij} = W_{ij}/D_i$. The (random walk)* ***Laplacian***[1] *of the graph $\mathcal{G}$ is the matrix $L = \mathrm{Id}_n - Q$.*

The spectral properties of the Laplacian (namely the eigenvalues and the eigenvectors of the matrix L) contain relevant information on the geometry of the graph $\mathcal{G}$, see Figure 6.7. Let $A \subset \{1, \dots, n\}$. We let e_A be the vector in $\mathbb{R}^n$ with entries $(e_A)_i = 1$ if $i \in A$ and $(e_A)_i = 0$ otherwise.

Proposition 6.2.4. *Let $\mathcal{G}$ be a weighted graph with associated Laplacian L.*

1. *All the eigenvalues of L are nonnegative.*
2. *The matrix L has always 0 as an eigenvalue.*
3. *The multiplicity of 0 as an eigenvalue is the number k of connected components of the graph $\mathcal{G}$. An orthogonal basis of the eigenspace associated to the eigenvalue 0 is given by $(e_{\mathcal{C}_1}, \dots, e_{\mathcal{C}_k})$ where $\mathcal{C}_1, \dots, \mathcal{C}_k$ are the connected components of $\mathcal{G}$.*

[1] The name Laplacian is also used in calculus to refer to a differential operator. The interested reader may find connections between the two concepts in the Appendix.

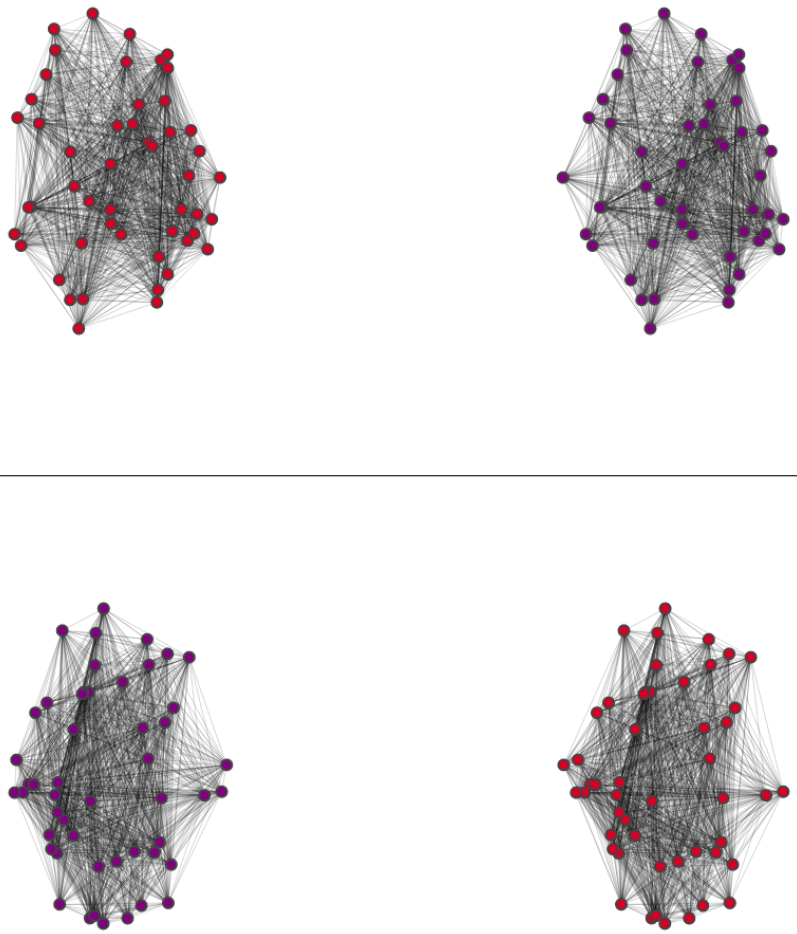


Figure 6.7: A basis of two eigenvectors of the eigenspace of L corresponding to the eigenvalue 0. The color of the vertex i can go from purple (the ith entry u_i of the eigenvector u is equal to 0) to red (u_i is equal to 1).

Proof. Introduce the matrix $L' = D^{1/2} L D^{-1/2}$. The eigenvalues and eigenvectors of L and L' are related: for $\lambda \in \mathbb{R}$ and $u \in \mathbb{R}^n$, it holds that $Lu = \lambda u$ if and only if $L'v = \lambda v$, where $v = D^{1/2}u$. In particular, L and L' have the same eigenvalues, and the eigenvectors of L and L' are related by the simple relation $v = D^{1/2}u$.

1. It suffices to show that L' is positive semi-definite. Let $v \in \mathbb{R}^n$. It holds that

$$\begin{aligned}
v^\top L' v &= v^\top v - v^\top D^{-1/2} W D^{-1/2} v \\
&= \sum_{i=1}^n v_i^2 - \sum_{1 \le i,j \le n} W_{ij} \frac{v_i v_j}{\sqrt{D_i D_j}} \\
&= \sum_{i=1}^n \frac{D_i}{D_i} v_i^2 - \sum_{1 \le i,j \le n} W_{ij} \frac{v_i v_j}{\sqrt{D_i D_j}} \\
&= \sum_{1 \le i,j \le n} W_{ij} \frac{v_i^2}{D_i} - \sum_{1 \le i,j \le n} W_{ij} \frac{v_i v_j}{\sqrt{D_i D_j}} \quad \text{(by definition of } D_i\text{)} \\
&= \frac{1}{2} \left(\sum_{1 \le i,j \le n} W_{ij} \frac{v_i^2}{D_i} - 2 \sum_{1 \le i,j \le n} W_{ij} \frac{v_i v_j}{\sqrt{D_i D_j}} + \sum_{1 \le i,j \le n} W_{ij} \frac{v_i^2}{D_i} \right) \\
&= \frac{1}{2} \left(\sum_{1 \le i,j \le n} W_{ij} \frac{v_i^2}{D_i} - 2 \sum_{1 \le i,j \le n} W_{ij} \frac{v_i v_j}{\sqrt{D_i D_j}} + \sum_{1 \le i,j \le n} W_{ij} \frac{v_j^2}{D_j} \right) \\
&= \frac{1}{2} \sum_{1 \le i,j \le n} W_{ij} \left(\frac{v_i}{\sqrt{D_i}} - \frac{v_j}{\sqrt{D_j}} \right)^2 \ge 0, \qquad (6.7)
\end{aligned}$$

   where we switch the roles of the dummy variables i and j in the last sum at the second to last line. All in all, this implies that L' is positive semi-definite, and therefore so is L.

2. It is clear from (6.7) that the vector $v = (\sqrt{D_1}, \ldots, \sqrt{D_n})$ is an eigenvector of L' with associated eigenvalue 0. Therefore 0 is also an eigenvalue of L.

3. Note also that the $D^{-1/2}v$ is an eigenvector of L, which is the vector with 1 in all of its entries. Assume that $k = 1$ (there is only one connected component), and let us show that this is the only eigenvector

of L' (up to a constant). Having $v^\top L' v = 0$ is equivalent to having $v_i/\sqrt{D_i} = v_j/\sqrt{D_j}$ for every vertices i, j with $W_{ij} > 0$. As all the vertex are connected (because $k = 1$), we can always find a path from any vertex i_1 to some other vertex i_m. Going from vertex to vertex in this path, we see that the quantity $v_i/\sqrt{D_i}$ stays constant along that path. In particular, this implies that the vector v must satisfy $v_i/\sqrt{D_i} = \text{cst}$ for every vertex i. This implies that $v = (\sqrt{D_1}, \ldots, \sqrt{D_n})$ is the only eigenvector of L' associated with 0 (up to a multiplicative constant). Therefore, $D^{-1/2}v$ is the only eigenvector of L associated with 0 (up to a multiplicative constant).

Let us now treat the case $k > 1$. In this case, up to relabelling the indexes, we can write the matrix L as a block diagonal matrix

$$L = \begin{pmatrix} L_1 & & \\ & \vdots & \\ & & L_k \end{pmatrix}$$

where the block L_l is the Laplace matrix associated with the subgraph given by the connected component $\mathcal{C}_l$. As 0 is an eigenvalue with multiplicity 1 of each of the matrix L_l, it is an eigenvalue of L with multiplicity k. Furthermore, a basis of the eigenspace of L associated with 0 is given by one eigenvector of each of the block L_l with 0 eigenvalue. According to the case $k = 1$, the vectors $e_{\mathcal{C}_l}$ are such eigenvectors.

□

To put it simply, the multiplicity of 0 as an eigenvalue of the Laplacian L gives the number of connected components, while the associated eigenvectors exactly give the said connected components: all the relevant information on the connected components is given by the spectral properties of the Laplacian matrix. However, the Laplacian matrix is a much richer object, that is more robust to perturbation. Indeed, if we consider a graph with two large connected components, and add a single edge between those two components, then only one single connected component remain, as said earlier. The multiplicity of 0 in the Laplacian matrix has moved from 2 to 1. However, one can show that there is now a nonzero eigenvalue λ in the spectrum of L that is very small. Also, an eigenvector corresponding to this eigenvalue will be approximately 0 on one connected component and approximately 1 on the

other, that is to say the eigenvector is an approximation of the eigenvector $e_{\mathcal{C}_1}$ given by Proposition 6.2.4. This phenomenon is showcased in Figure 6.8.

The spectral clustering method relies on this phenomenon, see Algorithm 4. For $i \in \{1, \dots, n\}$, let $e_i = e_{\{i\}}$ be the vector in $\mathbb{R}^n$ representing the vertex i. In the case where $\mathcal{G}$ contains exactly k connected components, one has $\langle e_i, e_{\mathcal{C}_l} \rangle = 1$ if $i \in \mathcal{C}_l$ and 0 otherwise. Therefore, in this "ideal" situation, the points $a_1, \dots, a_n \in \mathbb{R}^k$ are all on one of the k axes of $\mathbb{R}^k$ (in its canonical basis). In particular, Lloyd's algorithm will have no trouble clustering the points $a_1, \dots, a_k$. In the more realistic setting where only "approximate" connected components exist, one can show that the points a_i corresponding to the "approximate" lth connected component will stay close to the lth axis of $\mathbb{R}^k$. Therefore, Lloyd's algorithm will still be able to identify the clusters.

Algorithm 4: Spectral clustering

Input: weighted graph $\mathcal{G}$ with weight matrix W; number of clusters k;
compute the Laplace matrix L;
compute the eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and the associated eigenvectors $u_1, \dots, u_n$.
for $i = 1, \cdots, n$ **do**
  let $a_i = (\langle u_1, e_i \rangle, \dots, \langle u_k, e_i \rangle) \in \mathbb{R}^k$;
end
apply Lloyd's algorithm to $a_1, \dots, a_n \in \mathbb{R}^k$;

Appendix - the Laplace matrix and the heat equation

The reader familiar with calculus may have already heard about the Laplacian in another context. If f is a twice differentiable function from $\mathbb{R}^d$ to $\mathbb{R}$, the Laplacian of f is defined by

$$\Delta f = -\sum_{j=1}^{d} \frac{\partial^2 f}{\partial x_j^2}, \tag{6.8}$$

(note the sign convention that we use here). How is this operator related to the Laplacian on the graph that we have defined in this chapter? In 1822,

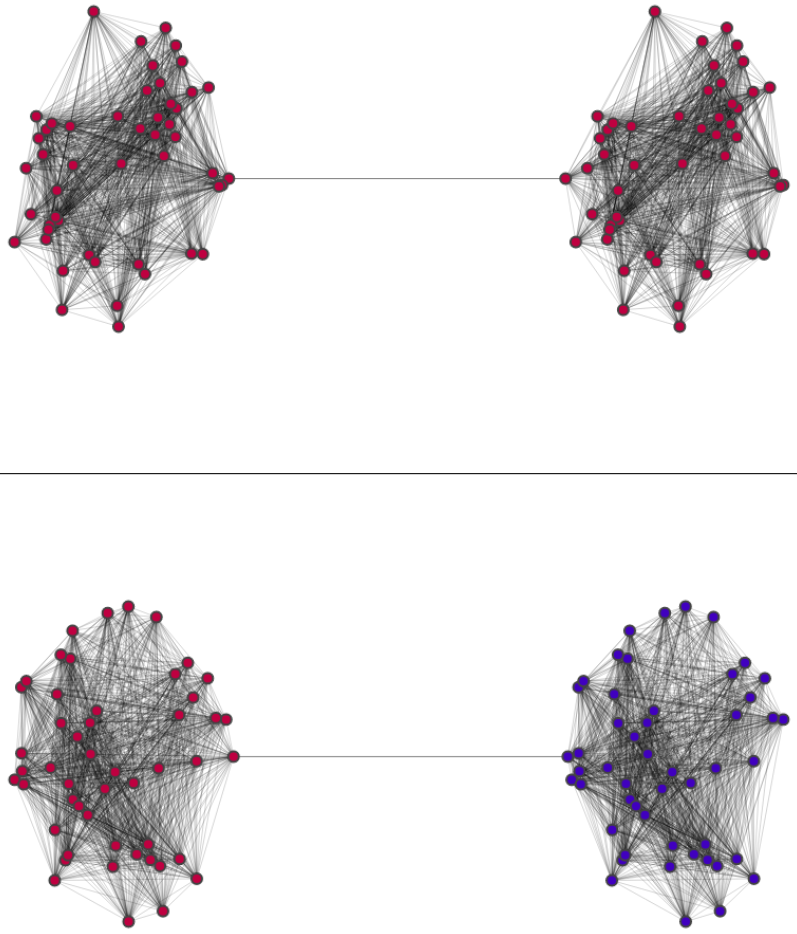


Figure 6.8: Top: the eigenvector of L corresponding to the eigenvalue 0. Bottom: the eigenvector of L corresponding to the smallest positive eigenvalue, equal to $\simeq 10^{-3}$.

Joseph Fourier published his treatise *Théorie analytique de la chaleur*, where he considered the following problem. Consider a metal rod (that we identify with $[0,1]$) whose temperature at both extremities is fixed at 1 throughout the whole experiment. Assume that at time $t = 0$ a certain distribution of heat f_0 is given in the rod (mathematically, a function $f_0 \geq 0$ with $\int f_0 = 1$). How will the distribution of heat evolve through time? Our intuition dictates that as time t progresses, the distribution of heat f_t at time t will become smoother, until the temperature is uniform in the rod at $t = \infty$, corresponding to $f_\infty = 1$. The equation governing the distribution of heat is the **heat equation**

$$\frac{\partial f_t}{\partial t} + \Delta f_t = 0. \tag{6.9}$$

Solving this equation indeed shows that the distribution of heat f_t will converge exponentially fast to a uniform temperature. A microscopic vision of the heat diffusion process consists in picturing a large number of small particles randomly moving in the rod. At each time t, the particle will with probability $1/2$ infinitesimally moves to the left, or to the right. The initial distribution f_0 represents exactly the original distribution of those particles in the rod (if the heat is originally high at a point x in the rod, then f_0 is high, meaning that there are originally a lot of particles oscillating around x). For a very small value of t, we can approximate the time derivative $\partial f_t/\partial t$ by $(f_t - f_0)/t$. The heat equation (6.9) then becomes

$$f_t \simeq f_0 - t\Delta f_0 = (\mathrm{Id} - t\Delta) f_0.$$

This means that the distribution of heat at a small time t is approximately given by the operator $(\mathrm{Id} - t\Delta)$.

Let us now go back to the discrete world. Consider the graph $\mathcal{G}_n$ with n vertices, and weights

$$W_{ij} = \begin{cases} 1 & \text{if } |i-j| = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{6.10}$$

The graph $\mathcal{G}_n$ is a "line" graph, that is a discrete analogue of the metal rod. If we start with a particle at position i, that moves at random, then after one step, the probabilities of the position of i are given by the vector Q_i (where $Q = D^{-1}W$ is the probability transition matrix). Let us now consider a large number of particles on the graph, distributed according to some distribution $u_0 \in \mathbb{R}^n$ (with nonnegative entries and such that $\sum_{i=1}^n (u_0)_i = 1$). We make all those particles take one random step on the graph. After this step, the

distribution of the particles is given by $Qu_1 = (\mathrm{Id}_n - L)u_0$ (by definition of the graph Laplacian). This is exactly the discrete time analogue of the equation $f_t \simeq (\mathrm{Id} - t\Delta)f_0$.

Therefore, the graph Laplacian and the Laplace operator from calculus share the same physical meaning: they both represent how particles moving at random behave on average on a small scale of time.

Bibliography

[Arthur and Vassilvitskii, 2006] Arthur, D. and Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Technical report, Stanford.

[Bach, 2022] Bach, F. (2022). *Learning theory from first principles*.

[Bansal and Gupta, 2019] Bansal, N. and Gupta, A. (2019). Potential-function proofs for gradient methods. *Theory of Computing*, 15(1):1–32.

[Biau and Devroye, 2015] Biau, G. and Devroye, L. (2015). *Lectures on the nearest neighbor method*, volume 246. Springer.

[Boyd et al., 2004] Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[Bronstein, 2008] Bronstein, E. M. (2008). Approximation of convex sets by polytopes. *Journal of Mathematical Sciences*, 153(6):727–762.

[Çinar et al., 2020] Çinar, İ., Koklu, M., and Taşdemir, Ş. (2020). Classification of raisin grains using machine vision and artificial intelligence methods. *Gazi Mühendislik Bilimleri Dergisi (GMBD)*, 6(3):200–209.

[Gower, 2018] Gower, R. M. (2018). Convergence theorems for gradient descent. *Lecture notes for Statistical Optimization*.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

[Nesterov, 2018] Nesterov, Y. (2018). *Lectures on convex optimization*, volume 137. Springer.

[Shalev-Shwartz and Ben-David, 2014] Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

[Sharma, 2017] Sharma, A. (2017). Sentiment package for r. https://github.com/abhy/sentiment.

[Smola and Schölkopf, 1998] Smola, A. J. and Schölkopf, B. (1998). *Learning with kernels*, volume 4. Citeseer.

[Tsybakov, 2008] Tsybakov, A. B. (2008). *Introduction to Nonparametric Estimation*. Springer Publishing Company, Incorporated, 1st edition.

[von Luxburg, 2007] von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.

[Yeh and Hsu, 2018] Yeh, I.-C. and Hsu, T.-K. (2018). Building real estate valuation models with comparative approach through case-based reasoning. *Applied Soft Computing*, 65:260–271.