Introduction to Machine Learning

Alessandro Rudi, Umut Şimşekli
Class notes by Antoine Groudiev

Last modified 15th April 2024

Contents

Abstract

This document is Antoine Groudiev's class notes while following the class *Introduction to Machine Learning* (Apprentissage Statistique) at the Computer Science Department of ENS Ulm. It is freely inspired by the class notes written by Pierre Gaillard, Alessandro Rudi, and Umut Şimşekli.

1 An overview of Machine Learning

2 Linear Least Squares Regression

2.1 Introduction

In this chapter, we will study the simple but still widely used problem of *Linear Least Square Regression*. We are given a set of points, which we assume to be sampled from some distribution: there exists some function which generated these points, and we want to retrieve or at least approximate this unknown function. To do so, we will naturally look for the function which best fits the points; nevertheless, assuming that it is unlikely that the function is overly-complicated, we will only approximate it using *linear function*. Finally, to choose which linear function "fits best" the data, we will introduce the mean square error, which we will minimize to find our linear approximation function.

Formally, our objective is to find a function f such that it explains well the distribution $(y_i)_{1 \leqslant i \leqslant n}$ as a function of $(x_i)_{1 \leqslant i \leqslant n}$, that is $y_i \sim f(x_i)$. To do this, we can choose a *function space* $\mathcal{F}$ and solve the empirical risk minimization problem:

$$\hat{f}_n \in \arg\min_{f \in \mathcal{F}} \hat{R}_n(f) := \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

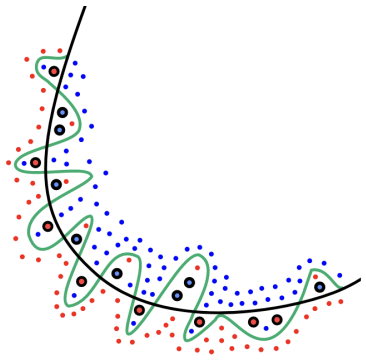


Figure 1: Example of overfitting

Care must be taken when selecting the function space to avoid overfitting[1]: for example, if we were to choose $\mathcal{F} := \mathbb{R}_d[X]$ for some $d \in \mathbb{N}$, it is in our best interest to keep d small to avoid getting a function f which fits perfectly the points but diverges between them. Although the empirical mean square error $\hat{R}_n$ decreases when the function space $\mathcal{F}$ becomes larger (i.e. larger polynomial degrees), the $\hat{f}_n$ estimator loses its predictive power. $\hat{f}_n$ will not necessarily perform well on new data. In what follows, we will consider the linear function space, containing functions of the form $f : x \mapsto ax + b$, which is the simplest.

2.2 General setup and notations for supervised learning

Definition (Training data set)**.** The *training data set*, often denoted $D_n := \{ (x_i, y_i) \mid i \in [\![1, n]\!] \}$, is the set of some observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. We will often make the assumption that the observations (x_i, y_i) are realizations of i.i.d. random variables from a distribution ν.

The distribution ν is unknown to the statistician; it's a matter of learning it from the data.

[1] We say that the function *overfits* the data when it corresponds too closely to the specific set of data (i.e. on the training data), such that it fails to fit additional data (i.e. test data).

Definition (Learning algorithm)**.** A *learning algorithm* – or *learning rule* – $\mathcal{A}$ is a function that associates to training data D_n a prediction function $\hat{f}_n$:

$$\mathcal{A} : \bigcup_{n\in\mathbb{N}} (\mathcal{X}\times\mathcal{Y})^n \longrightarrow \mathcal{Y}^{\mathcal{X}}$$
$$D_n \longmapsto \hat{f}_n$$

The estimated function $\hat{f}_n$ is constructed to predict a new output y from a new input x, where (x, y) is a pair of *test data*, i.e. not necessarily observed in the training data. The function $\hat{f}_n$ is an estimator because it depends on the data D_n and not on unobserved parameter, such as the distribution ν. If D_n is random, it is also a random function.

Definition (Squared Loss Risk)**.** Given an estimator $\hat{f}_n$, we define its risk:

$$\mathcal{R}(\hat{f}_n) := \mathbb{E}\left[(Y - \hat{f}_n(X))^2 \,\middle|\, D_n\right] \quad \text{where} \quad (X, Y) \sim \nu \tag{2.2.1}$$

This is also called the *generalization error*, as it measures how well the estimator performs on other inputs and outputs of the dataset.

In practice, the statistician cannot compute the risk, since one cannot acces the distribution ν. Therefore, a common method in supervised machine learning is to replace the risk (defined using the distribution ν through the expectation $\mathbb{E}$) by the empirical risk (defined using the training data set).

Definition (Squared Loss Empirical risk)**.** Given an estimator $\hat{f}_n$ and a data set $D_n = \{(x_i, y_i) \mid i \in [\![1, n]\!]\}$, we define its *empirical risk*:

$$\hat{\mathcal{R}}_n(\hat{f}_n) := \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}_n(x_i))^2 \tag{2.2.2}$$

However, one must be careful about overfitting, the case where $\hat{\mathcal{R}}_n(f)$ is much lower than $\mathcal{R}(f)$, as discussed previously. In this chapter, we will study the performance of the least square estimator in the case of the linear model.

Definition (Linear model)**.** When $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$, the simplest interesting function space is the set of affine functions. To ease the notation, we assume that the frist components of the inputs is 1 so that it is sufficient to consider linear functions. Therefore, the function space is:

$$\mathcal{F} := \left\{ x \mapsto \theta^\top x \,\middle|\, \theta \in \mathbb{R}^d \right\} \tag{2.2.3}$$

i.e. linear functions parametrized by $\theta \in \mathbb{R}^d$.

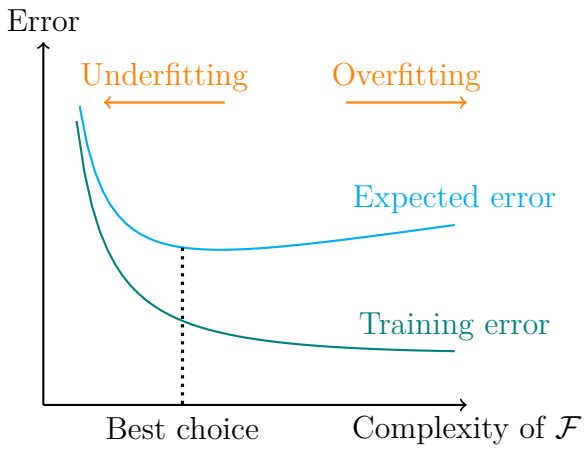


Figure 2: Overfitting and underfitting

Remark. *Choosing a linear model, the empirical risk minimization corresponds to the problem of minimizing the following quantity over* $\theta \in \mathbb{R}^d$*:*

$$\hat{\mathcal{R}}_n(\theta) := \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^\top x_i)^2$$

This expression can be rewritten using matrix notation. We let $Y = (Y_1, \dots, Y_n)^\top \in \mathbb{R}^n$ *be the vector of outputs and* $X \in \mathbb{R}^{n\times d}$ *the matrix of inputs, whose rows are* $x_i^\top$*. X is called the* design matrix*. The empirical risk is therefore given by:*

$$\hat{\mathcal{R}}_n(\theta) = \frac{1}{n}\|Y - X\theta\|_2^2$$

2.3 Ordinary Least Squares Estimator (OLS)

2.3.1 Definition and closed form

In the following, we assume that the design matrix is injective.[2] In particular, $d \leqslant n$, otherwise this is not possible.

Definition (Ordinary Least Squares)**.** If X is injective, the minimizer of the empirical risk (2.2.2) is called the *Ordinary Least Squares (OLS) estimator*. Said otherwise, it is the vector $\hat{\theta} \in \mathbb{R}^d$ minimizing $\hat{\mathcal{R}}_n$:

$$\hat{\theta} := \underset{\theta \in \mathbb{R}^d}{\arg\min}\, \hat{\mathcal{R}}_n(\theta) = \underset{\theta \in \mathbb{R}^d}{\arg\min}\, \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^\top x_i)^2 \tag{2.3.1}$$

Property 1 (Closed form soluion of the OLS estimator)**.** If X is injective, the OLS estimator exists and is unique. Moreover, it is given by:

$$\hat{\theta} = (X^\top X)^{-1} X^\top Y \tag{2.3.2}$$

Proof. Since $\hat{\mathcal{R}}_n$ is coercive[3] and continuous, it admits at least a minimizer. Furthermore, we have:

$$\hat{\mathcal{R}}_n(\theta) := \frac{1}{n}\|Y - X\theta\|_2^2 = \frac{1}{n}\left(\theta^\top (X^\top X)\theta - 2\theta^\top X^\theta Y + \|Y\|^2\right)$$

Since $\hat{\mathcal{R}}$ is differentialble any mimizer cancels its gradient:

$$\nabla\hat{\mathcal{R}}_n(\hat{\theta}) = \frac{1}{n}\left(\hat{\theta}^\top (X^\top X) + (X^\top X)\hat{\theta} - 2X^\top Y\right) = \frac{2}{n}\left((X^\top X)\hat{\theta} - Y^\top X\right)$$

where the last equality holds because $X^\top X \in \mathbb{R}^{d\times d}$ is symmetric. Since X is injective, $X^\top X$ is invertible[4]. Therefore, a solution of $\nabla\hat{\mathcal{R}}_n(\hat{\theta}) = 0$ satisfies:

$$\hat{\theta} = (X^\top X)^{-1} XY$$

Finally, this unique solution is indeed a minimum since its Hessian is definite positive:

$$\nabla^2\hat{\mathcal{R}}_n(\hat{\theta}) = \frac{2}{n}(X^\top X)$$

□

2.3.2 Geometric interpretation

The linear model aims modeling the ouput vector $Y \in \mathbb{R}^n$ by a linear combination of the form $X\theta \in \mathbb{R}^n$. The image of X is the solution space, denoted:

$$\mathrm{Im}(X) = \left\{ X\theta \in \mathbb{R}^n \,\middle|\, \theta \in \mathbb{R}^d \right\}$$

This is the vector subspace of $\mathbb{R}^n$ generated by the $d \leqslant n$ columns of the design matrix. As $\mathrm{rg}(X) = d$, it is of dimension d.

By minimizing $\|Y - X\theta\|$, we thus look for the element of $\mathrm{Im}(X)$ closest to Y. This is the orthogonal projection of Y on $\mathrm{Im}(X)$, denoted $\hat{Y}$. By definition of the OLS and by Property 1, we have:

$$\hat{Y} := X\hat{\theta} = X(X^\top X)^{-1}X^\top Y$$

In particular, $P_X := X(X^\top X)^{-1}X^\top$ is the projection matrix on $\mathrm{Im}(X)$.

[2] Said otherwise, the rank of X is d.

[3] $\|\hat{\mathcal{R}}_n(\theta)\| \xrightarrow[\|\theta\|\to+\infty]{} +\infty$

[4] It is even positive definite.

2.3.3 Numerical resolution

The closed form formula (2.3.2) of the OLS is useful in analyzing it; however, calculating it naively can be prohibitively expensive. For example, when d is large, one prefers to avoid inverting the design matrix $X^\top X$ which costs $O(d^3)$[5], and can be very unstable when the matrix is badly conditioned. The following methods are usually preferred.

QR factorization To improve stability, QR decomposition can be used. Since $\hat{\theta}$ is the solution of the equation:

$$(X^\top X)\hat{\theta} = X^\top Y$$

we write $X \in \mathbb{R}^{n\times d}$ as $X = QR$ where $Q \in \mathbb{R}^{n\times d}$ is an orthogonal matrix[6] and $R \in \mathbb{R}^{d\times d}$ is upper triangular. Upper triangular matrices are very useful for solving linear systems. Substituting in the previous equation, we get:

$$\begin{aligned} R^\top(Q^\top Q)R\hat{\theta} = R^\top Q^\top Y &\iff R^\top R\hat{\theta} = R^\top Q^\top Y \\ &\impliedby R\hat{\theta} = Q^\top Y \end{aligned}$$

All that remains is to solve a linear system with a triangular upper matrix, which is easy.

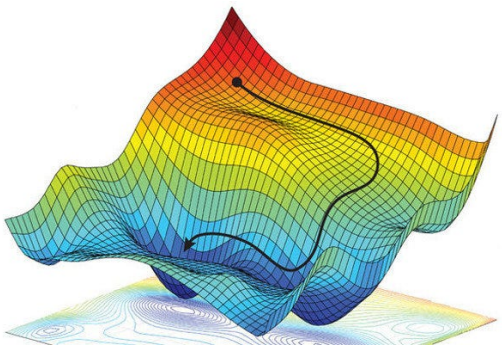


Figure 3: (Non-convex) Gradient descent

Gradient descent We can completely bypass the need of matrix inversion or factorization using gradient descent. It consists in solving the minimization problem step by step by approching the minimum throught gradient steps. For example, we initialize $\theta_0 := 0$[7], then update it using the following recurrence formula:

$$\begin{aligned} \theta_{i+1} &:= \theta_i - \eta \cdot \nabla\hat{\mathcal{R}}_n(\theta_i) \\ &= \theta_i - \eta \cdot \frac{2}{n}\left((X^\top X)\theta_i - Y^\top X\right) \end{aligned}$$

where $\eta > 0$ is a learning parameter called *learning rate*. We see that if the algorithm converges, then it converges to a point cancelling the gradient, thus to the (unique) OLS solution. For the algorithm to converge, the learning rate η must be well calibrated. This will be seen in more details in the following chapter about gradient descent.

If the data set is too big, i.e. when $n \gg 1$, loading all the data to make the gradient calculation $\nabla\hat{\mathcal{R}}(\theta_i)$ can be prohibitively expensive too. The common solution to this is to use *Stochastic Gradient Descent*, where gradient calculations for one step are made only on estimates of $\nabla\hat{\mathcal{R}}(\theta_i)$, calculated on a random subset of the data.

2.3.4 Nonlinear problem: polynomial, spline, and kernel regression

The assumption tha the observations y_i can be explained as a linear combination of the explanatory variables $x_{i,j}$ may seem strong. However, the previous linear framework can be applied to transformations of the variables $x_{i,j}$. For example, by adding the powers of the variables $x_{i,j}^k$ or their products $x_{i,j} \cdot x_{i',j'}$, this allows comparison to polynomial spaces. Doing a

[5] Using the Gauss-Jordan method

[6] That is $QQ^\top = I_n$

[7] In practice, θ_0 is often initialized to some random vector to avoid singularities.

linear regression on polynomial transformations of variables is equivalent to doing a polynomial regression.

Of course, other bases and transformations exist: for instance, *spline bases* are piecewise polynomials with constraints on the edgets. This is the model used for example by EDF to predict electricity consumption as a function of variables such as the time of the day, the day of the week, temperature or cloud cover. In general, we can consider transformations $\phi : \mathcal{X} \to \mathbb{R}^d$ and try to explain the outputs y_i with functions of the form $\theta \to \phi(x_i)^\top \theta$. Another form of regression that we will discuss in the following is therefore *kernel regression*, which allows computing efficiently the estimator even when ϕ maps to an infinite dimensional space.

2.4 Statistical analysis

In this section, we try to show some guarantees for the OLS estimator, such as a bound the excess risk of the OLS, which we will define later on.

2.4.1 Stochastic assumptions and bias/variance decomposition

To provide guarantees on the performance of OLS, we require assumptions about how the data is generated. In this section, we consider a stochastic framework that will allow us to statistically analyze OLS.

Assumption: linear model We assume that there exists a vector $\theta^* \in \mathbb{R}^d$ such that for all $1 \leqslant i \leqslant n$,

$$Y_i = x_i^\top \theta^* + Z_i \tag{2.4.1}$$

where $Z = (Z_1, \dots, Z_n)^\top \in \mathbb{R}^n$ is a vector of *errors*, also called *noise*. The Z_i are assumed to be centered independent variables of variance σ^2, i.e. $\mathbb{E}[Z_i] = 0$ and $\mathbb{V}[Z_i] = \sigma^2$. The assumption (2.4.1) can be rewritten in matrix form:

$$Y = X\theta^* + Z \tag{2.4.2}$$

where $Y = (Y_1, \dots, Y_n)^\top \in \mathbb{R}^n$, $X = (x_1, \dots, x_n)^\top \in \mathbb{R}^{n \times d}$, and $Z = (Z_1, \dots, Z_n)^\top \in \mathbb{R}^n$.

Remark. *We write Y_i and Z_i using capital letters to remind ourselves that they are random variables. The noise Z comes from the fact that in practice, the observations Y_i never completely fit the linear forecast. They are due to noise or unobserved explanatory variables. As before, we assume that the first vector of the explanatory variable is the constant vector, i.e. $\forall i, x_{i,1} = 1$*

Analysis settings Two settings of analysis can be chosen:

- In the *fixed desing* setting, the design matrix X is not random but deterministic and the features $x_1, \dots, x_n$ are fixed. The expectations are thus only with respect to the Z_i and the Y_i, and the goal is to minimize:

$$\mathcal{R}_X(\theta) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - x_i^\top \theta)^2\right]$$

  for new random observations Y_i, but on the same inputs.
- In the *random design* setting, both the inputs and the ouputs are random. This is the most standard setting in supervised machine learning. The goal is therefore to minimize the risk (2.2.1) – also called the generalization error.

In this chapter, we will consider the fixed design setting, because it eases both the notation and the calculations – we will only need some simple linear algebra.

Before analyzing the statistical properties of OLS, we state a general result under the linear model assumption, which illustrates the tradeoff between estimation and approximation – or bias and variance.

Property 2 (Risk decomposition)**.** Under the linear model assumption with fixed design, the following hols:

$$\forall \theta \in \mathbb{R}^d, \quad \mathbb{E}[\mathcal{R}_X(\theta) - \mathcal{R}_X(\theta^*)] = \|\theta - \theta^*\|_\Sigma^2$$

where $\Sigma := \frac{1}{n} X^\top X \in \mathbb{R}^{d \times d}$ and $\|\alpha\|_\Sigma^2 := \alpha^\top \Sigma \alpha$. Furthermore, if θ is a random variable – when it depends on a random dataset – we have:

$$\mathbb{E}[\mathcal{R}_X(\theta)] - \mathcal{R}(\theta^*) = \underbrace{\|\mathbb{E}[\theta] - \theta^*\|_\Sigma^2}_{\text{Bias}} + \underbrace{\mathbb{E}\left[\|\theta - \mathbb{E}[\theta]\|_\Sigma^2\right]}_{\text{Variance}}$$

Proof. □

Remark. *It is worth to note that the optimal risk satisfies:*

$$\mathcal{R}_X(\theta^*) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n (Y_i - x_i^\top \theta^*)^2\right] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n Z_i^2\right] = \frac{1}{n}\sum_{i=1}^n \mathbb{E}[Z_i^2] = \sigma^2$$

2.4.2 Statistical properties of OLS

We can now show some guarantees for the OLS estimator.

Property 3. Under the linear model assumption with fixed design, the OLS estimator $\hat{\theta}$ defined by (2.3.1) satisfies:

$$\mathbb{E}[\hat{\theta}] = \theta^* \quad \text{and} \quad \mathbb{V}[\hat{\theta}] = \frac{\sigma^2}{n}\Sigma^{-1}$$

i.e. it is unbiased, and its variance is some $O(n^{-1})$. Furthermore, we can even show that it satisfies the Gauss-Markov property: it is optimal among all unbiased estimators of θ, in the sense that is has a minimal variance-covariance matrix.

Proof. □

Definition (Excess risk)**.** The excess risk of an estimator $\hat{\theta}$ is defined by:

$$\mathscr{R}(\hat{\theta}) := \mathbb{E}[R_X(\hat{\theta})] - \mathcal{R}(\theta^*) \tag{2.4.3}$$

It allows to measure the risk induced by the use of an estimator instead of the optimal vector, without taking into account the inherent risk generated by the noise, which cannot be reduced.

Corollary (Excess risk of OLS)**.** Under the linear model assumption with fixed design, the excess risk of the OLS satisfies:

$$\mathscr{R}(\hat{\theta}) := \mathbb{E}[\mathcal{R}_X(\hat{\theta})] - \mathcal{R}(\theta^*) = \frac{\sigma^2 d}{n}$$

Proof. □

2.4.3 Gaussian noise model

A special case that is often considered is Gaussian noise, i.e. $Z_i \sim \mathcal{N}(0, \sigma^2)$, the normal distribution of expectation 0 and variance σ^2. This choice comes not only from the fact that it allows to compute many additional statistical properties on $\hat{\theta}$ and to perform tests, such as confidence intervals or significance of variables. In practice, it is also motivated by the central limit theorem, and the fact that noise is often an addition of many phonomena not explained by the linear combination of the explanatory variables.

Property 4. In the linear model with Gaussian noise, the maximum likelihood estimators of θ and σ satisfy respectively:

$$\hat{\theta}_{MV} = (X^\top X)^{-1} XY \quad \text{and} \quad \hat{\sigma}^2_{MV} = \frac{\|Y - X\hat{\theta}\|^2}{n}$$

We therefore find the least squares estimator obtained by minimizing the empirical risk. The variance estimator is biased. We will see more about maximum likelihood estimators in next chapters.

2.5 Ridge regression

2.5.1 Handling non-injective design matrices

If X is not injective, the matrix $X^\top X$ is no longer invertible and the OLS optimization problem admits several solutions. The problem is said to be *poorly posed* or *unindentifiable*. Since the variance of $\hat{\theta}$ depends on the conditioning of the matrix $(X^\top X)^{-1}$, the more columns of it are likely to be dependent, the less stable $\hat{\theta}$ will be. Several solutions allow dealing with the case where $\mathrm{rg}(X) < D$.

Explicit complexity control reduces the $\mathrm{Im}(X)$ solution space: this can be done by removing columns from the X matrix until it becomes injective (for example, by reducing the degree of polynomials). One can also set identifiability constraints of the form $\theta \in V$, a vector subspace of $\mathbb{R}^d$ such that any element $y \in \mathrm{Im}(X)$ has a unique antecedent $\theta \in V$ with $y = X\theta$. For example, we could choose $V = \ker(X)^\perp$.

Implicit complexity control regularizes the empirical risk minimization problem. The most common approach is to regularize by adding $\|\theta\|_2^2$ (Ridge regression) or $\|\theta\|_1$ (Lasso regression).

2.5.2 Ridge regression

Definition. For a regularization parameter λ, the Ridge regression estimator is defined as

$$\hat{\theta}_\lambda \in \arg\min \left\{ \frac{1}{n}\|Y - X\theta\|_2^2 + \lambda\|\theta\|_2^2 \,\middle|\, \theta \in \mathbb{R}^d \right\} \tag{2.5.1}$$

The regularization parameter $\lambda > 0$ regulates the trade-off between the variance of $\hat{\theta}$ and its bias.

Property 5. The Ridge regression estimator is unique and satisfies:

$$\hat{\theta}_\lambda = (X^\top X + n\lambda I_n)^{-1} X^\top Y$$

Proof. Similar to the one of the OLS and left as an exercise. We can see that there is no longer the problem of inverting $X \top X$ since the Ridge regression replaces $(X^\top X)^{-1}$ by $(X^\top X + n\lambda I_n)^{-1}$ in the OLS solution. □

Property 6 (Risk of Ridge regression)**.** Under the linear model assumption, the Ridge regression estimator satisfies:

$$\mathbb{E}[\mathcal{R}_X(\hat{\theta}_\lambda)] - \mathcal{R}_X(\theta^*) = \sum_{j=1}^{d} (\theta_j^*)^2 \frac{\lambda_j}{(1+\lambda_j/\lambda)^2} + \frac{\sigma^2}{n} \sum_{j=1}^{d} \frac{\lambda_j^2}{(\lambda_j+\lambda)^2}$$

where λ_j is the j-th eigenvalue of $\Sigma = \frac{1}{n} X^\top X$. In particular, the choice of:

$$\lambda^* = \frac{\sigma \sqrt{\operatorname{Tr}(\Sigma)}}{\|\theta^*\|_2 \sqrt{n}}$$

yiels the following excess risk:

$$\mathbb{E}[\mathcal{R}_X(\hat{\theta}_{\lambda^*})] - \mathcal{R}_X(\theta^*) \leqslant \frac{\sigma \sqrt{2 \operatorname{Tr}(\Sigma) \|\theta^*\|_2}}{\sqrt{n}}$$

Proof. Follows from the bias-variance decomposition, and is left as an exercise. □

Remark. *As $\lambda \to 0$, its excess risk converges to the one of OLS. The first term corresponds to the bias of the Ridge estimator. Thus, on the downside, the Ridge estimator is biased in constrast to the OLS. But on the positive side, its variance does not involve the inverse of Σ but of $\Sigma + \lambda I_d$ instead, which is better conditioned. It has therefore a lower variance. The parameter λ controls the trade-off.*

2.5.3 Comparaison to the OLS

We can compare the excess risk bound obtained by $\hat{\theta}_{\lambda^*}$ with the one of the OLS, which was $\sigma^2 d/n$.

- The OLS convergence is in $O(n^{-1})$ while the convergence of $O(n^{-1/2})$, which is slower
- The OLS dependency on the noise is in σ^2 while Ridge's is in σ, which is better
- Since $\operatorname{Tr}(\Sigma) \leqslant \max_{1 \leqslant i \leqslant n} \|x_i\|^2$, if the input norms are bounded by R, the excess risk of Ridge does not depend on the dimension d, which can even be infinite. It is called a *dimension free* bound.

The calibration of the regularization parameter is therefore essential in practice. It can for example be done analytically as in the proposition – but often, some quantities such as σ^2 and $\|\theta^*\|$ are unknown. In practice, one resorts to train/validation set or *cross-validation*.

3 Logistic regression and convex analysis

This chapter will introduce logistic regression, a widely used classification algorithm. Unlike linear regression, there is no closed-form solution and one needs to solve it using iterative convex optimization algorithms.

3.1 Logistic regression

We consider the binary classification problem: given inputs in $\mathbb{R}^d$, we want to predict outputs in $\{0, 1\}$. We are given a training set $D_n = \{ (X_i, Y_i) \mid i \in [\![1, n]\!] \}$, where the data points (X_i, Y_i) are i.i.d. random variables following a distribution ν in $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^d \times \{0, 1\}$

3.1.1 Motivation

We would like to use an algorithm similar to linear regression introduced in the previous chapter. However, since the outputs Y_i are binary and belong to $\{0, 1\}$, a discrete set, we cannot predict them using a linear transformation of the inputs X_i. We will thus classify the data based on a classification rule of the form

$$f : \mathbb{R}^d \longrightarrow \mathbb{R}$$

which will then be passed through a function with specification $\mathbb{R} \to \{0, 1\}$. The final estimator will therefore be:

$$\mathbb{1}_{\mathbb{R}_+} \circ f$$

meaning that the estimator will predict $Y_i = +1$ exactly when $f(X_i) \geqslant 0$.

More precisely, we will consider linear functions f of the form:

$$f_\beta = x \longmapsto x^\top \beta$$

This assumes that the data is *linearly separable*, meaning that it can be well-explained by a linear separation.

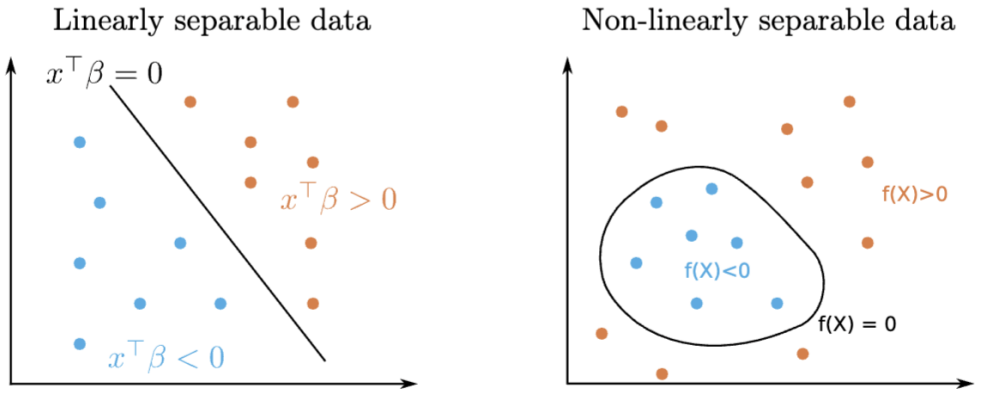


Figure 4: Data that can or cannot be well-explained by a linear separation

If the data does not seem to be linearly separable, we can use similar tricks as the one introduced for linear regression (polynomial regression, kernel regression, splines, ...). We will dive into more details in an upcoming chapter about kernels.

3.1.2 Loss function

To minimize the empirical risk, it remains to choose a loss function to assess the performance of a prediction.

Definition (Loss function)**.** A *loss function* ℓ is a function of the form

$$\ell : \mathcal{Y} \times \mathcal{Y}' \longrightarrow \mathbb{R}_+$$

such that for $(y, y') \in \mathcal{Y} \times \mathcal{Y}'$, $\ell(y, y')$ intuitively quantifies the mistake when predicting y' instead of y.

Definition (Empirical Risk associated to a Loss function)**.** Given an estimator $\hat{f}_n$, a data set $D_n = \{ (x_i, y_i) \mid i \in [\![1, n]\!] \}$ and a loss function ℓ, we define the *empirical risk associated to* ℓ by:

$$\hat{\mathcal{R}}_n(\hat{f}_n) := \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \hat{f}_n(x_i)) \tag{3.1.1}$$

Definition (Squared loss)**.** We define the *squared loss* $\ell_2 : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ used previously in linear regression by:

$$\ell_2(y, y') := (y - y')^2 \tag{3.1.2}$$

Using the squared loss ℓ_2 in (3.1.1), we obtain the same result as the original definition (2.2.2).

Definition (Binary loss)**.** A natural loss is the *binary loss*, also known as the *0-1 loss*. It takes 1 as a value if there is a mistake (i.e. $f(X_i) \neq Y_i$) and 0 otherwise:

$$\ell_1(X_i, Y_i) = \delta_{Y_i \neq \mathbb{1}_{\mathbb{R}_+}(f(X_i))} \tag{3.1.3}$$

Using ℓ_1, the empirical risk becomes:

$$\hat{\mathcal{R}}_n(\beta) = \frac{1}{n} \sum_{i=1}^{n} \delta_{Y_i \neq \mathbb{1}_{\mathbb{R}_+}(X_i^\top \beta)}$$

This loss function is however not convex in β; therefore, the problem of minimizing $\hat{\mathcal{R}}_n$ is extremely hard to solve. The idea of logistic regression consists in replacing the binary loss with another similar loss function, which is convex in β. This is the case of both the *Hinge loss* and the *logistic loss* which we will now introduce.

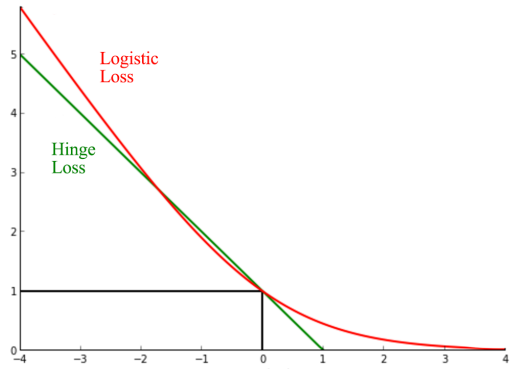


Figure 5: Binary, Hinge, and logistic loss for $y' = 1$

Definition (Hinge loss)**.** The *Hinge loss* $\ell_H : \mathbb{R} \times \{-1, 1\} \to \mathbb{R}_+$ is such that it values to 0 when both arguments have the same sign and that $|y| \geqslant 1$ (confident prediction), and increases linearly when their signs differ or when $|y| < 1$ (prediction not confident enough):

$$\ell_H(y, y') := \max(0, 1 - yy') \tag{3.1.4}$$

Definition (Logistic loss)**.** The *logistic loss* $\ell_l : \mathbb{R} \times \{0, 1\} \to \mathbb{R}_+$ is defined by:

$$\ell_l(y, y') := y' \log(1 - e^{-y}) + (1 - y') \log(1 + e^{y}) \tag{3.1.5}$$

The advantage of the logistic loss with respect to the Hinge loss is that it has a probabilistic interpretation, by modeling $\mathbb{P}(Y = 1|X)$, where (X, Y) is a couple of random variables following the law (X_i, Y_i). We will see more on this in the lecture on Maximum Likelihood.

Definition (Logistic regression estimator)**.** The logistic regression estimator is the solution of the following minimization problem:

$$\hat{\beta}_{(\text{logi.})} := \arg\min_{\beta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell_l(X_i^\top \beta, Y_i) \tag{3.1.6}$$

3.1.3 Computation of the estimator

Similarly to OLS, we may try to analyticall solve the minimization problem to find $\hat{\beta}_{(\text{logi.})}$. This could be done by cancelling the gradient of the empirical risk. Note that:

$$\frac{\partial \ell_l(y, y')}{\partial y} = \sigma(y) - y'$$

where σ is the logistic function

$$\sigma = z \longmapsto \frac{1}{1+e^{-z}}.$$

Therefore,

$$\nabla\hat{\mathcal{R}}_n(\beta) = \frac{1}{n}\sum_{i=1}^{n} X_i(\sigma(X_i^\top\beta) - Y_i) = \frac{1}{n}X(Y - \sigma(X\beta))$$

where $\sigma(X\beta)_i := \sigma(X_i^\top\beta)$. The problem is that the equation $\nabla\hat{\mathcal{R}}_n(\beta) = 0$ had no closed-form solution. Therefore, it needs to be solved through iterative algorithms (gradient descent, Newton's method, ...). Fortunately, this is possible, since the logistic loss is convex in its first argument. Indeed:

$$\frac{\partial^2\ell_l(y,y')}{\partial y^2} = \sigma(y)\sigma(-y) > 0$$

Furthermore, the loss being strictly convex, the solution is even unique. In this chapter and in the next one, we will see tools and methods to solve convex optimization problems.

3.1.4 Regularization

Similarly to linear regression, logistic regression may over-fit the data (especially when $p > n$). In this case, one needs to add a regularization term, such as $\lambda\|\beta\|_2^2$ to the logistic loss.

3.2 Convex analysis

3.2.1 Convexity and minimization problems

We will now see notions of convex analysis to solve convex optimization problems, such as logistic regression. This chapter will introduce convex analysis – the properties of convex functions and convex optimization problems, and the next chapter will approach convex optimization algorithms (gradient descent, Newton's method, stochastic gradient descent, ...).

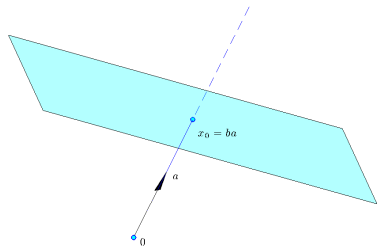


Figure 6: Hyperplane

Convexity is a crucial notion in many fields of mathematics and computer science. In machine learning, convexity creates well-defined problems with efficient solutions. A typical example is the problem of *empirical risk minimization*:

$$\hat{f}_n \in \arg\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \ell(f(X_i), Y_i) + \lambda\Omega(f)$$

where $D_n = \{ (X_i, Y_i) \mid i \in [\![1, n]\!] \}$ is the data set, $\mathcal{F}$ is a *convex* set of predictors $f : \mathcal{X} \to \mathbb{R}$, for all $y' \in \mathcal{Y}$, $y \mapsto \ell(y, y')$ is a convex loss function, and Ω is a convex penaly (such as $\|\cdot\|_2$, $\|\cdot\|_1$, ...).

Convexity will be useful to analyze bot statistical properties of the solution $\hat{f}_n$ and its generalization error:

$$\mathcal{R}(\hat{f}_n) := \mathbb{E}[\ell(f(X), Y)|D_n]$$

but also to derive efficient algorithms to solve the forementionned minimization problem and find $\hat{f}_n$.

3.2.2 Convex sets

In what follows, we will only consider finite dimensional Euclidean spaces (typically $\mathbb{R}^d$).

Definition (Convex set)**.** A set $K \subseteq \mathbb{R}^d$ is convex if and only if:

$$\forall x, y \in K, \forall \alpha \in [0,1], \quad \alpha x + (1-\alpha) y \in K$$

Said otherwise, K is stable by barycentration, or, for all points $x, y \in K$, $[x, y] \subseteq K$.

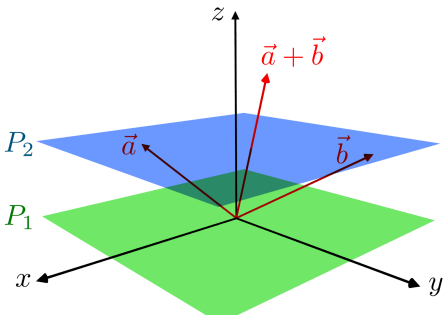


Figure 7: Affine space

Example. The following sets are convex:

- Hyperplans: $K = \left\{ x \in \mathbb{R}^d \,\middle|\, a^\top x = b, a \neq 0, b \in \mathbb{R} \right\}$
- Half spaces: $K = \left\{ x \in \mathbb{R}^d \,\middle|\, a^\top x \geqslant b, a \neq 0, b \in \mathbb{R} \right\}$
- Affine subspaces: $K = \left\{ x \in \mathbb{R}^d \,\middle|\, Ax = b, A \in \mathscr{M}_d(\mathbb{R}), b \in \mathbb{R} \right\}$
- Balls: $\left\{ x \in \mathbb{R}^d \,\middle|\, \|x\| \leqslant R \right\}$
- Cones: $K = \left\{ (x, r) \in \mathbb{R}^{d+1} \,\middle|\, \|x\| \leqslant r \right\}$
- Convex polytopes: intersections of half spaces

We know ennounce useful properties of convex sets.

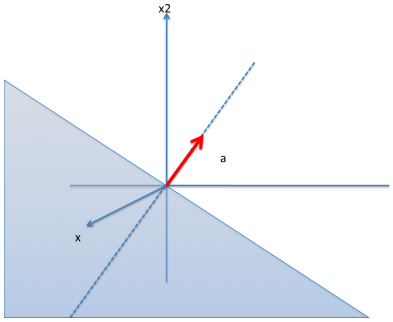


Figure 8: Half-space

Property 7 (Stability by intersection)**.** Convexity is stable by intersection. If $(K_i)_{i \in I}$ is a collection – non-necessarily countable – of convex sets, then:

$$\bigcap_{i \in I} K_i \quad \text{is convex}$$

Property 8 (Stability by affine transformation)**.** Convexity is stable by affine transformation. If $K \subseteq \mathbb{R}^d$ is a convex set, then for all $\lambda \in \mathbb{R}$ and $\beta \in \mathbb{R}^d$,

$$\lambda \cdot K + \beta := \{ \lambda \cdot x + \beta \mid x \in K \} \quad \text{is convex}$$

Property 9 (Convex serparation)**.** If C, D are disjoints convex sets (i.e. $C \cap D = \varnothing$), then there exists a hyperplane which separates C and D:

$$\exists a \neq 0, b \in \mathbb{R}, \quad C \subseteq \left\{ x \in \mathbb{R}^d \,\middle|\, a^\top x \geqslant b \right\}, D \subseteq \left\{ x \in \mathbb{R}^d \,\middle|\, a^\top x \leqslant b \right\}$$

Moreover, the inequalities are strict if C and D are compact.

When a set is not convex, a trick is to use its convex hull to show some properties on it.

Definition (Convex Hull)**.** Let $A \in \mathbb{R}^d$. The *Convex Hull* of A, denoted $\mathrm{Conv}(A)$, is the smallest convex set that contains A. In other words,

$$\begin{aligned} \mathrm{Conv}(A) &:= \bigcap \left\{ B \in \mathbb{R}^d \,\middle|\, A \subseteq B, B \text{ convex} \right\} \\ &= \left\{ \sum_{i=1}^{p} \alpha_i z_i \,\middle|\, p \geqslant 1, \alpha \in \mathbb{R}_+^p, (z_1, \ldots, z_p) \in A^p, \sum_{i=1}^{p} \alpha_i = 1 \right\} \end{aligned}$$

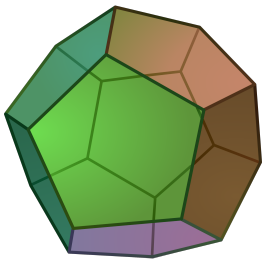


Figure 9: Polytope

3.2.3 Convex functions

Definition (Convex function)**.** A function $f : D \subseteq \mathbb{R}^d \to \mathbb{R}$ where D is convex is a *convex function* when:

$$\forall x, y \in D, \forall \alpha \in [0,1], \quad f(\alpha x + (1-\alpha)y) \leqslant \alpha f(x) + (1-\alpha) f(y) \tag{3.2.1}$$

Definition (Stricly convex function)**.** A function $f : D \subseteq \mathbb{R}^d \to \mathbb{R}$ where D is convex is a *strictly convex function* when:

$$\forall x, y \in D, \forall \alpha \in [0,1], \quad f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha) f(y) \tag{3.2.2}$$

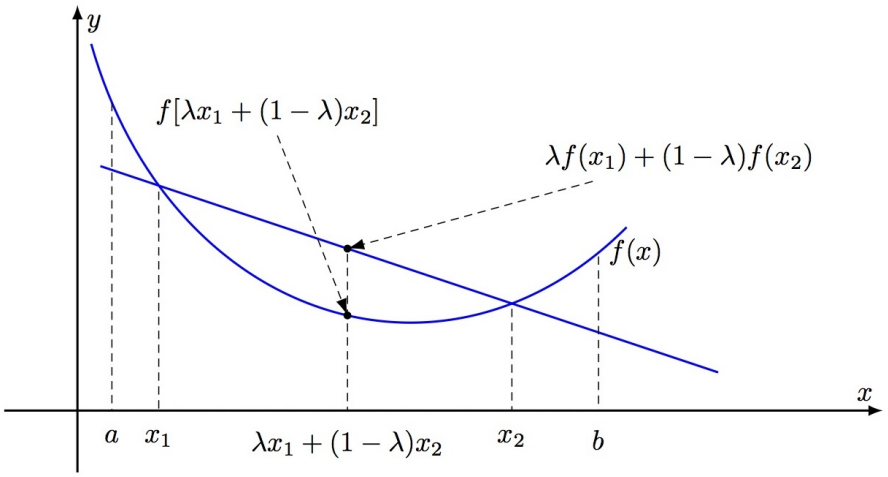


Figure 10: Geometric interpretation of convexity

Definition. A function $f : D \subseteq \mathbb{R}^d \to \mathbb{R}$ where D is convex is *μ-stricly convex* when

$$x \longmapsto f(x) - \frac{\mu}{2}\|x\|^2 \quad \text{is convex}$$

Example (Convex functions)**.** All the following functions are convex:

In dimension $d = 1$:

- $x \longmapsto x$
- $x \longmapsto x^2$
- $x \longmapsto -\log(x)$
- $x \longmapsto \log(1 + e^{-x})$
- $x \longmapsto |x|^p$ for $p \geqslant 1$
- $x \longmapsto -x^p$ for $p < 1$

Higher dimensions $d \geqslant 1$:

- Linear functions $x \longmapsto aa^\top x$
- Quadratic functions $x \longmapsto x^\top Q x$ for Q semidefinite symmetric positive matrix
- Norms
- $x \longmapsto \max\{ x_i \mid i \in [\![1, d]\!] \}$
- $x \longmapsto \log\left(\sum_{i=1}^d e_i^x\right)$

In practice, the functions which we will consider will be $\mathcal{C}^1$ and even $\mathcal{C}^2$. In this case, much simpler characterization are given for convex functions.

Property 10 (Differentiable convex functions)**.** If f is $\mathcal{C}^1$,

$$f \text{ convex} \iff \forall x, y \in D, f(x) \geqslant f(y) + f'(y)(x - y)$$

Property 11 (Twice differentiable convex functions)**.** If f is twice differentiable,

$$f \text{ convex} \iff \forall x \in D, \text{ its Hessian is semi-definite positive } (f''(x) \geqslant 0)$$

Property 12 (Operations which preserve convexity)**.** If $(f_i)_{i\in I}$ is a family of convex functions, then,

$$x \longmapsto \sup_{i\in I} f_i(x) \quad \text{is convex}$$

$$\forall \alpha_i \in \mathbb{R}_+, x \longmapsto \sum_{i\in I} \alpha_i f_i(x) \quad \text{is convex}$$

Furthermore, if f is convex on $C \times D$, then

$$y \longmapsto \inf_{x\in C} f(x,y) \quad \text{is convex on } D$$

Property 13 (Convexity and continuity)**.** If f is convex on D, then f is continuous on $\mathring{D}$. Furthermore, the epigraph of f,

$$\{\,(x,t) \in D \times \mathbb{R} \mid f(x) \leqslant t\,\} \quad \text{is convex.}$$

Property 14 (Jensen's inequality)**.** For f convex, $x_1, \ldots, x_n \in D$ and $\alpha_1, \ldots, \alpha_n \in \mathbb{R}_+$ such that $\sum_{i=1}^n \alpha_i = 1$. Then:

$$f\left(\sum_{i=1}^n \alpha_i x_i\right) \leqslant \sum_{i=1}^n \alpha_i f(x_i)$$

Property 15 (Jensen's integral inequality)**.** For f convex, $p(x) \geqslant 0$ over $S \subseteq D$ such that $\int_s p(x)\,\mathrm{d}x = 1$, then:

$$f\left(\int_S p(x)\ \mathrm{d}x\right) \leqslant \int_S p(x) f(x)\,\mathrm{d}x$$

Property 16 (Jensen's expectation inequality)**.** For f convex, and X a random variable such that $X \in D$ almost surely and with $\mathbb{E}[X] < +\infty$, then:

$$f(\mathbb{E}[X]) \leqslant \mathbb{E}[f(X)]$$

3.2.4 Unconstrained optimization problems

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. We consider the following minimization problem:

$$\inf_{x\in\mathbb{R}^d} f(x)$$

Note that the notation $\min_x f(x)$ is only used when the minimum is reached. If no point achieves the minimum, we use the notation $\inf_x f(x)$.

Three cases are possible, depending on $\inf_x f(x)$:

- $\inf_{x\in\mathbb{R}^d} f(x) = -\infty$ – there is no minimum.
- $\inf_{x\in\mathbb{R}^d} f(x) > -\infty$ and the minimum is not reached.
- $\inf_{x\in\mathbb{R}^d} f(x) > -\infty$ and the minimum is reached and equals $\min x \in \mathbb{R}^d f(x)$.

Definition (Local minimum)**.** Let $f : D \to \mathbb{R}$ and $x \in D$. x is a local minimum if and only if there exists an open set $V \subseteq D$ such that $x \in V$ and $f(x) = \min_{x'\in V} f(x')$.

Property 17. If f is convex, any local minimum of f is a global minimum.

Property 18. If f is stricly convex, it has at most one minimum.

Property 19. If f is convex and $\mathcal{C}^1$, then x is a minimum of f on $\mathbb{R}^d$ if and only if $f'(x) = 0$.

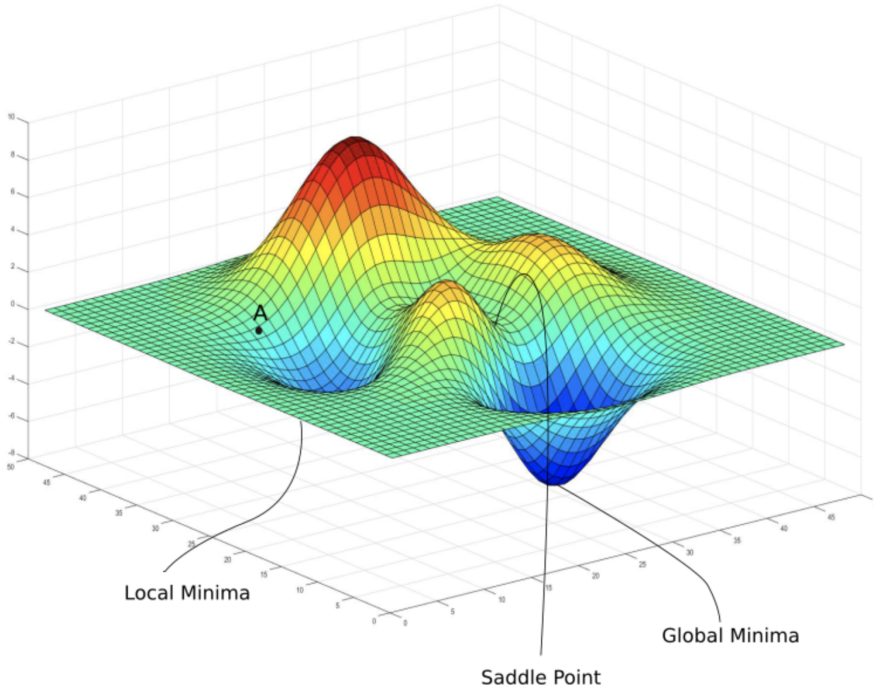


Figure 11: Local and global minima of a non-convex function

These three properties justify the apprach of canceling the gradient to efficiently find a solution to a minimization problem. Property 17 guarantees that the result of some gradient descent is a global minimum, and not just some dead-end for the descent path. Strict convexity guarantees that every converging gradient descent will end up at the same minimum.

In the next chapter, we will continue the analysis of this frequent minimization problem and develop formal tools and algorithm to solve it efficiently.

4 Convex analysis and convex optimization

4.1 Constrained optimization problems

Let $f : D \mapsto \mathbb{R}^d$ convex and $C \subseteq D$ convex. We consider the constrained minimization problem

$$\inf_{x \in C} f(x)$$

where C is the constraint set. It is often defined as the intersection of sets of the form $\{ h_i(x) = 0 \mid i \in I \}$ (hyperplanes) and $\{ g_j(x) \leqslant 0 \mid j \in J \}$ (half-spaces).

Example (Minimization of a linear function over a compact)**.** Let $A \subseteq \mathbb{R}^d$ be a compact, non-necessarily convex set, and $a \in \mathbb{R}^d \setminus \{0\}$. We can reformulate the non-convex minimization problem on A as a constrained convex optimization problem on $\mathrm{Conv}(A)$:

$$\min \left\{ a^\top x \,\middle|\, x \in A \right\} = \min \left\{ a^\top x \,\middle|\, x \in \mathrm{Conv}(A) \right\}$$

4.1.1 Lagrangian duality

A useful notion to solve constrained problems is Lagrangian duality. Assume that we are interested in the following constrained optimization problem:

$$\min_{x \in D} f(x) \quad \text{such that} \quad \begin{cases} h_i(x) = 0 & \text{for } i \in [\![1, m]\!] \\ g_j(x) \leqslant 0 & \text{for } j \in [\![1, r]\!] \end{cases} \tag{P}$$

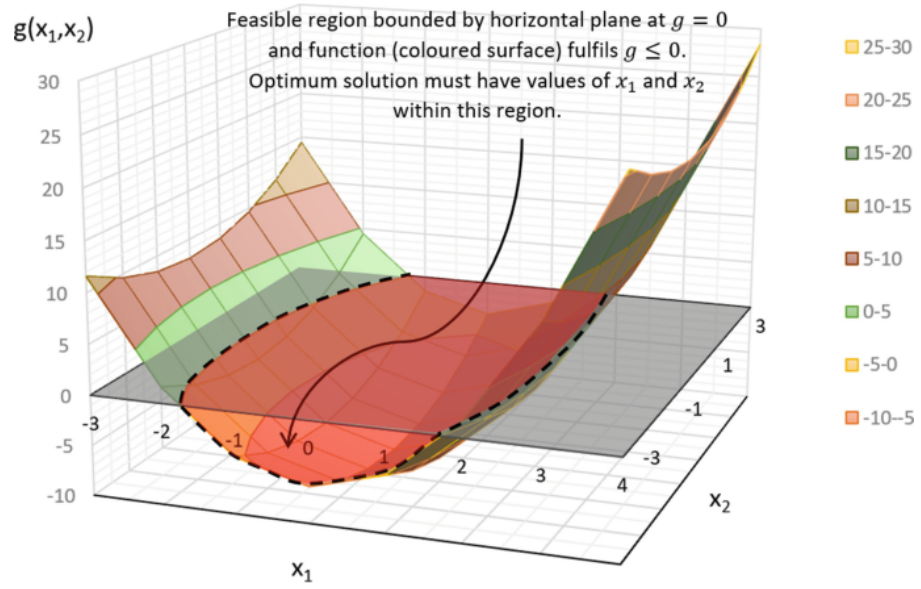


Figure 12: Constrained optimization: D^* is the red and orange part below the g plane

We denote by $D^* \subseteq D$ the set of points that satisfy the constraints:

$$D^* := \{\, x \in D \mid \forall i \in [\![1, m]\!], h_i(x) = 0 \ \wedge \ \forall j \in [\![1, r]\!], g_j(x) \leqslant 0 \,\}$$

Note that the equality constraints $h_i(x) = 0$ can be rewritten as inequalities:

$$h_i(x) \leqslant 0 \wedge -h_i(x) \leqslant 0$$

Unlike unconstrained optimization problems, canceling the gradient does not necessarily provide a solution for constrained optimization problems. The basic idea of Lagrangian duality is therefore to take the constraint D^* into account in the minimization problem by augmenting the objective function with a weighted sum of the constraint functions.

Definition (Lagrangian)**.** The *Lagrangian* associated to the optimization problem (P) is the function

$$\mathcal{L} : D \times \mathbb{R}^m \times \mathbb{R}^r_+ \longrightarrow \mathbb{R}$$

defines by:

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top h(x) + \mu^\top g(x)$$

Definition (Primal function)**.** We define the *primal function* associated to (P)

$$\bar{f} : D \longrightarrow \mathbb{R} \cup \{+\infty\}$$

by:

$$\bar{f} : x \longmapsto \sup_{\lambda \in \mathbb{R}^m,\, \mu \in \mathbb{R}^r_+} \mathcal{L}(x, \lambda, \mu) = \begin{cases} f(x) & \text{if } x \in D^* \\ +\infty & \text{otherwise} \end{cases}$$

Definition (Primal problem)**.** With these definitions, we observe that the optimization problem (P) can be re-written as a constraints-free problem of minimization of the primal function.

$$\begin{aligned} \inf_{x \in D^*} &= \inf_{x \in D} \bar{f}(x) \\ &= \inf_{x \in D} \sup_{\lambda \in \mathbb{R}^m,\, \mu \in \mathbb{R}^r_+} \mathcal{L}(x, \lambda, \mu) \end{aligned}$$

Definition (Dual problem)**.** The *Dual problem* is obtained by exchanging inf and sup in the primal problem:

$$\sup_{\lambda\in\mathbb{R}^m,\,\mu\in\mathbb{R}^r_+} f^*(\lambda,\mu) := \sup_{\lambda\in\mathbb{R}^m,\,\mu\in\mathbb{R}^r_+} \inf_{x\in D} \mathcal{L}(x,\lambda,\mu)$$

where

$$f^* : (\lambda,\mu) \longmapsto \inf_{x\in D} \mathcal{L}(x,\lambda,\mu)$$

is the *dual function*. If f is convex, this function is concave. Note that the dual of the dual is the primal.

The admissibility domain of the primal is $D^* := \left\{ x \in D \;\middle|\; \bar{f}(x) < +\infty \right\}$, and the admissibility domain of the dual is $C^* := \left\{ (\lambda,\mu) \in \mathbb{R}^m \times \mathbb{R}^r_+ \;\middle|\; f^*(\lambda,\mu) > -\infty \right\}$. There is no solution to the optimization problem when $D^* = \varnothing$. The problem is unbounded when $C^* = \varnothing$.

4.1.2 Link between primal and dual problems

The primal and dual problems are not necessarily identical, but have a strong relationship. For any (λ,μ), $f^*(\lambda,\mu)$ provides a lower bound on the solution of (P). The dual problem finds the best lower bound.

Property 20 (Weak duality principle)**.**

$$d^* := \sup_{\lambda\in\mathbb{R}^m,\,\mu\in\mathbb{R}^r_+} \inf_{x\in D} \mathcal{L}(x,\lambda,\mu) \leqslant \inf_{x\in D} \sup_{\lambda\in\mathbb{R}^m,\,\mu\in\mathbb{R}^r_+} \mathcal{L}(x,\lambda,\mu)$$

The solution of the dual problem is therefore always smaller than the solution of the primal. A good mnemonic is to remember this inequality as "the largest dwarf is always smaller than the smallest giant".

Definition (Dual gap)**.** The *dual gap* of the optimization problem is the difference between the primal and dual solutions:

$$p^* - d^* \geqslant 0$$

Definition (Strong duality)**.** There is *strong duality* when $p^* = d^*$. In this case, the two problems are equivalent – they share the same solutions. The existence of the solultions are related with the existence of saddle point of the Lagrangian. Note that strong duality does not always hold.

4.1.3 Strong duality

Sometimes, the dual problem is easier to solve than the primal problem. It is then useful to know if there is strong duality.

Definition (Strictly feasible point)**.** A point $x_0 \in D$ is *strictly feasible* when:

$$\exists x_0 \in D, \quad \begin{cases} h_i(x_0) = 0 & \forall i \in [\![1,m]\!] \\ g_j(x_0) < 0 & \forall j \in [\![1,r]\!] \end{cases}$$

Theorem (Slater's condition)**.** If f and D are convex, h_i are affine and g_j are convex, then the existence of a strictly feasible point implies strong duality.

Example. Let $D = \mathbb{R}^d_+$ and consider the following linear programming minimization problem:

$$\min_{x \in D,\ Ax=b} c^\top x$$

where $A \in \mathscr{M}_{m,d}(\mathbb{R})$ and $b \in \mathbb{R}^m$. The constraints can be written as $Ax - b = 0$. Therefore, the Lagrangian is:

$$\mathcal{L} : (x, \lambda) \in \mathbb{R}^d_+ \times \mathbb{R}^m \longrightarrow c^\top x + \lambda^\top (b - Ax)$$

The primal problem can be rewritten using the Lagrangian:

$$\begin{aligned} \min_{x \in D,\ Ax=b} c^\top x &= \min_{x \in D} \sup_{\lambda \in \mathbb{R}^m} c^\top x + \lambda^\top (b - Ax) \\ &= \min_{x \in D} \sup_{\lambda \in \mathbb{R}^m} b^\top \lambda + x^\top (c - A^\top \lambda) \end{aligned}$$

The problem is convex since the objective function is convex, and the equality constraints are affine. By Slater's condition, we can swap the min and the sup, giving:

$$\begin{aligned} \min_{x \in D,\ Ax=b} c^\top x &= \sup_{\lambda \in \mathbb{R}^m} \min_{x \in D} c^\top x + \lambda^\top (b - Ax) \\ &= \sup_{\lambda \in \mathbb{R}^m,\ A^\top \lambda \leqslant c} b^\top \lambda \end{aligned}$$

This is the dual formulation of the problem.

4.1.4 Karush-Kuhn-Tucker optimality conditions

We will now see conditions playing the same role as gradient cancelling for unconstrained optimization problems. These conditions will be useful to find equations to compute analytically the solutions of the minimization problem.

Assume that the function f, h_i, and g_i are all differentiable. Let x^*, and (λ^*, μ^*) be any primal and dual solutions, and assume that there is strong duality.

Property 21 (KKT1)**.** Since x^* minimizes $\mathcal{L}(x, \lambda^*, \mu^*)$ over x, its gradient must be canceled at x^*:

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla h(x^*) + \sum_{j=1}^{r} \mu_j \nabla g_j(x^*) = 0 \tag{KKT1}$$

Property 22 (KKT2)**.** Since $x^* \in D^*$ and $(\lambda^*, \mu^*) \in C^*$ are feasible we have:

$$\begin{aligned} \forall i \in [\![1, m]\!], \quad & h_i(x^*) = 0 \\ \forall i \in [\![1, r]\!], \quad & g_j(x^*) \leqslant 0 \\ \forall j \in [\![1, r]\!], \quad & \mu_j^* \geqslant 0 \end{aligned} \tag{KKT2}$$

Property 23 (KKT3)**.** The *complementary condition* holds, i.e.:

$$\forall j \in [\![1, r]\!], \quad \mu_j^* g_j(x^*) = 0 \tag{KKT3}$$

Proof. By contradiction, if the complementary condition does not hold, we could improve μ^* by setting $\mu_j^* = 0$, since $g_j(x^*) \leqslant 0$ and (λ^*, μ^*) maximizes $\mathcal{L}(x^*, \lambda, \mu)$. □

These conditions are called the Karush-Kuhn-Tucker (KKT) conditions. When the primal problem is convex, these conditions are also sufficient.

Theorem (Karush–Kuhn–Tucker theorem)**.** If there is strong duality, then:

$$\text{(KKT) conditions are satisfied} \iff \begin{cases} x^* & \text{is a solution to the primal problem} \\ (\lambda^*, \mu^*) & \text{is a solution of the dual problem} \end{cases}$$

The KKT conditions play an important role in optimization. In some cases, it is possible to solve them analytically. Many optimization methods are conceived for solving the KKT conditions.

4.2 Optimization algorithms for unconstrained convex optimization

In this section we will see two widely used optimization algorithms for the problem of unconstrained optimization, *Gradient descent* and *Stochastic Gradient descent*.

Since the goal of minimizing a function $f : \mathbb{R}^d \to \mathbb{R}$ for $d \in \mathbb{N}$ is to find the point x for which the function has minimum value, the fundamental idea behind gradient descent consists in strating from a given $x_0 \in \mathbb{R}^d$ and finding the next point following a descet direction iteratively. In particular, we will consider f to be a convex function.

4.2.1 Gradient Descent

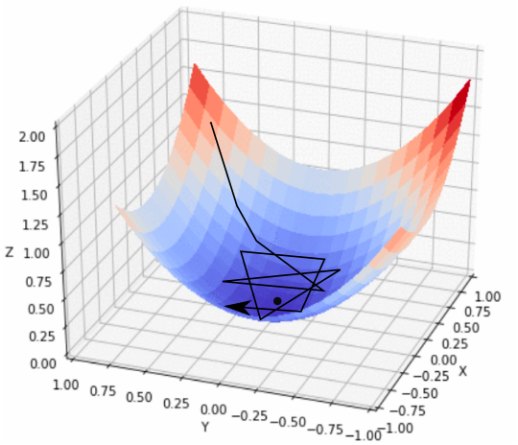


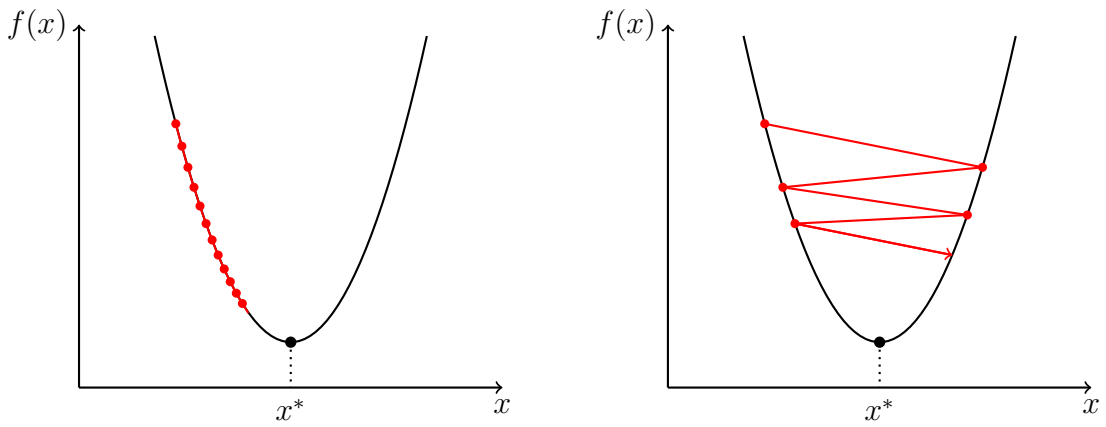
Figure 13: Convex gradient descent

When f is diffenrentiable, the gradient of f in x, denoted $\nabla f(x)$, determines the direction of maximum increase of the function in a suitable neighborhood of x – and so $-\nabla f(x)$ determines the direction of maximum decrease of the function. The gradient descent algorithm then reads as follows:

$$\begin{cases} x_0 & \text{given} \\ x_{t+1} := x_n - \gamma_t \nabla f(x_t) & \forall t \in [\![1, T]\!] \end{cases}$$

where γ_n is denoted as *step-size* and is small enough such that $-\gamma_t \nabla f(x_t)$ is still a decrease direction in the neighborhood of x_t.

The choice of γ_t is crucial for the optimization algorithm. If γ_t is too big, it makes the algorithm unstable and possibly divergince, since it follows the direction $-\nabla f(x_t)$ out of the region where it is a descent direction. On the other hand, if γ_t is too small, the chosen direction is a descent direction, but each step is very short, leading to a larger number of steps required to arrive to the minimum solution – with a big impact on the total computational complexity.



Too small: converges very slowly

Too big: overshoots and diverges

Figure 14: Choice of step size

We will prove in the next theorem that there exists a step-size that guarantees the convergence of the solution of the gradient descent algorithm to the minimizer of f, and we characterize how fast gradient descent achieves it.

4.2.2 Gradient Descent for strongly convex functions

Definition (L-Lipschitz continuous gradients)**.** For $L > 0$, f has L-Lipschitz continuous gradients when:

$$\forall x, y, \in \mathbb{R}^d, \quad \|\nabla f(x) - \nabla f(y)\| \leqslant L\|y - x\|$$

Lemma 1. Let $L > 0$ and $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function with L-Lipschitz continuous gradients, then:

$$f(y) \leqslant f(x) + \nabla f(x)^\top (y - x) + L\|y - x\|^2 \tag{4.2.1}$$

Proof. □

Remark. *Using a different argument, it is possible to prove the tighter result:*

$$f(y) \leqslant f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|^2 \tag{4.2.2}$$

Lemma 2 (Gradient descent is a descent algorithm with $\gamma_t \in]0, 1/L[$)**.** Let f be convex with L-Lipschitz gradient, let $x_0 \in \mathbb{R}^d$ and $\gamma_t > 0$, then:

$$\forall t \in \mathbb{N}^*, \quad f(x_t) \leqslant f(x_{t-1}) - \gamma_t(1 - L \cdot \gamma_t) \cdot \|\nabla f(x_{t-1})\|^2$$

In particular, if $\gamma_t \in]0, 1/L[$, we have that $\gamma_t(1 - L \cdot \gamma_n) > 0$. Therefore, for all $t \in \mathbb{N}^*$:

$$f(x_t) < f(x_{t-1})$$

whenever x_{t-1} is not a global optimum, otherwise $x_t = x_{t-1}$ and $f(x_t) = f(x_{t-1})$ (a fixpoint is reached).

Proof. □

Recall that when f is μ-stricly convex, we have:

$$f(y) \geqslant f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2}\|y - x\|^2$$

This will be used in the next theorem.

Theorem. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a μ-stricly convex function with L-Lipschitz continuous gradients. Let $\gamma \in]0, 1/2L[$, and choose a constant step-size $\gamma_t = \gamma$ for $t \in \mathbb{N}$. Denote by x^* the global optimum of f. Finally, let $x_0 \in \mathbb{R}^d$ and $T \in \mathbb{N}$. Then:

$$\|x_T - x^*\|^2 \leqslant (1 - \gamma\mu)^T \|x_0 - x^*\|^2$$

Proof. □

Remark. *Using the tighter inequality* (4.2.2)*, the previous result can be extended to* $\gamma \in]0, 1/L[$.

Note that since:

$$(1 - \gamma\mu)^T \leqslant e^{-\gamma\mu T}$$

the choice of $T = \frac{\gamma\mu}{\log}(\|x_0 - x^*\|^2/\varepsilon)$ gives:

$$\|x_T - x^*\|^2 \leqslant \varepsilon$$

Corollary. Under the same hypothesis on f, γ, we have:

$$\|x_T - x^*\| \xrightarrow[T \to +\infty]{} 0$$

Therefore, in this case, the Gradient Descent algorithm converges.

4.2.3 Stochastic Gradient Descent

In this section, we will introduce Stochastic Gradient Descent. In some cases, especially when n is big, computing the actual gradient of the empirical risk function $\hat{\mathcal{R}}_n$ is extremely costly. The approach of SGD is therefore to replace the exact gradient by a stochastic approximation of it, in practice calculating it from a random subset of the training data. This enables faster iterations, at the cost of a slower convergence rate.

The algorithm is useful to find the global minimum of conex functions of the form

$$f(x) = \mathbb{E}_\theta\, g(x,\theta), \quad \text{where} \quad \mathbb{E}_\theta g(x,\theta) = \int_\Omega g(x,\theta)\,\mathrm{d}p(\theta)$$

where p is a probability distribution over Ω and $g : \mathbb{R}^d \times \Omega \to \mathbb{R}$.

Example (Empirical Risk Minimization)**.** We consider f of the form:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} L(x,\theta_i), \quad \text{where} \quad \begin{cases} L(x,\theta_i) = \ell(x^\top z_i, y_i) \\ \theta_i = (z_i, y_i) \end{cases}$$

In this context, we can assume that due to the size of Ω (in the previous example, n), we are not able to evaluate the integral above, but we assume that we are able to sample some θ_t from p and to compute the gradient $\nabla_x g(x,\theta_t)$.

The algorithm is similar to classical gradient descent, but the iteration step is changed to:

$$x_t = x_{t-1} - \gamma_t \nabla_x g(x,\theta_t) \quad \text{where} \quad \theta_t \sim p$$

As previously, γ_t is a sequence of step sizes and x_0 is given; moreover, specifically for stochastic GD, θ_t is independently and identically distributed according to p.

Note that, by linearity of the integral $\nabla f(x) = \mathbb{E}_\theta \nabla_x g(x,\theta)$, so:

$$\begin{aligned} \mathbb{E}_\theta x_t &= x_{t-1} - \gamma_t \nabla_x g(x_{t-1},\theta_t) \\ &= x_{t-1} - \gamma_t \nabla f(x_{t-1}) \end{aligned}$$

Intuitively, the SGD seems to behave in expectation like gradient descent. Let's analyze this property in more details with the following theorem.

Definition (Variance of the estimator of the gradient)**.** We define the *variance of the estimator of the gradient* as:

$$\sigma^2(x) := \mathbb{E}_\theta \|\nabla f(x) - \nabla_x g(x,\theta)\|^2$$

Theorem (Expectation bound for SGD)**.** Assuming that $f : \mathbb{R}^d \to \mathbb{R}$ is μ-stricly convex and with L-Lipschitz continuous gradients, let $\gamma_t = \gamma$ for $t \in \mathbb{N}$, with $\gamma \in]0, 1/2L[$. Assume that there exists σ_*^2 such that

$$\forall x \in \mathbb{R}^d, \quad \sigma^2(x) \leqslant \sigma_*^2$$

Under such assumptions, we have that:

$$\mathbb{E}_{\theta_1,\dots,\theta_T}\|x_T - x^*\|^2 \leqslant (1-\mu\gamma)^T \|x_0 - x^*\|^2 + \frac{\gamma}{\mu}\sigma^2$$

Proof. □

Corollary (Convergence of SGD)**.** Under the same assumptions:

$$\mathbb{E}_{\theta_1,\dots,\theta_T}\|x_T - x^*\|^2 \xrightarrow[T\to+\infty]{} 0$$

Hence, stochastic gradient descent converges in expectation towards the global minimum.

5 Kernels

5.1 Introduction to kernels

In this course, we often focused on prediction methods which are *linear*, that is, the input data are vectors and the prediction function is linear (e.g. $f(x) = w^\top x$ for $w \in \mathbb{R}^d$). In this situation with given data (x_i, y_i), the vector w can be obtained by minimizing

$$\hat{L}(w) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, w^\top x_i) + \lambda\Omega(w)$$

Classical examples are logistic regression or least-squares regression. These methods look at first sight of limited practical significance, because input data may not be vectors, and relevant prediction functions may not be linear.

The goal of kernel methods is therefore to go beyond these limitations while keeping the good aspects. The underlying principle is to replace x by a function $\varphi(x) \in \mathbb{R}^d$, *explicitly* or *implicitly*, and consider linear predictions in $\Phi(x)$, i.e. $f(x) = w^\top\varphi(x)$. We call $\varphi(x)$ the *feature* associated to x.

Example (Linear regression)**.** In the case of linear regression, $\varphi(x) = x$ for $x \in \mathbb{R}^d$. As expected, this gives us linear models:

$$f(x) = w^\top x = \sum_{j=1}^{d} w_j x_j$$

Example (Polynomial regression of degree r)**.** With $x \in \mathbb{R}$, we have $\varphi(x) \in \mathbb{R}^{r+1}$ defined by:

$$\varphi(x) = (1, x, x^2, \dots, x^r)$$

Therefore, the prediction functions will be general polynomials of degree at most r:

$$f(x) = w^\top\varphi(x) = \sum_{j=0}^{r}(\varphi(x))_j = \sum_{j=1}^{r} w_j x^j$$

Example (Polynomial multivariate regression of degree r)**.** We consider $x \in \mathbb{R}^d$ and

$$\varphi(x) = (x_1^{\alpha_1}, \dots, x_d^{\alpha_d})$$

with $\sum_{i=1}^{d}\alpha_i = r$. In this situation, $p = \binom{d+r-1}{r}$ might be too big for an explicit representation to be feasible;

Example (Generic set of functions)**.** Let $\phi_1, \phi_r : \mathbb{R}^d \to \mathbb{R}$ be a set of functions of interest (e.g. a subset of the Fourier basis); we define $\varphi(x) = (\phi_1(x), \dots, \phi_r(x))$ to have:

$$f(x) = w^\top\varphi(x) = \sum_{j=1}^{r} w_j\phi_j(x)$$

5.2 Representer theorem

5.2.1 Theorem statement

Given a dataset $x_1, \dots, x_n$, let φ be a feature map, such that we are able to compute the observed feature maps $\varphi(x_1), \dots, \varphi(x_n)$. We can ask ourselves if there exists an easier representation for

w in terms of these observed feature maps, i.e. we want to know if it is possible to characterize the minimum $\hat{w}$ of:

$$\hat{L}(w) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, w^\top \varphi(x_i)) + \lambda w^\top w$$

in the form of $\hat{w} = \sum_{i=1}^{n} \alpha_i \varphi(x_i)$, with $\alpha_i \in \mathbb{R}$. The following theorem guarantees such characterization under basic properties of $\hat{L}$.

Theorem (Representer theorem)**.** Let $\varphi : \mathcal{X} \to \mathbb{R}^d$. Let $(x_1, \ldots, x_n) \in \mathcal{X}^n$ and assume that $\Psi : \mathbb{R}^{n+1} \to \mathbb{R}$ is strictly increasing with respect to the last variable. Then, the minimum of

$$\hat{L}(w) := \Psi(w^\top \varphi(x_1), \ldots, w^\top \varphi(x_n), w^\top w)$$

is obtained for

$$w = \sum_{i=1}^{n} \alpha_i \varphi(x_i)$$

for some $\alpha \in \mathbb{R}^n$.

Proof. □

Corollary. For $\lambda > 0$,

$$\min_{w \in \mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, w^\top \varphi(x_i)) + \frac{\lambda}{2} w^\top w$$

is obtained for

$$w = \sum_{i=1}^{n} \alpha_i \varphi(x_i).$$

Note that there is no assumption on ℓ, and in particular, no convexity assumption. This result is extendable to Hilbert spaces (RKHS), as we will see in the next section.

5.2.2 Finite dimensional representation of the learning problem

Using the representer theorem, we know that the minimum of $\hat{L}$ is of the form $w = \sum_{i=1}^{n} \alpha_i \varphi(x_i)$; we can therefore directly optimize this characterization, i.e. we can then write:

$$\min_{w \in \mathbb{R}^r} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, w^\top \varphi(x_i)) + \frac{\lambda}{2} w^\top w = \min_{\alpha \in \mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, (K\alpha)_i) + \frac{\lambda}{2} \alpha^\top K \alpha$$

where K is an $n \times n$ matrix with values

$$K_{i,j} = \varphi(x_i)^\top \varphi(x_j).$$

Indeed,

$$\varphi(x_i)^\top w = \sum_{j=1}^{n} \alpha_j \varphi(x_i)^\top \varphi(x_j) = (K\alpha)_i$$

moreover,

$$||w||^2 = w^\top w = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \varphi(x_i)^\top \varphi(x_j) = \alpha^\top K \alpha$$

We finally have a closed form representation for the function evaluation. Defining the *kernel function* $k(x, x') := \varphi(x)^\top \varphi(x')$, we have:

$$f(x) = w^\top \varphi(x) = \sum_{i=1}^{n} \alpha_i \varphi(x_i)^\top \varphi(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x).$$

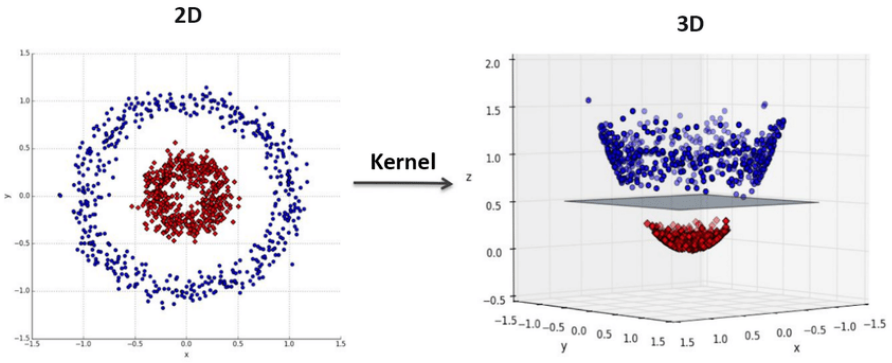


Figure 15: Use of a kernel to linearly separate data

Remark (Kernel trick)**.** *The whole learning problem can be written in terms of the kernel k; indeed, f depends only on k, $\hat{L}$ depends on K with $K_{i,j} = k(x_i, x_j)$. Therefore, we have the so-called* kernel trick*, i.e. we do not need to compute explicitly the features φ to be able to represent and solve the learning problem, we just need to be able to compute their inner product.*

Example (Power of the kernel trick with infinite dimensional feature maps)**.**
Consider $\mathcal{X} = [-1, 1]$ and the feature map

$$\varphi(x) = (1, x, x^2, \dots).$$

The resulting model space would have the form

$$f(x) = \sum_{j=0}^{+\infty} w_j x^j$$

with $\sum_{j=1}^{+\infty} w_j^2 < +\infty$. This model space is the set of analytic function on $\mathcal{X}$, which is a very rich space. In particular, it is dense in the space of continuous functions. However, it is not possible to compute $\varphi(x)$ explicitly since it is infinite dimensional. The kernel trick provides an elegant way to compute the solution of the learning problem in closed form; indeed, the inner product can be computed in closed form in $O(1)$:

$$k(x, x') = \varphi(x)^\top \varphi(x') = \sum_{j=0}^{+\infty} x^j x'^j = \frac{1}{1 - xx'}$$

Therefore, the kernel trick allow to replace $\mathbb{R}^d$ by $\mathbb{R}^n$, which is interesting when d is very large. Furthermore, it allows to separate the representation problem (design a kernel on a set $\mathcal{X}$), algorithms, and analysis (which only use the kernel matrix K).

5.3 Properties of kernels

Since the learning problem is completely defined in terms of the kernel function, the explicit knowledge of the feature map is not required anymore. In particular, given a function $k : X \times X \to \mathbb{R}$, tu use it in a learning problem, we need to be sure that it is a *positive definite kernel*, i.e. that there exists a feature map φ such that

$$\forall x, x' \in \mathcal{X}, \quad k(x, x") = \varphi(x)^\top \varphi(x')$$

Kernel functions admits many characterizations, which we will now present.

Property 24 (Characterization in terms of positive-definitness)**.** k is a positive definite kernel if and only if the kernel matrix k is positive semi-definite (i.e. all its eigenvalues are non-negative).

Theorem (Aronsazjn)**.** k is a positive definite kernel if and only if there exists a Hilbert space $\mathcal{F}$, and $\varphi : X \to \mathcal{F}$ such that

$$\forall x, y \in X, \quad k(x, y) = \langle \varphi(x), \varphi(y) \rangle$$

If such objects exist, $\mathcal{F}$ is called the *feature space* and φ the *feature map.*

Property 25. The sum and product of kernels are kernels.

Example (Linear kernel)**.** The linear kernel corresponds to $\varphi = x \mapsto x$:

$$k(x, y) = x^\top y$$

Example (Polynomial kernel)**.** The kernel $k(x, y) = (x^\top y)^r$ can be expanded as:

$$k(x, y) = \left(\sum_{i=1}^{d} x_i y_i \right)^r = \sum_{\alpha_1 + \cdots + \alpha_p = r} \binom{r}{\alpha_1, \ldots, \alpha_p} \underbrace{(x_1 y_1)^{\alpha_1} \ldots (x_p y_p)^{\alpha_p}}_{(x_1^{\alpha_1} \ldots x_p^{\alpha_p})(y_1^{\alpha_1} \ldots y_p^{\alpha_p})}$$

Example (Translation-invariant kernels on a bounded interval)**.**

Example (Translation-invariant kernels on $\mathbb{R}^d$)**.**

6 Elements of Statistical Machine Learning

In this calss we will introduce the main elements of PAC Learning. Probably Apporximately Correct Learning is a theoretical and mathematical framework for analysing machine learning algorithms. Such framework allows, for instance, to evaluate the complexity of a problem in the context of supervised learning. It was introduced by Leslie Valiant in 1984.

6.1 Introduction

Recall that a *learning algorithm* – or *learning rule* – is a function $\mathcal{A}$ that maps a training set D_n to an estimator $\hat{f}_n : \mathcal{X} \to \mathcal{Y}$:

$$\mathcal{A} : \bigcup_{n \in \mathbb{N}} (\mathcal{X} \times \mathcal{Y})^n \longmapsto \mathcal{Y}^{\mathcal{X}}$$

We denote $\hat{f}_n := \mathcal{A}(D_n)$ the estimator, which is a random variable in $\mathcal{Y}^{\mathcal{X}}$, since it depends on the dataset D_n.

Remark. *Sometimes, the prediction set can differ from the actual output set. For instance, binary classification in* $\{0, 1\}$ *can be replaced by predictions in* $[0, 1]$*. We used this technique previously by introducing Hinge and Logistic losses to extend the binary loss and obtain a convex optimization problem.*

Definition (Fundamental problem of Supervised Learning)**.** Given a dataset D_n, a loss function ℓ, the goal of supervised learning is to estimate f^* that satisfies:

$$f^* \in \operatorname*{arg\,min}_{f : \mathcal{X} \to \mathcal{Y}} R(f)$$

Recall the *excess risk*, defined in (2.4.3) by:

$$\mathscr{R}(\hat{f}_n) := \mathcal{R}(\hat{f}_n) - \mathcal{R}(f^*)$$

It measures how close given predictor $\hat{f}_n$ is to the best possible f^*, in terms of expected risk $\mathcal{R}$, i.e. in terms of *average error on new examples*. Both $\mathcal{R}(\hat{f}_n)$ and $\mathscr{R}(\hat{f}_n)$ are random variables, since $\hat{f}_n$ depends on the dataset D_n.

Definition (Consistency)**.** Let $\delta \in]0, 1]$. The algorithm $\mathcal{A}$ is *consistent* (i.e. it is a proper learning algorithm) when:

$$\lim_{n \to +\infty} \mathbb{E}_{D_n}[\mathscr{R}(\hat{f}_n)] = 0$$

Definition (Strong consistency)**.** An algorithm $\mathcal{A}$ is strongly consistent when the equation above holds with probability 1:

$$\lim_{n \to +\infty} \mathscr{R}(\hat{f}_n) = 0$$

Not all algorithms learn at the same rate: for a given $n \in \mathbb{N}^*$, some algorithms might have more precise predictions than others. We can define more quantitative versions of the requirements above, which are useful to characterize how precise are the predictions.

Definition (Learning rates)**.** The sequence $(e_n)_{n \in \mathbb{N}} \in \mathbb{R}_+^{\mathbb{N}}$ is a learning rate in expectation if:

$$\forall n \in \mathbb{N}, \quad \mathbb{E}_{D_n}[\mathscr{R}(\hat{f}_n)] \leqslant e_n$$

Given $\delta \in]0, 1]$, a sequence $(p_{n,\delta})_{n \in \mathbb{N}} \in [0, 1]^{\mathbb{N}}$ is a learning rate in probability if:

$$\forall n \in \mathbb{N}, \quad \mathbb{P}_{D_n}\left(\mathscr{R}(\hat{f}_n) > p_{n,\delta}\right) \leqslant \delta$$

6.2 Empirical Risk Minimization

A classical way to estimate f^* is via *empirical risk minimization*. This is the approach we used in the specific examples of linear and logistic regressions. We will now define it more formally.

Let $\mathcal{F}$ be a set of functions called *function space* containing some candidate estimators of choice. The estimator is defined as:

$$\hat{f}_n := \arg\min_{f \in \mathcal{F}} \hat{\mathcal{R}}_n(f) \quad \text{where} \quad \hat{\mathcal{R}}_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i))$$

As before, $\hat{\mathcal{R}}_n$ is the empirical risk, which measures the average error performed by f on the training set. The intuition is that $\hat{\mathcal{R}}_n(f)$ approximates $\mathcal{R}(f)$ – the expected error – increasingly better when n goes to infinity. A crucial question we need to address is to understand under which conditions *empirical risk minimization* is a learning algorithm and has learning rates.

6.3 Preliminary results

We will first recall some results that will be useful for the proof.

6.3.1 Results on random variables

Lemma 3 (Union bound)**.** Let $\mathcal{F}$ be a finite set indexing a family of sets $(A_f)_{f \in \mathcal{F}}$, then:

$$\mathbb{P}\left(\bigcup_{f \in \mathcal{F}} A_f\right) \leqslant \sum_{f \in \mathcal{F}} \mathbb{P}(A_f)$$

Lemma 4 (Supremum of random variables)**.** Let $t > 0$ and $\mathcal{F}$ be a finite set indexing real random variables $(u_f)_{f\in\mathcal{F}}$, we have:

$$\mathbb{P}(\sup_{f\in\mathcal{F}} |u_f| > t) \leqslant \sum_{f\in\mathcal{F}} \mathbb{P}(|u_f| > t)$$

Property 26. Let X be a random variable and let f, g be real functions, with $t \in \mathbb{R}$ such that $f(X) \geqslant g(X) > t$ almost surely, then:

$$\mathbb{P}(g(X) > t) \leqslant \mathbb{P}(f(X) > t)$$

6.3.2 Bernstein's inequality

Lemma 5 (Exponential moments of bounded random variables)**.** Let u be a random variable such that $|u| \leqslant B$, for $B > 0$ almost surely and $\mathbb{E}[u] = 0$. Define $\sigma^2 := \mathbb{E}[u^2]$. Let $0 \leqslant \theta < B^{-1}$, then:

$$\mathbb{E}[e^{\theta u}] \leqslant \exp\left(\frac{\theta^2\sigma^2}{2(1-\theta B)}\right)$$

Lemma 6 (Bernstein inequality for random variables)**.** Let $U_1, \ldots, U_n$ be independently and identically distributed random variables, such that $\mathbb{E}[u] = 0$ and $|u| \leqslant B$ almost surely, for $B > 0$. Define $\sigma^2 := \mathbb{E}[u^2]$. Let $t > 0$. The following holds:

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} U_i > t\right) \leqslant \exp\left(-\frac{t^2 n}{2(\sigma^2 + Bt)}\right) \tag{6.3.1}$$

Corollary. Under the same assumption as the previous lemmas,

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} U_i\right| > t\right) \leqslant 2\exp\left(-\frac{t^2 n}{2(\sigma^2 + Bt)}\right)$$

6.4 Consistency and Learning rates for Empirical Risk Minimization

6.4.1 Introduction

6.4.2 Bounding the variance term: PAC bounds

6.4.3 Bounds in probability

6.5 Bounding the Bias

7 k-Nearest Neighbours

7.1 Introduction

7.1.1 Goal

We would like to classify objects, described with vectors $x \in \mathbb{R}^d$, among $L+1$ classes $\mathcal{Y} := \{0, \ldots, L\}$, automatically. To do so, we have at hand a labelled data set of n data points $(x_i, y_i) \in \mathbb{R}^d \times \mathcal{Y}$ for $i \in [\![1, n]\!]$, which we assume to be the realizations of i.i.d. random variables (X_i, Y_i) following a distribution ν. The goal of this lesson is to build a classifier, i.e. a function

$$g : \mathbb{R}^d \longrightarrow \mathcal{Y}$$

which minimizes the probability of mistakes:

$$\mathbb{P}_{(X,Y)\sim\nu}(g(X) \neq Y) = \mathcal{R}(g) := \mathbb{E}_{(X,Y)\sim\nu}[\mathbb{1}_{g(x)\neq Y}]$$

7.1.2 Intuition of the base algorithm

The k-nearest neighbours classifier works as follows. Given a new input $x \in \mathbb{R}^d$, the classifier analyzes the k nearest points x_i in the data set $D_n = \{ (x_i, y_i) \mid i \in [\![1, n]\!] \}$ and predicts a majority vote among them.

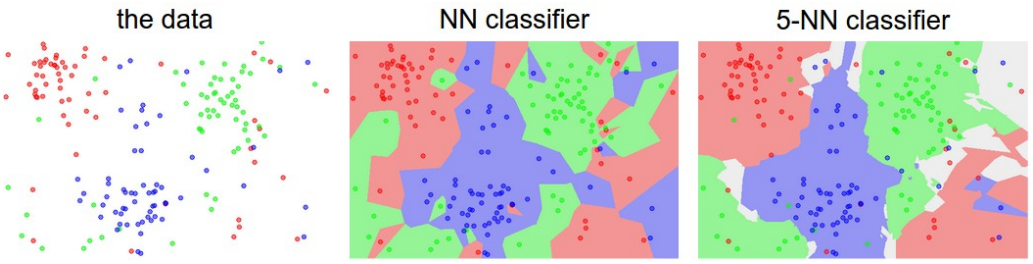


Figure 16: k-NN for different values of k

The k-nearest neighbor classifier is quite popular because it is simple to code and to understand; it has nice theoretical guarantess as soon as k is appropriately chosen and performs reasonably well in low dimensional spaces.

7.1.3 Assumptions and notations

8 Unsupervised Learning

As described in the introduction of this course, Machine Learning is divided into three main categories: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. In this chapter, we will analyze *Unsupervised Learning*: while Supervised Learning provided a dataset containing both features (x_i) and labels (y_i); in Unsupervised Learning, the dataset contains only features x_i.

8.1 K-means

K-means clustering is a method of vector quantization. K-means clustering is an algorithm of alternate minimization that aims at partitioning n observations into K clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype to the cluster.

8.1.1 Distortion

Definition (Observations)**.** In the clustering problem, we are given $x_i \in \mathbb{R}^p$ for $i \in [\![1, n]\!]$, the *observations* we want to partition.

Definition (Means)**.** For some given K, we denote $\mu_k \in \mathbb{R}^p$ for $k \in [\![1, K]\!]$ the means. μ_k is the center of the cluster k. We will denote μ the associated matrix.

Definition (Hard membership variables)**.** $z_i^k \in \{0, 1\}$ are indicator variables associated to x_i, such that:

$$z_i^k := \begin{cases} 1 & \text{if } x_i \text{ belongs to the cluster } k \\ 0 & \text{otherwise} \end{cases}$$

We denote by z the matrix which components are equal to z_i^k.

Definition (Distortion)**.** We define the distortion $J(\mu, z)$ by:

$$J(\mu, z) := \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k \|x_i - \mu_k\|^2 \tag{8.1.1}$$

Distortion measures the overall distance of the centers μ_k to the points of the cluster.

Definition (K-means optimization problem)**.** The *K-means optimization problem* consists in finding matrices μ and z which minimize the distortion:

$$\min_{\substack{\mu \in \mathscr{M}_{n,k}(\mathbb{R}) \\ z \in \mathscr{M}_{n,k}(\mathbb{R})}} J(\mu, z) = \min_{\substack{\mu \in \mathscr{M}_{n,k}(\mathbb{R}) \\ z \in \mathscr{M}_{n,k}(\mathbb{R})}} \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k \|x_i - \mu_k\|^2 \tag{8.1.2}$$

8.1.2 Algorithm

To minimize $J(\mu, z)$, we proceed using an alternating minimization: we will minimize J alternatively with respect to z and μ.

1. Randomly choose a vector μ.
2. Minimize J with respect to z:

$$z_i^k = \begin{cases} 1 & \text{if } \|x_i - \mu_k\|^2 = \min_s \|x_i - \mu_s\|^2 \\ 0 & \text{otherwise} \end{cases} \tag{8.1.3}$$

   This is equivalent to associating to x_i the nearest center μ_k.
3. Minmize J with respect to μ:

$$\mu_k = \frac{\sum_{i=1}^{n} z_i^k x_i}{\sum_{i=1}^{n} z_i^k} \tag{8.1.4}$$

4. Come back to step 2 until convergence.

Proof. The formula (8.1.4) can be obtained by minmizing the gradient of J with respect to μ_k:

$$\nabla_{\mu_k} J = -2 \sum_{i=1}^{n} z_i^k (x_i - \mu_k)$$

Therefore:

$$\nabla_{\mu_k} J = 0 \iff \mu_k = \frac{\sum_{i=1}^{n} z_i^k x_i}{\sum_{i=1}^{n} z_i^k}$$

□

8.1.3 Convergence and initialization: K-means++

We can show that this algorithm converges in a finite number of iterations. Nevertheless, the convergence could be towards to local minimum, introducing the problem of initialization.

A classic method is to use random restarts. It consists in choosing serveral random vectors μ, computing the algorithm for each case and finally keeping the partition wihch minimizes the distortion. Thus, we hope that at least one of the local minima is close enought to a global minimum.

Another well known method is the K-means++ algorithm, which aims at correcting a major theoretical shortcoming of the K-means algorithm: the approximation found can be arbitrarily bad with respect to the objective function compared to the optimal clustering.

The K-means++ algorithm addresses this obstacles by specifying a procedure to initialize the cluster centers before proceeding with the standard K-means optimization iterations. With the K-means++ initialization, the algorithm is guaranteed to find a solution that is $O(\log K)$ competitive to the optimal K-means solution.

The intuition behind this approach is that spreading the K initial cluster centers will improve the quality of the clusters found by the algorithm. At each iteration of the algorithm we will build a new center, and repeat this step until we have K centers. Here are the steps of the algorithm:

1. Choose the first center uniformly among the data points.
2. For each data point x_i of the data set, compute the distance between x_i and the nearest center that has already been chosen. We denote this distance $D_{\mu_t}(x_i)$ where μ_t is specified to recall that we are minimizing over the current chosen centers.
3. Choose a new data point at random as a new center, but now using a weighted probability distribution where a point x_i is chosen with probability proportional to $D_{\mu_t}(x_i)^2$.
4. Come back to step 2 until K centers have been chosen.

We see that we have now built K vectors with respect to our first intuition which was to well spread the centers – because we used a well-chosen weighted probability. We can now use those vectors as the initialization of our standard K-means algorithm.

8.1.4 Choice of K

It is important to point out that the choice of K is not universal. Indeed, we see that if we increase K, the distortion J decreases, until reaches 0 when $K = n$ – when each data point is the center of its own cluster. To address this issue, one solution could be to add to J a penalty over K. Usually, it takes the following form:

$$J(\mu, z, K) := \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k \|x_i - \mu_k\|^2 + \lambda K$$

Once again, the choice of the penalty coefficient λ is arbritrary. There is no general way to choose K.

8.1.5 Other problems

We can also point out that K-means will work properly when the width of the different clusters are similar – for example when dealing with spheres. But clustering by K-means could also be disappointing in some cases such as the following example:

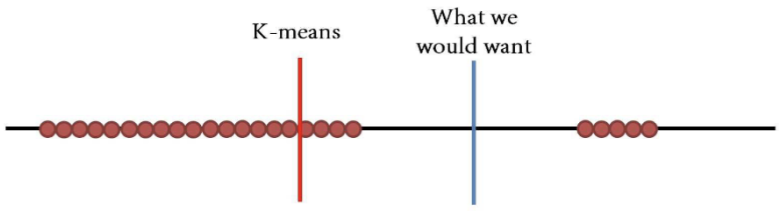


Figure 17: Example where K-means does not provide a satisfactory clustering result

For instance, the use of Gaussian mixtures provides a way to avoid this problem.

8.2 Dimensionality reduction: principal component analysis

Imagine that we want to process some video data $(x_i)_{i\in[\![1,n]\!]}$ where $x_i \in \mathbb{R}^p$. In the case of video data, p can be prohibitively huge to manipulate, e.g. $p \simeq 10^{11}$. We would like to reduce the dimensionality of the origin space, i.e. finding a new representation for each of the x_i:

$$x_i \in \mathbb{R}^p \longmapsto \hat{x}_i \in \mathbb{R}^{p'} \quad \text{where} \quad p' \ll p$$

without loosing too much information.

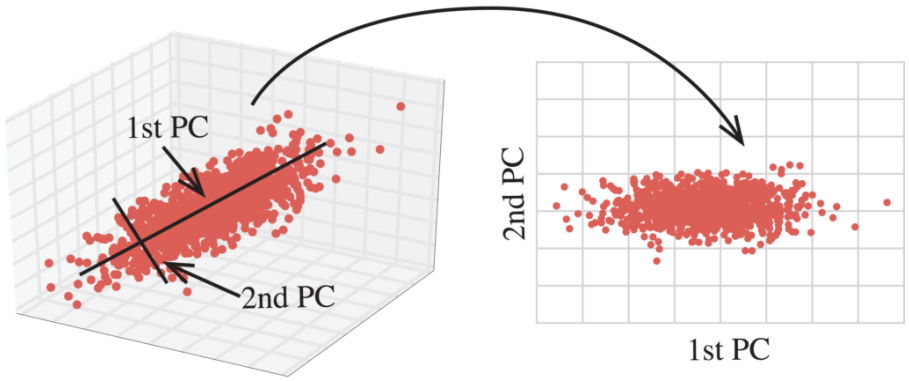


Figure 18: Dimensionality reduction from $\mathbb{R}^3$ to $\mathbb{R}^2$, i.e. using two principal components

8.2.1 Analysis view

We will start by introducing formally the problem. Assume that there is some data distribution ν over $\mathbb{R}^p$. A random variable $X \in \mathbb{R}^p \sim \nu$ samples this data distribution over this high-dimensional space. Assume that $\mathbb{E}[X] = 0$, and let

$$\Sigma := \mathrm{Cov}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^\top\right] = \mathbb{E}[XX^\top]$$

Σ is the covariance matrix of X; note that it is definite non-negative.

The intuition is that we want to preserve as much "diversity" in the output after the dimension reduction. This leads to the following definition of the problem in the case where $p' = 1$.

Definition (PCA problem)**.** Find $w \in \mathbb{R}^p$ such that:

$$w \in \underset{\|w\|=1}{\arg\max}\, \mathbb{V}(w^\top X) \tag{8.2.1}$$

Theorem. w verifies (8.2.1) exactly when w is the eigenvector of Σ corresponding to the largest eigenvalue.

Proof. Note that:

$$\begin{aligned}
\mathbb{V}(\omega^\top X) &= \mathbb{E}[\omega^\top X]^2 - (\mathbb{E}[\omega^\top X])^2 \\
&= \mathbb{E}[(\omega^\top X)(\omega^\top X)] \\
&= \mathbb{E}[\omega^\top X X^\top \omega] \\
&= \omega^\top \mathbb{E}[XX^\top]\,\omega \\
&= \omega^\top \Sigma\, \omega
\end{aligned}$$

Therefore,

$$\sup_{\|\omega\|=1} \mathbb{V}(\omega^\top X) = \sup_{\|\omega\|=1} \omega^\top \Sigma\, \omega$$

Thus, by property, ω is exactly the eigenvector of Σ corresponding to the largest eigenvalue. We can write $\Sigma = UDU^\top$ with $U \in \mathbb{R}^{p\times p}$, and $UU^\top = I_p$, with $D = \text{diag}(\lambda_1, \dots, \lambda_p)$. If we assume $\lambda_1 \geqslant \dots \geqslant \lambda_p$, we have that ω is the first column of U. □

Remark. *To generalize this procedure to p' dimensions, we take the p' eigenvectors associated to the p' largest eigenvalues. Another approach would be to use* deflation*, defined by the following algorithm:*

1. *Find w using the largest eigenvalue.*
2. *Project the data onto* $\text{Span}(w)$.
3. *Start at Step 1.*

8.2.2 In practice

In practice, we do not know Σ – since it depends on the unknown distribution ν – but we are given a dataset to work with. Given $(x_i)_{i\in[\![1,n]\!]}$, we want to find a vector $w \in \mathbb{R}^p$ – or a matrix $W \in \mathscr{M}_{p,p'}(\mathbb{R})$ – such that the associated projection maximizes the variance of the projected dataset. In the theoretical analysis, we assumed that $\mathbb{E}[X] = 0$; to verify the assumptions, we need the dataset to be centered, or *center* it if necessary. Thus, we introduce the mean of the dataset (a vector) and substract it to the initial dataset:

$$x_i \leftarrow x_i - \mu \quad \text{where} \quad \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

This being done, we can define the *empirical covariance matrix*, which is the approximation of the covariance matrix X with respect to the dataset available:

$$\hat{\Sigma} := \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top$$

Similarly to the theoretical setting, we can use the normlized eigenvectors associated to the largest eigenvalues to construct a matrix maximizing the variance.

9 Model-Based Machine Learning

10 Maximum Likelihood

11 MCMC Sampling

12 Neural Networks

13 Appendix

13.1 Complexity of linear algebra computations

In the following, we consider $K, L \in \mathscr{M}_n(\mathbb{R})$ to be two square matrices, and $y \in \mathbb{R}^n$ to be a vector. We give asymptotic computation time for different operations of linear algebra for simple algorithms.

Operation	**Notation**	**Complexity**
Matrix multiplication	KL	$O(n^3)$
Inversion	K^{-1}	$O(n^3)$
Vector multiplication	Ky	$O(n^2)$
Eigenvalues/vectors decomposition	$\mathrm{Sp}(K)$	$O(n^3)$
Largest eigenvector		$O(n^2)$