Introduction to Machine Learning

Alessandro Rudi, Umut Şimşekli
Notes by Antoine Groudiev

2nd April 2024

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.

Contents

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:

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

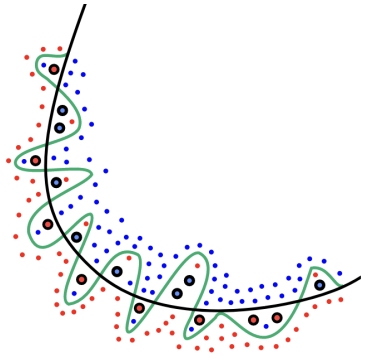

Figure 1: Example of overfitting

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.

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

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

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

[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).

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.

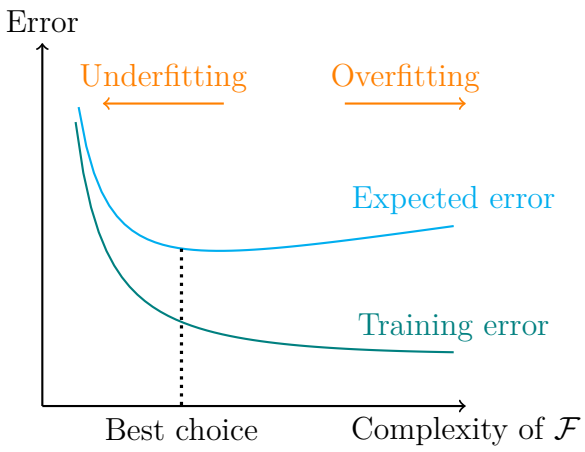

Figure 2: Overfitting and underfitting

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 $\theta \in \mathbb{R}^d$.

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 $Y = (Y_1, \ldots, Y_n)^\top \in \mathbb{R}^n$ 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 $\hat{\mathcal{R}}_n$:

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

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:

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

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

By minimizing $\|Y - X\theta\|$, we thus look for the element of $\operatorname{Im}(X)$ closest to Y. This is the orthogonal projection of Y on $\operatorname{Im}(X)$, 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 $\operatorname{Im}(X)$.

[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.

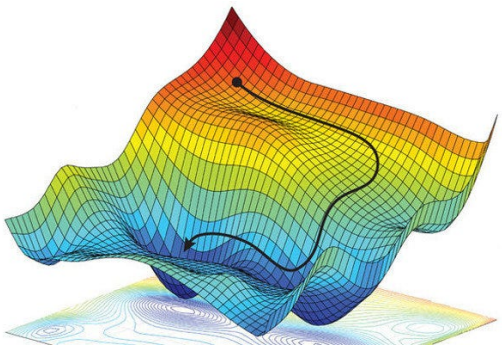

Figure 3: (Non-convex) Gradient descent

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:

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

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:

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

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

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

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. Conversly to 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.

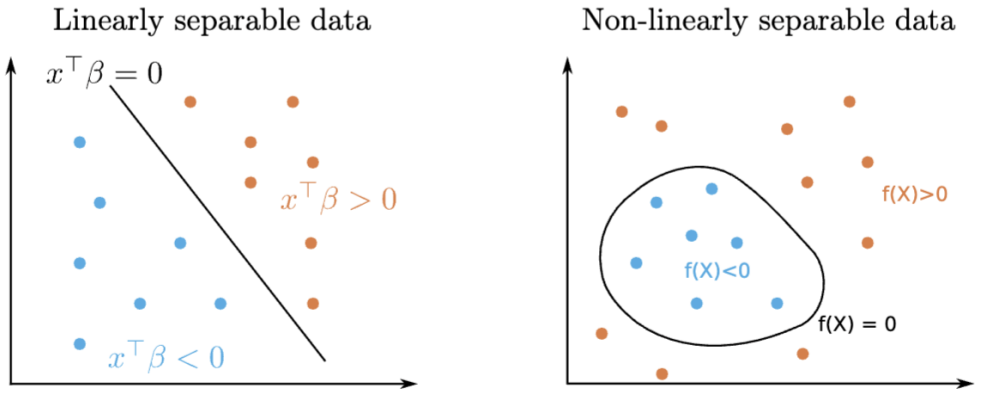

Figure 4: Data that can or cannot be well-explained by a linear separation

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.

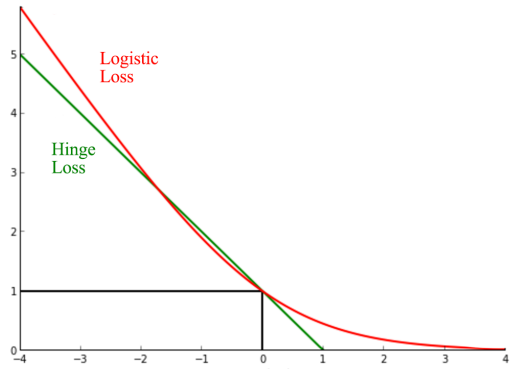

Figure 5: Binary, Hinge, and logistic loss for $y' = 1$

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:

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

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

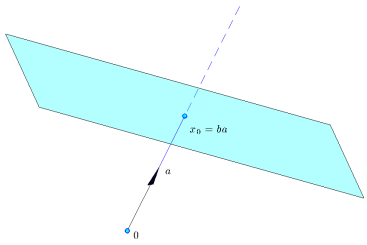

Figure 6: Hyperplane

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, ...).

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*:

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

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 $x, y \in K$, $[x, y] \subseteq K$.

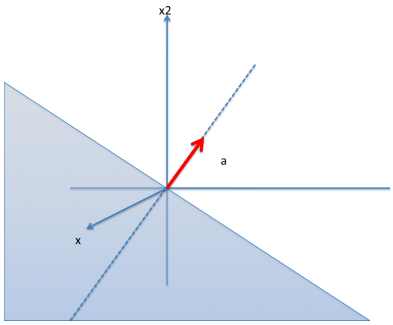

Figure 7: Half-space

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

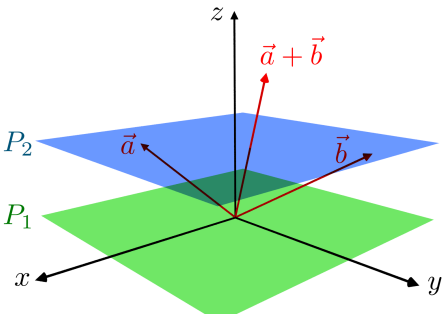

Figure 8: Half-space

We know ennounce useful properties of convex sets.

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 $\beta \in \mathbb{R}^d$,

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

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,

$$\begin{aligned} \mathrm{Conv}(A) &:= \bigcap \left\{ B \in \mathbb{R}^d \,\middle|\, A \subseteq B, B \text{ convex} \right\} \\ &= \left\{ \sum_{i=1}^{p} \alpha_i z_i \,\middle|\, p \geqslant 1, \alpha \in \mathbb{R}_+^p, (z_1, \ldots, z_p) \in A^p, \sum_{i=1}^{p} \alpha_i = 1 \right\} \end{aligned} \tag{3.2.1}$$

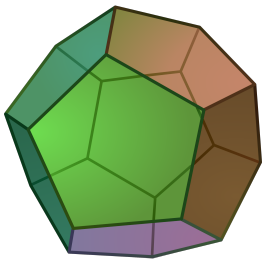

Figure 9: Polytope

3.2.3 Convex functions

3.2.4 Unconstrained optimization problems

4 Convex analysis and convex optimization

4.1 Constrained optimization problems

4.2 Optimization algorithms for unconstrained convex optimization

4.2.1 Gradient Descent

4.2.2 Stochastic Gradient Descent

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:

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

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

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

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

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

is obtained for

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

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

Proof. □

Corollary. For $\lambda > 0$,

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

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:

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

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:

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

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 10 (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 11. 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

7 Model-Based Machine Learning

8 Maximum Likelihood

9 Unsupervised Learning

10 MCMC Sampling

11 Neural Networks