Introduction to Machine Learning

Alessandro Rudi, Umut Simsekli
Notes by Antoine Groudiev

1st April 2024

Contents

Introduction

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 of Pierre Gaillard Alessandro Rudi, and Umut Şimşekli.

1 An overview of Machine Learning

1.1 What is ML?

Considering a problem, such as image classification: given an input image of a dog or a cat, the program is asked to determine whether the image is a dog or a cat. Conventional programming would hardcode the solution to this problem. But this process takes time and is not easily generalisable. Instead, an ML model is trained on a dataset to produce a program to solve the problem.

Many successfull applications of Machine Learning are:

- Face recognition
- Spam filtering
- Speech recognition
- Self-driving systems; pedestrian detection

1.2 Topics in Machine Learning

1.2.1 Supervised Learning

Example (Classification)**.** Features $x \in \mathbb{R}^d$, labels $y \in \{1, \dots, k\}$

Definition (Regression)**.** Features $x \in \mathbb{R}^d$, labels $y \in \mathbb{R}$. To tackle such problem, we look for a parametrized function $f_\theta(x_i) \simeq y_i$ for some f_θ in a function space

$$\mathcal{F} = \{f_\theta : \theta \in \Theta\}$$

Our goal is therefore to find the best function in $\mathcal{F}$ such that f "fits" the training data. For example, we can say that f "fits" the training data when

$$\frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2$$

is "small". Such a function is not interesting in general, like for classification.

Definition (Loss function)**.** Assums that the features are in $\mathcal{X}$ and the labels are in $\mathcal{Y}$. We introduce the more general *loss function* notion:

$$l : \mathcal{Y}^2 \to \mathbb{R}_+$$

For a regression task, we can use $l(\hat{y}, y) = (\hat{y} - y)^2$. For a classification task, $l(\hat{y}, y) = \mathbb{1}_{\hat{y}=y}$.

Therefore, for a regression problem, we might choose:

$$f^\star = \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i)$$

In the parametric case, when $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$, we might minimize with respect to θ:

$$\theta^\star = \operatorname*{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i)$$

1.2.2 Probabilistic approach

Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ be the feature space. Let D be a distribution on $\mathcal{Z}$; we make the assumption that the training data is iid from D:

$$(x_i, y_i) \sim D$$

and the same thing hold for the test data:

$$(\tilde{x_i}, \tilde{y_i}) \sim D$$

According to the Strong Law of Large Numbers, the test loss converges almost surely:

$$\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} l(f_\theta(\tilde{x_i}), \tilde{y_i}) = \mathbb{E}_{(x,y)\sim D}[l(f_\theta(x), y)] =: R(\theta) = R(f_\theta)$$

where $R(\theta)$ is the *population risk*.

Definition (Risk minimization)**.**

1.2.3 Unsupervised Learning

Example (Clustering)**.**

Example (Dimensionnality reduction)**.** We are given features $x \in \mathbb{R}^d$ and labels $y \in \{0, 1\}$ which form a "training" dataset $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. We assume that $d \gg 1$; our goal is to find $d' \ll d$ such that $(x_1, y_1, \ldots,)$

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:

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

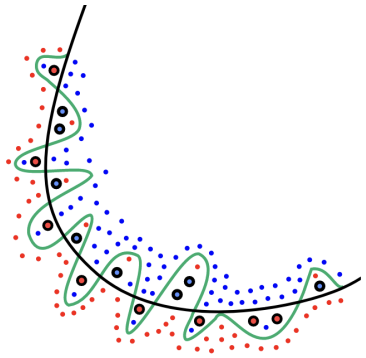

Figure 1: Example of overfitting

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

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

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).

[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 (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*:

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

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

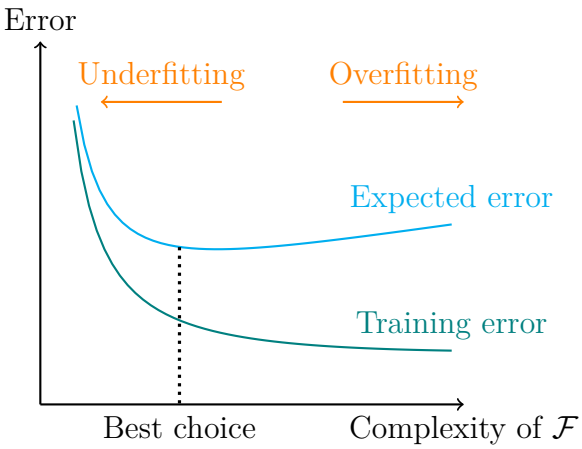

Figure 2: Overfitting and underfitting

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) is called the *Ordinary Least Squares (OLS) estimator*. Said otherwise, it is the vector $\hat{\theta} \in \mathbb{R}^d$ minimizing $\hat{\mathcal{R}}_n$:

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

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:

$$\hat{\theta} = (X^\top X)^{-1} X^\top Y \tag{2.3.1}$$

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

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 $\mathrm{Im}(X)$.

2.3.3 Numerical resolution

The closed form formula (2.3.1) 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:

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

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

[4] It is even positive definite.

[5] Using the Gauss-Jordan method

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.

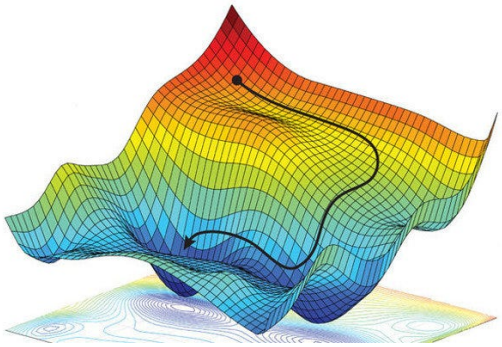

Figure 3: (Non-convex) Gradient descent

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 steps 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 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.

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

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

2.4 Statistical analysis

2.4.1 Stochastic assumptions and bias/variance decomposition

2.4.2 Statistical properties of OLS

2.5 Ridge regression

3 Logic regression and convex analysis

Recap of important notions and notations

We are given an input space X and an output space Y. We want to learn the relationship between input and output, modelised by a probability distribution $\rho \in \mathbb{P}(X \times Y)$. Thus, we try to find the best function $f_\star : X \to Y$, given a loss function $l : Y \times Y \to \mathbb{R}$. Therefore, $f_\star$ is often defined by:

$$f_\star = \underset{f:X\to Y}{\operatorname{argmin}} \mathbb{E}_{X,Y}[l(f(X), Y)]$$

where

$$\mathbb{E}_{X,Y}[g(X,Y)] = \int_{\mathbb{R}^2} g(x,y) \cdot \mathrm{d}\rho(x,y)$$

In practice, you only know some samples $D_N = [(x_1, y_1), \ldots, (x_N, y_N)]$ with $(x_i, y_i) \sim \rho$, making it impossible to choose such an $f_\star$. Therefore, we try to find a good model $\hat{f}_{D_N}$, such that

$$\lim_{N\to+\infty} \mathcal{E}(\hat{f}_{D_N}) - \mathcal{E}(f) = 0$$

Such a result will often be given by a *learning rate function* $c(N)$, with

$$\mathbb{E}_{D_N}[\mathcal{E}(\hat{f}_{D_N}) - \mathcal{E}(f)] \leqslant c(N) = o(1)$$

The function $\hat{f}_{D_N}$ can be chosen such that it minimizes the empirical error:

$$\hat{f}_{D_N} = \underset{f\in\mathcal{H}}{\operatorname{argmin}} \hat{\mathcal{E}}(f) = \underset{f\in\mathcal{H}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} l(f(x_i), y_i)$$

3.1

We consider the case where $X = \mathbb{R}^d$ and $Y = \mathbb{R}$. We define the loss l to be the least squares, $l(y, y') = (y - y')^2$, and we choose our functions to be of the form of $f_\star = \theta_\star^T X$. In this case, ERM is OLS:

$$\hat{\theta}_N = \underset{\theta\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} (\theta^T x_i - y_i)^2$$

We can also define $\hat{\theta}_{N,\lambda}$ to be:

$$\hat{\theta}_{N,\lambda} = \underset{\theta\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} (\theta^T x_i - y_i)^2 + \lambda \|\theta\|^2$$

This allows to regulate the "complexity" of the function to avoid overfitting. This is called Tikhonov regularization. In this case, we have

$$\mathbb{E}_{\hat{Y}}[\hat{\theta}_{\mathcal{N}} - \mathcal{E}(\theta_\star)] = \frac{\sigma^2 d}{N}$$

and therefore the optimal function is

$$\hat{f}_{N,\lambda} = \operatorname*{argmin}_{f \in \mathcal{H}} \hat{\mathcal{E}}(f) + \lambda R(f)$$

We define $X \in \mathbb{R}^{N \times d} := (x_1^T, \ldots, x_N^T)$, and $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_n)$. We this notation, we have

$$\hat{\theta}_{N,\lambda} = \frac{1}{N}||X\theta - \hat{Y}||^2 + \lambda||\theta||^2$$

Thus, we have

$$\nabla \mathcal{L}(\theta) := \frac{2}{N} X^T X\theta - 2\frac{X^T\hat{Y}}{N} + 2\lambda\theta = 0$$
$$(\frac{X^T X}{N} + \lambda)\theta = X^T\hat{Y}$$

therefore,

$$\hat{\theta}_{N,\lambda} = \left(\frac{X^T X}{N} + \lambda I\right)^{-1} \frac{X^T\hat{Y}}{N} = \left(X^T X + \lambda N I\right)^{-1} X^T\hat{Y}$$

We introduce the singular value decomposition of X:

$$X = U\Sigma V^T$$

where $U^T U = UU^T = I_N$, $V^T V = VV^T = I_d$, and Σ is diagonal with $\forall i$, $\Sigma_{i,i} \geqslant 0$. In this case,

$$X^T X + \lambda N I_d = V\Sigma U^T U\Sigma V^T + \lambda N I_d$$
$$= V(\underbrace{\Sigma^2 + \lambda N I}_{\text{invertible}})V^T$$

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:

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

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 $x_1, \dots, x_n$, 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 2 (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 3. 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