Basic models in machine learning

Pierre Gaillard

Sept. 26 2022

1 Introduction to supervised learning

Learning goals: Understand the general concepts of machine learning: training/validation/testing data, algorithm, loss function, risk, empirical risk.

Let's start with an example of a practical problem. In order to better optimize its production, a producer is interested in modeling electricity consumption in France as a function of temperature (cf. Figure 1).

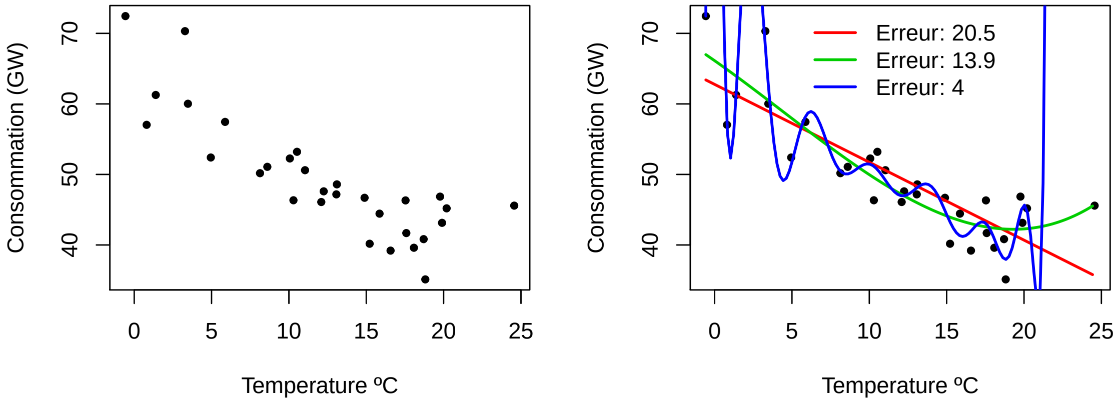


Figure 1: French power consumption (GW) as a function of temperature (ºC). To the right are plotted error minimizing functions for polynomial spaces of degrees 1 (red), 3 (green) and 30 (blue).

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

$$\widehat{f}_n \in \mathop{\arg\min}_{f\in\mathcal{F}} \widehat{\mathcal{R}}(f) := \mathop{\arg\min}_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 . \tag{1}$$

Care must be taken when selecting the function space to avoid over-fitting (see Figures 1). Although the empirical mean square error decreases when the $\mathcal{F}$ space becomes larger (larger polynomial degrees), the $\widehat{f}_n$ estimator loses its predictive power. The question is: will $\widehat{f}_n$ perform well on new data?

Supervised learning: general setup and notation

Goal. In supervised machine learning, the goal is given some observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ of inputs/outputs and given a new input $x \in \mathcal{X}$ to predict well the next output $y \in \mathcal{Y}$. The training data set will be denoted $D_n := \{(x_i, y_i), i = 1, \dots, n\}$. 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 D_n data. A learning rule $\mathcal{A}$ is a function that associates to training data D_n a prediction function $\widehat{f}_n$ (the hat on f indicates that it is an estimator):

$$\begin{array}{rcl} \mathcal{A} : \cup_{n\in\mathbb{N}} \big(\mathcal{X} \times \mathcal{Y}\big)^n & \to & \mathcal{Y}^{\mathcal{X}} \\ D_n & \mapsto & \widehat{f}_n \end{array} .$$

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

Risk and empirical risk. The objective is to find an estimator $\widehat{f}_n$ that predicts well new data by minimizing the risk:

$$\mathcal{R}(\widehat{f}_n) := \mathbb{E}\Big[\big(y - \widehat{f}_n(x)\big)^2 \,\Big|\, D_n\Big] \qquad \text{where} \qquad (x, y) \sim \nu\,. \qquad \text{(Risk)}$$

However, the statistician cannot compute the expectation (and thus the risk) because he does not know ν. A common method in supervised machine learning is therefore to replace the risk with the empirical risk.

$$\widehat{\mathcal{R}}(f) = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - f(x_i)\big)^2\,. \qquad \text{(Empirical risk)}$$

However, one must be careful about over-fitting (case where $\widehat{\mathcal{R}}(f)$ is much lower than $\mathcal{R}(f)$, see Figure 2). In this class, we will study the performance of the least square estimator in the case of the linear model.

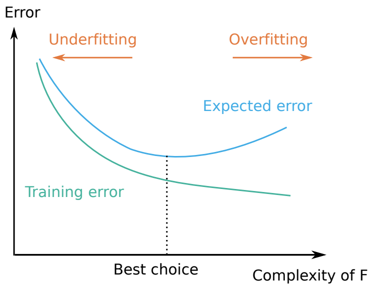


Figure 2: Over-fitting and under-fitting according to the complexity of $\mathcal{F}$. In blue the risk $\mathcal{R}(f)$ which we want to minimize, in green the empirical risk $\widehat{\mathcal{R}}(f)$ that we observe on the training data.

2 Linear least-squares regression

Learning goals: Understand the general concepts of linear regression: definition, how to derive the closed-form solution and gradient descent updates, understand matrix notations, know how linear regression can learn non-linear functions using features, have a highlevel idea about bias variance trade-off and its impact on overfitting, know how to regularize.

We refer the interested reader to Bach, 2022 for more details and exercises on this section.

In this section, we study the simple but still widely used problem of linear least-squares regression. The linear regression problem can be traced back to Legendre (1805) and Gauss (1809). The word "regression" is said to have been introduced by Galton in the 19th century. By modeling the size of individuals according to that of their fathers, Galton observed a return (regression) towards average height. Larger-than-average fathers tend to have smaller children and vice versa for smaller fathers.

Here, we consider real outputs ($\mathcal{Y} = \mathbb{R}$) and square loss $\ell(y, z) = (y - z)^2$. Given a parametrized family of prediction function $\mathcal{F} := \{f_\theta : \mathcal{X} \to \mathcal{Y}, \theta \in \Theta\}$, we minimize the empirical risk

$$\widehat{\mathcal{R}}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \big(y_i - f_\theta(x_i)\big)^2 .$$

In linear least-square regression, the functions $\theta \mapsto f_\theta(x)$ are assumed to be linear in θ.

⚠ Being linear in θ or x is different. Nothing forces $f_\theta(x)$ to be linear in x. Typically,

$$f_\theta(x) = \langle \theta, \varphi(x) \rangle$$

for some feature map $\varphi(x) \in \mathbb{R}^d$. For example, affine functions may be obtained with $\varphi(x) = (x^\top, 1)^\top$ and polynomials with $\varphi(x) = \big(1, x_1, x_2, x_1^2, x_2^2, x_1 x_2, \dots\big)^\top$. In Figure 1, we have in this way minimized the empirical risk on polynomial spaces of degree 1 (linear model), 3 and 30. We can see that we must be careful not to consider spaces that are too large, at the risk that the model is badly posed (design matrix non injective as seen thereafter). Conversely, for the statistical analysis that we will see next to be verified, one must be in the true model $y = \langle \varphi(x), \theta^* \rangle +$ centered noise. We must therefore make sure that $\varphi(x)$ contains enough descriptors so that the dependency between y and $\varphi(x)$ is indeed linear. Otherwise we pay an additional bias term.

Why should we study linear regression?

- It captures many concepts of learning theory: bias-variance trade-off, need of regularization,...
- It is simple: the analysis can be done in basics maths (linear algebra).
- Using non-linear features, it can be extended to non-linear predictions ↦ kernel methods.

Matrix notation The empirical risk can be rewritten in matrix notation. Let $y = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$ be the vector of outputs and $\varphi \in \mathbb{R}^{n \times d}$ the matrix of inputs (also called design matrix or data matrix), which rows are $\varphi(x_i)^\top$:

$$\varphi = \Big(\varphi(x_1), \varphi(x_2), \dots, \varphi(x_n)\Big)^\top \in \mathbb{R}^{n \times d} .$$

The empirical risk is then

$$\widehat{\mathcal{R}}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - \langle\theta, \varphi(x_i)\rangle\big)^2 = \frac{1}{n}\big\|y - \varphi\theta\big\|_2^2 . \tag{2}$$

⚠ The matrix notation is very useful to simplify calculation.

2.1 Ordinary Least Squares Estimator (OLS)

In the following, we assume that the design matrix φ is injective (i.e., the rank of φ is d). In particular, $d \leqslant n$.

Definition 2.1. *If φ is injective, the minimizer of the empirical risk*

$$\widehat{\theta} = \arg\min_{\theta\in\Theta} \frac{1}{n}\big\|y - \varphi\theta\big\|_2^2 ,$$

is called the Ordinary Least Squares (OLS) estimator.

Proposition 2.1 (Closed form solution)**.** *If φ is injective, the OLS exists and is unique. It is given by*

$$\widehat{\theta} = \big(\varphi^\top\varphi\big)^{-1}\varphi^\top y\,.$$

Proof. Since $\widehat{\mathcal{R}}$ is coercive (goes to infinity in infinity) and continuous, it admits at least a minimizer. Furthermore, we have

$$\widehat{\mathcal{R}}(\theta) = \frac{1}{n}\big\|y - \varphi\theta\big\|_2^2 = \frac{1}{n}\big(\theta^\top(\varphi^\top\varphi)\theta - 2\theta^\top\varphi^\top y + \|y\|^2\big).$$

Since $\widehat{\mathcal{R}}$ is differentiable any minimizer should cancel the gradient:

$$\nabla\widehat{\mathcal{R}}(\widehat{\theta}) = \frac{1}{n}\big(\widehat{\theta}^\top(\varphi^\top\varphi) + (\varphi^\top\varphi)\widehat{\theta} - 2\varphi^\top y\big) = \frac{2}{n}\big((\varphi^\top\varphi)\widehat{\theta} - \varphi^\top y\big).$$

where the last equality is because $\varphi^\top\varphi \in \mathbb{R}^{d\times d}$ is symmetric. Since φ is injective, $\varphi^\top\varphi$ is invertible (Exercise: show this implication). Therefore, a solution of $\nabla\widehat{\mathcal{R}}(\widehat{\theta}) = 0$ satisfies

$$\widehat{\theta} = (\varphi^\top\varphi)^{-1}\varphi^\top y.$$

However, it remains to check that this is indeed a minimum and therefore that the Hessian is definite positive, which is the case because: $\nabla^2\widehat{\mathcal{R}}(\widehat{\theta}) = \frac{2}{n}(\varphi^\top\varphi)$. □

Geometric interpretation The linear model seeks to model the output vector $y \in \mathbb{R}^n$ by a linear combination of the form $\varphi\theta \in \mathbb{R}^n$. The image of φ is the solution space, denoted $\mathrm{Im}(\varphi) = \{z \in \mathbb{R}^n : \exists\theta \in \mathbb{R}^d \text{ s.t. } z = \varphi\theta\} \subseteq \mathbb{R}^n$. This is the vector subspace of $\mathbb{R}^n$ generated by the $d < n$ columns of the design matrix. As $\mathrm{rg}(\varphi) = d$, it is of dimension d.

By minimizing $\|y - \varphi\theta\|$ (cf. Definition 2.1), we thus look for the element of $\mathrm{Im}(\varphi)$ closest to y. This is the orthogonal projection of y on $\mathrm{Im}(\varphi)$, denoted $\widehat{y}$. By definition of the OLS and by the Proposition 2.1, we have:

$$\widehat{y} \quad \overset{\text{Def 2.1}}{=} \quad \varphi\widehat{\theta} \quad \overset{\text{Prop. 2.1}}{=} \quad \varphi(\varphi^\top\varphi)^{-1}\varphi^\top y.$$

In particular, $P_\varphi := \varphi(\varphi^\top\varphi)^{-1}\varphi^\top \in \mathbb{R}^{n\times n}$ is the projection matrix on $\mathrm{Im}(\varphi)$.

Numerical resolution

The closed form formula $\widehat{\theta} = \left(\varphi^\top \varphi\right)^{-1} \varphi^\top y$ from the OLS is useful in analyzing it. However, calculating it naively can be prohibitively expensive. Especially when d is large, one prefers to avoid inverting the design matrix $\varphi^\top \varphi$ which costs $\mathcal{O}(d^3)$ by the Gauss-Jordan method 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. Recall that $\widehat{\theta}$ is the solution to the equation:

$$(\varphi^\top \varphi)\widehat{\theta} = \varphi^\top y \,,$$

We write $\varphi \in \mathbb{R}^{n\times d}$ of the form $\varphi = QR$, where $Q \in \mathbb{R}^{n\times d}$ is an orthogonal matrix (i.e., $Q^\top Q = I_d$) 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:

$$\begin{aligned} R^\top (Q^\top Q) R\widehat{\theta} = R^\top Q^\top y \quad &\Leftrightarrow \quad R^\top R\widehat{\theta} = R^\top Q^\top y \\ &\Leftarrow \quad R\widehat{\theta} = Q^\top y \,. \end{aligned}$$

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

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 approaching the minimum through gradient steps. For example, we initialize $\widehat{\theta}_0$ =0, then update:

$$\begin{aligned} \widehat{\theta}_{i+1} &= \widehat{\theta}_i - \eta \nabla \widehat{\mathcal{R}}(\widehat{\theta}_i) \\ &= \widehat{\theta}_i - \frac{2\eta}{n}\left((\varphi^\top \varphi)\widehat{\theta}_i - y^\top \varphi\right) , \end{aligned}$$

where $\eta > 0$ is a learning parameter. We see that if the algorithm converges, then it converges to a point canceling the gradient, thus to the OLS solution. To have convergence, the η parameter must be well calibrated, but this is beyond the scope of these notes.

If the data set is much too big, $n \gg 1$. It can also be prohibitively expensive to load all the data to make the $\nabla \widehat{\mathcal{R}}(\widehat{\theta}_i)$ calculation. The common solution is then to do Stochastic Gradient Descent, where gradient steps are made only on estimates of $\nabla \widehat{\mathcal{R}}(\widehat{\theta}_i)$, calculated on a random subset of the data.

2.2 Statistical analysis

In this section, we will provide theoretical guarantees for the OLS. To do so, we will need some probabilistic assumptions.

> ⚠ This section is here to provide theoretical insights on bias-variance trade-off in machine learning and how to compute it on a simple model like linear regression and how regularization helps. Detailed calculations will not be asked at the final exam.

2.2.1 Stochastic assumptions

Any kind of guarantees requires assumption about how the data is generated. In this section, we consider a stochastic framework that will allow us to analyze the performance of OLS.

Assumption 1 (Linear model)**.** *We assume that there exists a vector $\theta^* \in \mathbb{R}^d$ such that for all $1 \leqslant i \leqslant n$*

$$y_i = \langle \varphi(x_i), \theta^* \rangle + \varepsilon_i \,, \tag{3}$$

where $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)^\top \in \mathbb{R}^n$ is a vector of errors (or noise). The ε_i are assumed to be centered independent variables $\mathbb{E}[\varepsilon_i] = 0$ and with variance $\mathbb{E}[\varepsilon_i] = \sigma^2$.

Recall that x_i, y_i and ε_i (from now on) are random variables. The noise ε_i comes from the fact that in practice the observation y_i never completely fits the linear forecast. This is due to noise or unobserved explanatory variables. The Equation (3) can be rewritten in matrix form:

$$y = \varphi\theta^* + \varepsilon$$

where $y = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$, $\varphi = (\varphi(x_1), \dots, \varphi(x_n))^\top \in \mathbb{R}^{n\times d}$ and $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)^\top \in \mathbb{R}^n$.

From here, there are two settings of analysis for least squares:

- *Fixed design.* In this setting, the design matrix φ is not random but deterministic and the features $\varphi(x_1), \dots, \varphi(x_n)$ are fixed. The expectations are thus only with respect to ε_i and y_i and the goal is to minimize the risk

  $$\mathcal{R}_\varphi(\theta) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n \left(y_i - \varphi(x_i)^\top\theta\right)^2\right] = \mathbb{E}\left[\frac{1}{n}\|y - \varphi\theta\|_2^2\right] \,, \tag{4}$$

  for new random observations y_i (different from the ones observed in the dataset) but on the same inputs.

- *Random design.* Here, both the inputs and the outputs are random. This is the most standard setting of supervised machine learning. The goal is to minimize the risk (sometimes called the generalization error) defined in Equation (Risk).

In this class, we consider the fixed design setting because it eases the notation and the calculation (we only need simple linear algebra).

2.2.2 Bias/variance decomposition

Before analyzing the statistical properties of OLS, we state a general result under the linear model which illustrate the trade-off between estimation and approximation (or bias and variance).

Proposition 2.2 (Risk decomposition)**.** *Under the linear model (Assumption 1) with fixed design, for any $\theta \in \mathbb{R}^d$ it holds*

$$\mathbb{E}\big[\mathcal{R}_\varphi(\theta) - \mathcal{R}_\varphi(\theta^*)\big] = \|\theta - \theta^*\|_\Sigma^2$$

where $\Sigma = \frac{1}{n}\varphi^\top\varphi \in \mathbb{R}^{d\times d}$ and $\|\theta\|_\Sigma^2 = \theta^\top\Sigma\theta$. If θ is a random variable (because it depends on a random data set) but independent from the test data then

$$\mathbb{E}\big[\mathcal{R}_\varphi(\theta)\big] - \mathcal{R}_\varphi(\theta^*) = \underbrace{\big\|\mathbb{E}[\theta] - \theta^*\big\|_\Sigma^2}_{\textit{Bias}} + \underbrace{\mathbb{E}\Big[\big\|\theta - \mathbb{E}[\theta]\big\|_\Sigma^2\Big]}_{\textit{Variance}} \,.$$

Proof. Now, let $\theta \in \mathbb{R}^d$. Then, taking the expectation over y,

$$\begin{aligned}
\mathcal{R}_\varphi(\theta) &= \mathbb{E}\Big[\frac{1}{n}\big\|y-\varphi\theta\big\|_2^2\Big] \\
&= \mathbb{E}\Big[\frac{1}{n}\big\|y-\varphi\theta^* + \varphi(\theta^*-\theta)\big\|_2^2\Big] \\
&= \mathbb{E}\Big[\frac{1}{n}\big\|y-\varphi\theta^*\big\|^2\Big] + \frac{2}{n}\mathbb{E}\Big[\cancel{\big(y-\varphi\theta^*\big)^\top}\Big]\varphi(\theta^*-\theta) + \frac{1}{n}\big\|\varphi(\theta^*-\theta)\big\|_2^2 \qquad (5) \\
&= \mathcal{R}_\varphi(\theta^*) + \big\|\theta-\theta^*\big\|_\Sigma^2 .
\end{aligned}$$

If θ is random but independent from y, we have the following bias-variance decomposition

$$\begin{aligned}
\mathbb{E}\big[\mathcal{R}_\varphi(\theta)\big] - \mathcal{R}_\varphi(\theta^*) &= \mathbb{E}\Big[\big\|\theta-\mathbb{E}[\theta]+\mathbb{E}[\theta]-\theta^*\big\|_\Sigma^2\Big] \\
&= \mathbb{E}\Big[\big\|\theta-\mathbb{E}[\theta]\big\|_\Sigma^2\Big] + \mathbb{E}\Big[(\theta-\mathbb{E}[\theta])^\top\Sigma(\mathbb{E}[\theta]-\theta^*)\Big] + \mathbb{E}\Big[\big\|\mathbb{E}[\theta]-\theta^*\big\|_\Sigma^2\Big] \\
&= \mathbb{E}\Big[\big\|\theta-\mathbb{E}[\theta]\big\|_\Sigma^2\Big] + 2\mathbb{E}\big[\cancel{(\theta-\mathbb{E}[\theta])}\big]^\top\Sigma(\mathbb{E}[\theta]-\theta^*) + \big\|\mathbb{E}[\theta]-\theta^*\big\|_\Sigma^2 \\
&= \mathbb{E}\Big[\big\|\theta-\mathbb{E}[\theta]\big\|_\Sigma^2\Big] + \mathbb{E}\Big[\big\|\mathbb{E}[\theta]-\theta^*\big\|_\Sigma^2\Big] .
\end{aligned}$$

□

It is worth to note that the optimal risk satisfies

$$\mathcal{R}_\varphi(\theta^*) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n \big(y_i - \varphi(x_i)^\top\theta^*\big)^2\right] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n \varepsilon_i^2\right] = \frac{1}{n}\sum_{i=1}^n \mathbb{E}\big[\varepsilon_i^2\big] = \sigma^2 .$$

⚠ Note also that, here, we assumed that θ is independent from the test data y, allowing us to factor it out of the expectation in the red term of Equation (5). However, when θ and y are correlated (e.g., because θ is an estimator that used some test data or because y is obtained from train data) this term becomes non-zero (see Exercise 2.1).

2.2.3 Statistical properties of OLS

We now show some guarantees for the OLS estimator.

Proposition 2.3. *Under the linear model (i.e., Assumption 1) with fixed design, the OLS estimator $\widehat{\theta}$ defined in Definition 2.1 satisfies:*

- *it is unbiased* $\mathbb{E}\big[\,\widehat{\theta}\,\big] = \theta^*$.
- *its variance is* $\mathrm{Var}(\widehat{\theta}) = \frac{\sigma^2}{n}\Sigma^{-1}$.

We can even show that the OLS satisfies the Gauss-Markov property. It is optimal among unbiased estimators of θ, in the sense that it has a minimal variance-covariance matrix.

Proof. Using $\mathbb{E}[\varepsilon_i] = 0$ and $y = \varphi\theta^* + \varepsilon$, we have

$$\mathbb{E}[\widehat{\theta}] = \mathbb{E}\big[(\varphi^\top\varphi)^{-1}\varphi^\top y\big] = \mathbb{E}\big[(\varphi^\top\varphi)^{-1}\varphi^\top\varphi\theta^* + \cancel{(\varphi^\top\varphi)^{-1}\varphi^\top}\varepsilon\big] = \theta^* .$$

Furthermore, using $\mathrm{Var}(y) = \mathrm{Var}(\varepsilon) = \sigma^2 I_n$, we have

$$\mathrm{Var}(\widehat{\theta}) = \mathrm{Var}\big((\varphi^\top\varphi)^{-1}\varphi^\top y\big) = (\varphi^\top\varphi)^{-1}\varphi^\top\mathrm{Var}(y)\varphi(\varphi^\top\varphi)^{-1} = \sigma^2(\varphi^\top\varphi)^{-1} = \frac{\sigma^2}{n}\Sigma^{-1} .$$

□

Corollary 2.4 (Excess risk of OLS)**.** *Under the linear model with fixed design, the excess risk of the OLS satisfy*

$$\mathbb{E}\big[\mathcal{R}_\varphi(\widehat{\theta})\big] - \mathcal{R}_\varphi(\theta^*) = \frac{\sigma^2 d}{n}\,.$$

Proof. Using the bias-variance decomposition and the fact that θ^* is unbiased (i.e., $\mathbb{E}[\widehat{\theta}] = \theta^*$), we have

$$\begin{aligned}
\mathbb{E}\big[\mathcal{R}_\varphi(\widehat{\theta})\big] - \mathcal{R}_\varphi(\theta^*) &= \cancel{\|\mathbb{E}[\widehat{\theta}] - \theta^*\|_\Sigma^2} + \mathbb{E}\Big[\|\widehat{\theta} - \mathbb{E}[\widehat{\theta}]\|_\Sigma^2\Big] = \mathbb{E}\Big[\|\widehat{\theta} - \theta^*\|_\Sigma^2\Big] \\
&= \mathbb{E}\Big[(\widehat{\theta} - \theta^*)^\top \Sigma (\widehat{\theta} - \theta^*)\Big] \\
&= \frac{1}{n}\mathbb{E}\big[(\widehat{\theta} - \theta^*)^\top \varphi^\top \varphi (\widehat{\theta} - \theta^*)\big] \\
&= \frac{1}{n}\mathbb{E}\big[\operatorname{Tr}\big((\widehat{\theta} - \theta^*)^\top \varphi^\top \varphi (\widehat{\theta} - \theta^*)\big)\big] \\
&= \frac{1}{n}\mathbb{E}\big[\operatorname{Tr}\big(\varphi(\widehat{\theta} - \theta^*)(\widehat{\theta} - \theta^*)^\top \varphi^\top\big)\big] && \leftarrow \text{because } \operatorname{Tr}(AB) = \operatorname{Tr}(BA) \\
&= \frac{1}{n}\operatorname{Tr}\Big(\varphi\mathbb{E}\big[(\widehat{\theta} - \theta^*)(\widehat{\theta} - \theta^*)^\top\big]\varphi^\top\Big) && \leftarrow \text{because } \mathbb{E} \text{ and Tr are linear operators} \\
&= \frac{1}{n}\operatorname{Tr}\big(\varphi \operatorname{Var}(\widehat{\theta})\varphi^\top\big) \\
&= \frac{\sigma^2}{n}\operatorname{Tr}\big(\varphi(\varphi^\top\varphi)^{-1}\varphi^\top\big) \\
&= \frac{\sigma^2}{n}\operatorname{Tr}\big((\varphi^\top\varphi)^{-1}\varphi^\top\varphi\big) && \leftarrow \text{because } \operatorname{Tr}(AB) = \operatorname{Tr}(BA) \\
&= \frac{\sigma^2}{n}\operatorname{Tr}(I_d) = \frac{\sigma^2 d}{n}\,.
\end{aligned}$$

□

Exercise 2.1. *Show that the empirical risk of the OLS estimator satisfies the equality*

$$\mathbb{E}\big[\widehat{\mathcal{R}}_\varphi(\widehat{\theta})\big] = \frac{n-d}{n}\sigma^2.$$

In particular, an unbiased estimator of the noise variance σ^2 is

$$\widehat{\sigma}^2 = \frac{\|y - \varphi\widehat{\theta}\|^2}{n-d}\,.$$

Gaussian noise model A very considered special case is Gaussian noise $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. This choice comes not only from the fact that it allows to compute many additional statistical properties on $\widehat{\theta}$ and to perform tests (confidence intervals, 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 phenomena not explained by the linear combination of the explanatory variables.

Proposition 2.5. *In the linear model with Gaussian noise, the maximum likelihood estimators of θ and σ satisfy respectively:*

$$\widehat{\theta}_{MV} = \big(\varphi^\top\varphi\big)^{-1}\varphi y \qquad \textit{and} \qquad \widehat{\sigma}^2_{MV} = \frac{\|y - \varphi\widehat{\theta}\|^2}{n}\,.$$

We will prove more formally this proposition in the maximum likelihood section (Section 4. We therefore find the least-squares estimator obtained by minimizing the empirical risk. The variance estimator is biased.

2.3 Ridge regression

If φ is not injective (i.e., $\mathrm{rg}(\varphi) \neq d$), the matrix $\Sigma := \frac{1}{n}\varphi^\top \varphi$ is no longer invertible and the OLS optimization problem admits several solutions. The problem is said to be poorly posed or unidentifiable.

The Proposition 2.3 reminds us that the variance of $\widehat{\theta}$ depends on the conditioning of the matrix $\Sigma^{-1} = n(\varphi^\top \varphi)^{-1}$. The more the columns of the latter are likely to be dependent, the less stable $\widehat{\theta}$ will be. Several solutions allow to deal with the case where $\mathrm{rg}(\varphi) < d$:

- *explicit complexity control* by reducing the solution space $\mathrm{Im}(\varphi)$. This can be done by removing columns from the φ 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 $y \in \mathrm{Im}(\varphi)$ has a unique antecedent $\theta \in V$ with $y = \varphi\theta$. For example, we could choose $V = \mathrm{Ker}(\varphi)^\perp$.
- *implicit complexity control* by regularizing the empirical risk minimization problem. The most common is to regularize by adding $\|\theta\|_2^2$ (Ridge regression, which we see below) or $\|\theta\|_1$ (Lasso regression).

Definition 2.2. *For a regularization parameter λ, the Ridge regression estimator is defined as*

$$\widehat{\theta}_\lambda \in \operatorname*{arg\,min}_{\theta\mathbb{R}^d} \left\{ \frac{1}{n}\|y - \varphi\theta\|_2^2 + \lambda\|\theta\|_2^2 \right\} .$$

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

Proposition 2.6. *The Ridge regression estimator is unique (even if φ is not injective) and satisfies*

$$\widehat{\theta}_\lambda = \left(\varphi^\top \varphi + n\lambda I_n\right)^{-1} \varphi^\top y \,.$$

The proof is similar to the one of OLS and left as exercise. We can see that there is no longer the problem of inverting $\varphi^\top\varphi$ since the Ridge regression amounts to replacing $\left(\varphi^\top\varphi\right)^{-1}$ by $\left(\varphi^\top\varphi + n\lambda I_n\right)^{-1}$ in the OLS solution.

Proposition 2.7 (Risk of Ridge regression)**.** *Under the linear model (Assumption 1), the Ridge regression estimator satisfies*

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

where λ_j is the j-th eigenvalue of $\Sigma = \frac{1}{n}\varphi^\top\varphi$. In particular, the choice $\lambda^* = \frac{\sigma\sqrt{\mathrm{Tr}(\Sigma)}}{\|\theta^*\|_2\sqrt{n}}$ yields

$$\mathbb{E}\big[\mathcal{R}_\varphi(\widehat{\theta}_{\lambda^*})\big] - \mathcal{R}_\varphi(\theta^*) \leqslant \frac{\sigma\sqrt{2\,\mathrm{Tr}(\Sigma)}\|\theta^*\|_2}{\sqrt{n}} \,.$$

The proof, which follows from the bias-variance decomposition (Proposition 2.2) is left as exercise.

Note that as $\lambda \to 0$, its 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 contrast to the OLS. But on the positive side, its variance does not involve the inverse of Σ but of $\Sigma + \lambda I_d$ which is better conditioned. It has therefore a lower variance. The parameter λ controls this trade-off.

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

- First, the one of OLS decreases in $O(1/n)$ while this one converges slower in $O(1/\sqrt{n})$ which could seem worse. Yet Ridge has a milder dependence on the noise σ instead of σ^2.
- Furthermore, since $\mathrm{Tr}(\Sigma) \leqslant \max_{1\leqslant i\leqslant n} \|\varphi(x_i)\|^2$, 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 essential in practice. It can for example be done analytically as in the proposition (but some quantities are unknown σ^2, $\|\theta^*\|$,...). In practice one resorts to train/validation set or *cross-validation (generalized).*

3 Logistic regression

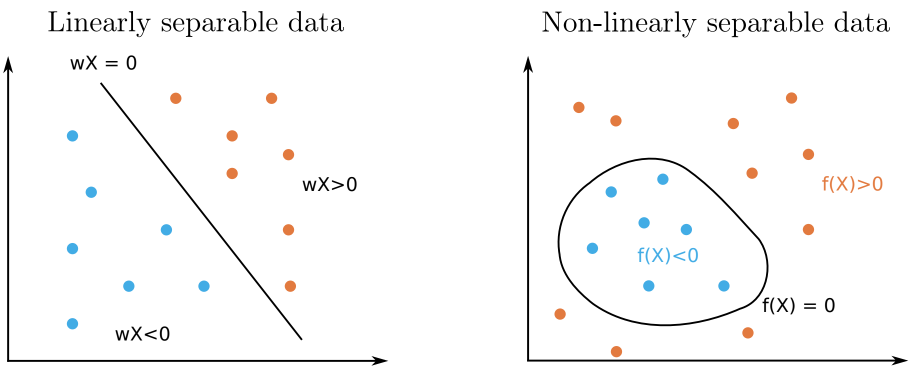
Learning objectives: understand the main concepts of logistic regression, what loss function it minimizes, how it can be seen as a convexification of zero-one loss, how to perform gradient descent, how to classify data when it is not linearly separable.

We will consider the binary classification problem in which one wants to predict outputs in $\{0,1\}$ from inputs in $\mathbb{R}^d$. We consider a training set $D_n := \big\{(x_i, y_i)\big\}_{1 \leqslant i \leqslant n}$. The data points (x_i, y_i) are i.i.d. random variables and follow a distribution $\mathcal{P}$ in $\mathcal{X} \times \mathcal{Y}$. Here, $\mathcal{Y} = \{0,1\}$ but it is also common to consider $\{-1,1\}$.

Goal We would like to use a similar algorithm to linear regression. However, since the outputs y_i are binary and belong to $\{0,1\}$ we cannot predict them by linear transformation of the inputs x_i (which belong to $\mathbb{R}^d$). We will thus classify the data thanks to classification rules $f : \mathbb{R}^d \mapsto \mathbb{R}$ such that:

$$f(x_i) \begin{cases} \geqslant 0 & \Rightarrow \quad y_i = +1 \\ < 0 & \Rightarrow \quad y_i = 0 \end{cases},$$

to separate the data into two groups. In particular, we will consider linear functions f of the form $f_\theta : x \mapsto \langle x, \theta \rangle$. This assumes that the data are well-explained by a linear separation (see figure below).


Linearly separable data

Non-linearly separable data

Of course, if the data does not seem to be linearly separable, we can use similar tricks that we mentioned for linear regression (polynomial regression, kernel regression, splines,...). We search a feature map $x \mapsto \varphi(x)$ into a higher dimensional space in which the data are linearly separable. This will be the topic of the class on Kernel methods.

Loss function To minimize the empirical risk, it remains to choose a loss function to assess the performance of a prediction. A natural loss is the *binary loss*: 1 if there is a mistake ($f(x_i) \neq y_i$) and 0 otherwise. The empirical risk is then:

$$\widehat{\mathcal{R}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{y_i \neq \mathbb{1}_{\langle x_i, \theta \rangle \geqslant 0}} .$$

This loss function is however not convex in θ. The minimization problem $\min_\theta \widehat{\mathcal{R}}(\theta)$ 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 θ. This is the case of the *Hinge loss* and of the logistic loss $\ell : \{0,1\} \times \mathbb{R} \to \mathbb{R}_+$. The latter assigns to a linear prediction $z = x^\top \theta$ and an observation $y \in \{0,1\}$ the loss

$$\ell(y,z) := y \log\left(1+e^{-z}\right) + (1-y)\log\left(1+e^{z}\right). \tag{6}$$

The binary loss, Hinge loss and logistic loss are plotted in Figure 3. Note that if the output space is $\mathcal{Y} = \{-1,1\}$, the logistic loss is defined differently: $\ell(y,z) := \log(1+e^{-zy})$.

Definition 3.1 (Logistic regression estimator)**.** *The logistic regression estimator is the solution of the following minimization problem:*

$$\widehat{\theta}_{(logit)} = \operatorname*{arg\,min}_{\theta \in \mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell\big(y_i, \langle x_i, \theta\rangle\big),$$

where ℓ is the logistic loss defined in Equation (6).

An 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 of (x_i, y_i). We will see more on this in the lecture on Maximum Likelihood.

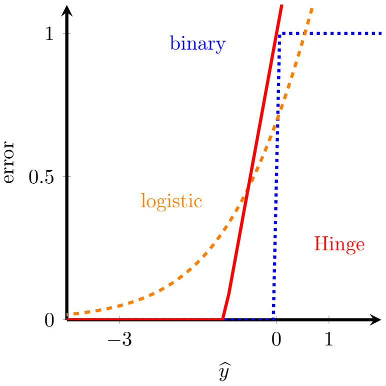


Figure 3: Binary, logistic and Hinge loss incured for a prediction $z := \langle x, \theta\rangle$ when the true observation is $y = 0$.

Computation of $\widehat{\theta}_{(\text{logit})}$ Similarly to OLS, we may try to analytically solve the minimization problem by canceling the gradient of the empirical risk. Since

$$\frac{\partial \ell(y,z)}{\partial z} = \sigma(z) - y, \qquad \text{where} \quad \sigma : z \mapsto \frac{1}{1+e^{-z}}$$

is the logistic function, we have:

$$\nabla \widehat{\mathcal{R}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} x_i\big(\sigma(\langle x_i, \theta\rangle) - y_i\big) = \frac{1}{n} x\big(y - \sigma(x\theta)\big)$$

where $x := (x_1, \dots, x_n)^\top$, $y := (y_1, \dots, y_n)$, and $\sigma\big(\langle x, \theta\rangle\big)_i := \sigma(\langle x_i, \theta\rangle)$ for $1 \leqslant i \leqslant n$. Bad news: the equation $\nabla \widehat{\mathcal{R}}(\theta) = 0$ has no closed-form solution. It needs to be solved through iterative algorithm (gradient descent, Newton's method,...). Fortunately, this is possible because the logistic loss is convex in its first argument. Indeed,

$$\frac{\partial^2 \ell(y,z)}{\partial z} = \sigma(z)\sigma(-z) > 0\,.$$

The loss is strictly convex, the solution is thus unique.

Regularization Similarly to linear regression, logistic regression may over-fit the data (especially when $d > n$). One needs then to add a regularization such as $\lambda \|\theta\|_2^2$ to the logistic loss.

4 Probabilistic models: maximum likelihood estimation

Learning objectives: understand the highlevel idea and definition of maximum likelihood, know how to compute it for simple models like Gaussians or Binomials, understand the connexion with logistic and linear regression.

In probabilistic modeling, we are given a set of observations $D_n = (y_1, \dots, y_n)$ in $\mathcal{Y}$ that we assume to be generated from some unknown i.i.d. distribution. The objective is to find a probabilistic model that explains well the data. For instance by estimating the density of the underlying distribution. If possible, we would like the model to predict well new data and to be able to incorporate prior knowledge and assumptions.

Let μ denote some reference measure on the output set $\mathcal{Y}$. Typically, μ is the counting measure if $\mathcal{Y} \subset \mathbb{N}$ or the Lebesgue measure if $\mathcal{Y} \subset \mathbb{R}^p$.

Definition 4.1 (Parametric model)**.** *Let $d \geqslant 1$ and $\Theta \subseteq \mathbb{R}^d$ be a set of parameters. A parametric model $\mathcal{P}$ is a set of probability distributions taking value in $\mathcal{Y}$ with a density with respect to μ and indexed by Θ: $\mathcal{P} := \{p_\theta d\mu | \theta \in \Theta\}$.*

Example 4.1. *Here are a few examples of statistical parametric models based on well known family distributions:*

- *Bernoulli model: $\mathcal{Y} = \{0, 1\}$, $\Theta = [0, 1]$, and $p_\theta(k) = \theta^k (1-\theta)^{1-k}$ for $k \in \mathcal{Y}$.*
- *Binomial model: $\mathcal{Y} = \mathbb{N}$, $\Theta = [0, 1]$ and $p_\theta(k) = \binom{n}{k} \theta^k (1-\theta)^{n-k}$;*
- *Gaussian model: $\mathcal{Y} = \mathbb{R}$, $\Theta = \{(\mu, \sigma) \in \mathbb{R} \times \mathbb{R}_+\}$ and $p_{(\mu,\sigma)}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$*
- *Multidimensional Gaussian model: $\mathcal{Y} = \mathbb{R}^d$, $\Theta = \{(\mu, \Sigma) \in \mathbb{R}^d \times \mathcal{M}_d(\mathbb{R})\}$ and*

$$p_{(\mu,\Sigma)}(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1} (x-\mu)}.$$

- *Exponential model on $\mathcal{Y} = \mathbb{R}_+, \dots$*

Now, we assume that we are given some model $\mathcal{P}$ indexed by $\theta \in \Theta$ and we assume that the data D_n is generated independently from $p_{\theta^*} \in \mathcal{P}$ for some unknown parameter θ^*. We would like to recover the best parameter θ^* from the data. Note that in practice the data might come from a distribution which is not in $\mathcal{P}$: we call this misspecification but we will not enter into this details in this class.

4.1 Maximum likelihood estimation

The idea behind maximum likelihood estimation is to choose the most probable parameter $\theta \in \Theta$ for the observed data. Assume that $\mathcal{Y}$ is discrete and that $y \sim p_{\theta^*} d\mu$ for some $\theta^* \in \Theta$. Then, given any observation y_i, the probability that y takes the value y_i equals $p_{\theta^*}(y_i)$. Similarly, the probability of observing $(y_1, \dots, y_n) \in \mathcal{Y}^n$ if all the samples were sampled independently from p_θ is $\prod_{i=1}^n p_\theta(y_i)$. Hence, the high level idea of maximum likelihood estimation will be to maximize this probability over $\theta \in \Theta$. This is formalized by the definition of the likelihood which also holds for non-discret set $\mathcal{Y}$.

Definition 4.2 (Likelihood)**.** *Let $\mathcal{P} = \{p_\theta, \theta \in \Theta\}$ be a parametric model and $y \in \mathcal{Y}$. The likelihood of a data point x is the function $\theta \mapsto p_\theta(x)$. The likelihood $L(.|D_n)$ of a data set $D_n = (y_1, \dots, y_n)$ is the function*

$$L(\cdot|D_n) : \theta \mapsto \prod_{i=1}^{n} p_\theta(y_i) \, .$$

The maximum likelihood estimator (MLE) is then the parameter which maximizes the likelihood, i.e.,

$$\widehat{\theta}_n \in \operatorname*{arg\,max}_{\theta\in\Theta} \left\{ \prod_{i=1}^{n} p_\theta(y_i) \right\} \, .$$

This principle was proposed by Ronal Fisher in 1922 and was validated since with good theoretical properties. It is worth pointing out that since log is an increasing function, the maximum likelihood estimator can also be obtained by maximizing the log-likelihood:

$$\widehat{\theta}_n \in \operatorname*{arg\,max}_{\theta\in\Theta} \left\{ \sum_{i=1}^{n} \log(p_\theta(y_i)) \right\} \, . \tag{MLE}$$

This turns out to be much more convenient in practice because it is easier to maximize a sum than a product. Convince yourself by computing the gradients!

Examples

- Bernoulli model: $\mathcal{Y} = \{0, 1\}$, $\Theta = [0, 1]$, $p_\theta(y) = \theta^y(1-\theta)^{(1-y)}$. We assume that D_n was generated from a Bernoulli distribution of parameter θ^*, then the maximum likelihood estimator is:

  $$\widehat{\theta}_n = \operatorname*{arg\,min}_{0\leqslant\theta\leqslant1} \frac{1}{n} \sum_{i=1}^{n} \big(y_i \log\theta + (1-y_i)\log(1-\theta)\big) \, .$$

  Denoting $\bar{y}_n = \frac{1}{n}\sum_{i=1}^{n} y_i$ the empirical average and solving $d\log L(\widehat{\theta}_n|D_n)/d\theta = 0$ yields

  $$\frac{\bar{y}_n}{\widehat{\theta}_n} - \frac{1-\bar{y}_n}{1-\widehat{\theta}_n} = 0 \quad \Rightarrow (1-\bar{y}_n)\widehat{\theta}_n = (1-\widehat{\theta}_n)\bar{y}_n \quad \Rightarrow \quad \widehat{\theta}_n = \bar{y}_n \, .$$

  Therefore the maximum likelihood estimator is in this case the empirical mean.

- As an exercise, compute the maximum likelihood estimator for the models seen in Example 4.1.

Link with empirical risk minimization In density estimation, the goal is to find the density of the distribution which generated the data. Assuming that the density belongs to the model $\mathcal{P}$, the possible densities are p_θ, for $\theta \in \Theta$. A standard loss function in this setting is the negative log-likelihood: $\ell : (\theta, y) \in \Theta \times \mathcal{Y} \mapsto -\log\big(p_\theta(y)\big)$. The risk (or generalization error) is then:

$$\mathcal{R}(\theta) = -\mathbb{E}_y\big[\log(p_\theta(y))\big] \, .$$

In particular, if $y \sim p_{\theta^*}d\mu$ for some $\theta^* \in \Theta$, θ^* minimize the risk and the objective is to recover θ^*. The empirical risk is then by definition

$$\widehat{\mathcal{R}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \log\big(p_\theta(y_i)\big) \, .$$

Therefore, the empirical risk minimizer matches the estimator obtained from maximum likelihood in Equation (MLE).

Conditional modeling

Until now, we considered the problem of density estimation when the data set has only outputs $y_i \in \mathcal{Y}$. However, the principle of maximum likelihood can be extended to couples of input outputs $D_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ in $\mathcal{X} \times \mathcal{Y}$. We can then distinguish two different modeling:

- Generative modeling: we aim at estimating the density of couples of input outputs (x, y) among a family of densities $(x, y) \in \mathcal{X} \times \mathcal{Y} \mapsto p_\theta(x, y)$ on $\mathcal{X} \times \mathcal{Y}$. Then the risk and the empirical risks are:

$$\mathcal{R}(\theta) = -\mathbb{E}\big[\log(p_\theta(x, y)\big] \qquad \widehat{\mathcal{R}}(\theta) = -\frac{1}{n}\sum_{i=1}^n \log\big(p_\theta(x_i, y_i)\big)\,.$$

  This can be useful to generate some new samples (see what is obtained with GANs).
- Conditional modeling: we aim at estimating the density of an output y given an input x. The family of densities are now conditional densities $y \in \mathcal{Y} \mapsto p_\theta(.|x)$ on $\mathcal{Y}$ only but that depend on the inputs. The risks are then

$$\mathcal{R}(\theta) = -\mathbb{E}\big[\log(p_\theta(y|x)\big] \qquad \widehat{\mathcal{R}}(\theta) = -\frac{1}{n}\sum_{i=1}^n \log\big(p_\theta(y_i|x_i)\big)\,.$$

  This is useful if one want to predict the distribution or the value of a new output y given x.

4.2 Probabilistic interpretation of least-squares and logistic regression

4.2.1 Probabilistic insight of linear regression

We consider a data set $D_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of samples in $\mathcal{X} \times \mathcal{Y}$. We assume that the outputs y_i were independently generated from a Gaussian distribution of mean $\langle w, \varphi(x_i)\rangle$ and variance σ^2. In other words, we model an output y given an input x as

$$y = \langle w_*, \varphi(x)\rangle + \varepsilon, \qquad \text{where} \quad \varepsilon \sim \mathcal{N}(0, \sigma_*^2)\,.$$

for some unknown $\theta^* = (w_*, \sigma_*^2) \in \mathbb{R}^d \times \mathbb{R}_+$. Our family of possible conditional densities is indexed by parameters $\theta = (w, \sigma^2) \in \mathbb{R}^d \times \mathbb{R}_+$

$$p_\theta(y|x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(-\frac{\big(y - \langle w, \varphi(x)\rangle\big)^2}{2\sigma^2}\Big)\,.$$

The empirical risk (or conditional log-likelihood) is then

$$\widehat{\mathcal{R}}(\theta) = -\frac{1}{n}\sum_{i=1}^n \log\big(p_\theta(y_i|x_i)\big) = \frac{1}{2n\sigma^2}\sum_{i=1}^n \big(y_i - \langle w, \varphi(x_i)\rangle\big)^2 + \frac{1}{2}\log(2\pi\sigma^2)\,.$$

Therefore, the maximum likelihood estimator $\widehat{w}_n$ of w_* in a Gaussian model is the estimator obtained by least-squares linear regression. As an exercise, you may show that the maximum likelihood estimator for σ_* is

$$\widehat{\sigma}_n^2 = \frac{1}{n}\sum_{i=1}^n \big(y_i - \widehat{w}_n^\top \varphi(x_i)\big)^2\,.$$

Note that as we saw in the lecture on linear least-squares regression, the estimator $\widehat{\sigma}_n^2$ is biased: $\mathbb{E}[\widehat{\sigma}_n^2] = (1 - d/n)\sigma^2$.

4.2.2 Probabilistic insight of logistic regression

The advantage of the logistic loss with respect to the Hinge loss is that it has a probabilistic interpretation by modeling $\mathbb{P}(y=1|x)$. Denote by $p(x|y=1)$ the density of x when the label is 1 and by $p(x|y=0)$ the conditional density when the label is 0.

By Bayes rules, we have

$$\mathbb{P}(y=1|x) = \frac{p(x|y=1)\mathbb{P}(y=1)}{p(x|y=1)\mathbb{P}(y=1)+p(x|y=0)\mathbb{P}(y=0)} = \frac{1}{1+\frac{\mathbb{P}(y=0)p(x|y=0)}{\mathbb{P}(y=1)p(x|y=1)}} .$$

Denote by

$$f(x) := \log\left(\frac{\mathbb{P}(y=1|x)}{\mathbb{P}(y=0)|x)}\right) = \log\left(\frac{\mathbb{P}(y=1)}{\mathbb{P}(y=0)}\right) + \log\left(\frac{p(x|y=1)}{p(x|y=0)}\right)$$

the logarithmic ratio of the probability of observing y equals 1 with the one of observing $y=0$. Then,

$$\mathbb{P}(y=1|x) = \frac{1}{1+e^{-f(x)}} =: \sigma(f(x)) \qquad \text{with} \qquad \sigma(z) = \frac{1}{1+e^{-z}} .$$

The function σ is called the logistic function and satisfies $\sigma(-z) = 1-\sigma(z)$ et $\frac{\mathrm{d}\sigma(z)}{\mathrm{d}z} = \sigma(z)\sigma(-z)$. Its interest is that it allows to transform a function f with value in $\mathbb{R}$ into a probability between 0 and 1.

Then, the proposition below shows that performing maximum likelihood estimation on this probabilistic model with linear function $f(x) = \langle\theta,\varphi(x)\rangle$ is actually equivalent with logistic regression.

Proposition 4.1. *Assuming, that $(x_i,y_i)_{1\leqslant i\leqslant n}$ is a n-sample such that $\mathbb{P}(y_i=1|x_i) = \sigma(\langle\theta,\varphi(x)\rangle)$, then the maximum likelihood estimator of θ is*

$$\widehat{\theta}_{(logit)} \in \operatorname*{arg\,min}_{\theta\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n \ell\big(y_i,\langle\theta,\varphi(x_i)\rangle\big) ,$$

where $\ell(y,z) = y\log\big(1+e^{-z}\big) + (1-y)\log\big(1+e^{z}\big)$ is the logistic loss.

Proof. The log-likelihood can be written

$$\begin{aligned}\log L(\theta|D_n) &= \sum_{i=1}^n \log\big(\mathbb{P}_\theta(y_i=1|x_i)^{y_i}(1-\mathbb{P}_\theta(y_i=0|x_i)^{1-y_i}\big)\\ &= \sum_{i=1}^n \log\Big(\sigma\big(\langle\theta,\varphi(x_i)\rangle\big)^{y_i}\sigma\big(-\langle\theta,\varphi(x_i)\rangle\big)^{1-y_i}\Big)\\ &= -\sum_{i=1}^n \ell\big(y_i,\langle\theta,\varphi(x_i)\rangle\big)\end{aligned}$$

where ℓ is the logistic loss. Therefore,

$$\operatorname*{arg\,max}_{\theta\in\mathbb{R}^d} \log L(\theta|D_n) = \operatorname*{arg\,min}_{\theta\in\mathbb{R}^d} \sum_{i=1}^n \ell\big(y_i,\langle\theta,\varphi(x_i)\rangle\big) .$$

The logistic regression estimator is therefore the maximum likelihood estimator. □

Exemple: Gaussian mixture. Assume that you have classes such that the covariate

$$x \quad \text{follows} \quad \begin{cases} \mathcal{N}(\mu_0, \Sigma_0) & \text{if } y = 0 \\ \mathcal{N}(\mu_1, \Sigma_1) & \text{if } y = 1 \end{cases},$$

for some $\mu_0, \mu_1 \in \mathbb{R}^p$ and symmetric definite positive matrices $\Sigma_0, \Sigma_1 \in \mathbb{R}^{p \times p}$. The data is plotted in Figure 4. Then, the density of x for each class $i \in \{0, 1\}$ is

$$p(x|y = i) = \det\big(2\pi\Sigma_i\big)^{-1/2} \exp\Big(-\frac{1}{2}(x - \mu_i)^\top \Sigma_i^{-1}(x - \mu_i)\Big) .$$

Therefore,

$$\begin{aligned} f(x) &= \log\left(\frac{\mathbb{P}(y = 1)}{\mathbb{P}(y = 0)}\right) + \log\left(\frac{p(x|y = 1)}{p(x|y = 0)}\right) \\ &= \log\left(\frac{\mathbb{P}(y = 1)}{\mathbb{P}(y = 0)}\right) + \frac{1}{2}\log\det\Sigma_0 - \frac{1}{2}\log\det\Sigma_1 \\ &\qquad\qquad + \frac{1}{2}(x - \mu_0)^\top \Sigma_0^{-1}(x - \mu_0) - \frac{1}{2}(x - \mu_1)^\top \Sigma_1^{-1}(x - \mu_1) . \end{aligned}$$

It is a quadratic function in $x \in \mathbb{R}^p$. There exists thus $\theta \in \mathbb{R}^d$ with $d = p(p + 1)$ such that $f(x) = \langle \theta, \varphi(x) \rangle$ with quadratic features

$$\varphi(x) = (1, x_1, x_2, \dots, x_d, x_1x_2, x_1x_3, \dots) \in \mathbb{R}^d .$$

Therefore, Gaussian mixtures fits into this probabilistic model. And logistic regression with quadratic features $\varphi(x)$ corresponds to maximum likelihood estimation in this model. Actually, logistic regression incorporates many possible laws for $p(x|y = i)$ beyond Gaussian.

The classification performed with logistic regression (or maximum likelihood) is plotted in Figure 4 on a few examples. As an exercise you might reproduce this examples in python.

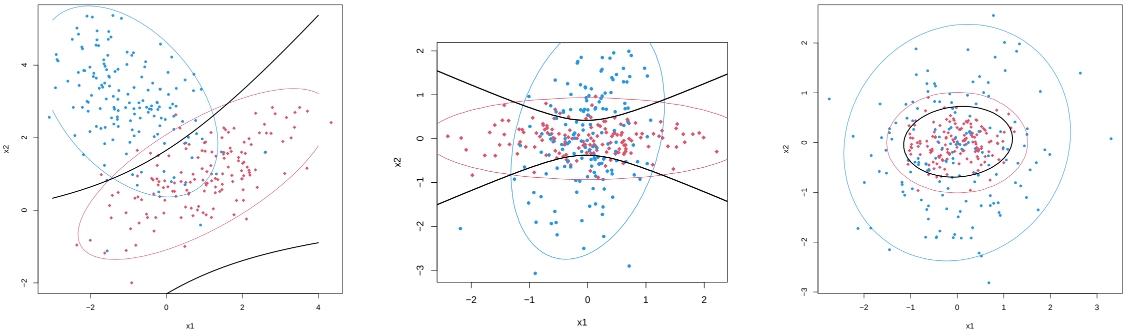


Figure 4: Examples of logistic regression classification for mixtures of Gaussian. Each covariate x is sampled from a 2-dimensional Gaussian distribution in $\mathbb{R}^2$ according to the class $y = 0$ and $y = 1$. The frontier of logistic regression with 2-degree polynomials as features is plotted in black line.

5 K-Nearest Neighbors

⚠ This chapter gives a detailed analysis of the consistency of kNN and is available for the curious student. The student will not be expected to master the analysis precisely for the final exam. The student will just be expected to have a general idea of the nearest neighbor estimator and how it works.

Learning objectives: understand the main concepts of k-nearest neighbors, how to compute it, when does it converge to the optimal classifier, definition of a plug-in estimator, how to compute the risk in simple cases

We would like to classify objects, described with vectors x in $\mathbb{R}^d$, among $L+1$ classes $\dagger := \{0, \dots, L\}$ in an automatic fashion. 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 $1 \leqslant i \leqslant n$. The data is assumed to be i.i.d. random variables from a distribution ν. The goal of this lesson is to build a classifier, i.e., a function

$$f : \mathbb{R}^d \to \mathcal{Y}$$

which minimizes the probability of mistakes: $\mathbb{P}_{(x,y)\sim\nu}\big\{f(x) \neq y\big\}$. The latter can be rewritten as the expected risk $\mathcal{R}(f) := \mathbb{E}_{(x,y)\sim\nu}\big[\mathbb{1}_{f(x)\neq y}\big]$ of the 0-1 loss $\ell(y, z) = \mathbb{1}\{y \neq z\}$.

In previous lectures, we considered linear least-squares and logistic regression, wich are based on the empirical risk minimization approach:

$$\widehat{f} \in \operatorname*{arg\,min}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i))$$

for some parametric function set $\mathcal{F}$. In this lecture, we will see another approach based on local averaging.

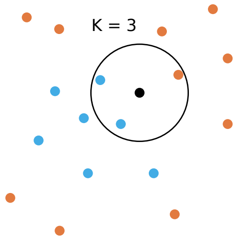


Figure 5: k-nearest neighbors with two classes (orange and blue) and $k = 3$. The new input (i.e., the black point) is classified as blue which corresponds to the majority class among its three nearest neighbors.

The k-nearest neighbors classifier works as follows. Given a new input $x \in \mathbb{R}^d$, it looks at the k nearest points x_i in the data set $D_n = \{(x_i, y_i)\}$ and predicts a majority vote among them. The k-nearest neighbors classifier is quite popular because it is simple to code and to understand; it has nice theoretical guarantees as soon as k is appropriately chosen and performs reasonably well in low dimensional spaces. In this notes, we will investigate the following questions:

- consistency: does k-NN has the smallest possible probability of error when the number of data grows?

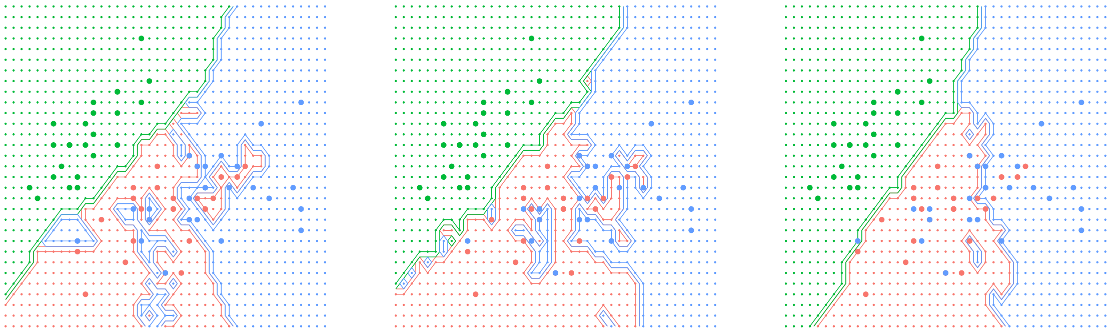


Figure 6: Prediction landscape of the k-NN classifier with three classes with $k = 1$ (left) $k = 3$ (middle), and $k = 20$ (right).

- how to choose k?

There are plenty of other possible interesting questions. How should we choose the metric (invariance properties,...)? Can we get improved performance by using different weights between neighbors (local averaging methods)? Is it possible to improve the computational complexity (by reducing the data size or keeping some data in memory,...). These questions are however beyond the scope of these lecture notes and we refer the interested reader to the book Devroye et al., 2013.

⚠ k-NN may also be used for regression by predicting the average value of the y_i among the nearest neighbors instead of doing majority vote.

Pros and cons of kNN

- *Pros:* The algorithm does not require any learning (no use of gradient descent or any optimization algorithm to train the algorithm). Furthermore, it is very easy to implement. Finally, it can get good performance in practice and is theoretically optimal as we will see in this lecture.
- *Cons:* The algorithm is slow at query time (to make a prediction) since it must pass through all data for each prediction (to compute which are the nearest neighbors). Moreover it may be easily fooled by irrelevant outputs and it has poor performance for high-dimensional data ($d \gtrsim 20$).

What hyper-parameters? When applying kNN, the learner must make two decisions: the number of neighbors k and the metric $\|\cdot\|$. The number of neighbors balances the bias-variance tradeoff. Figure 6 shows the predictions obtained with kNN for different values of k: the boundary becomes smoother as k increases. The metric (or distance between neighbors) is also a crucial choice, especially for complex data (structured data like images, speeches or graphs).

5.1 Bayes classifier and plug-in estimator

Assumptions and notation For simplicity, we assume the binary case: $L = 1$ and $\mathcal{Y} = \{0, 1\}$. And we define the function $\eta : \mathbb{R}^d \to [0, 1]$ by:

$$\eta(x') := \mathbb{P}_{(x,y)\sim\nu}\big(y = 1 | x = x'\big) \qquad \forall x' \in \mathbb{R}^d\,. \tag{7}$$

⚠ In the following except when stated otherwise the expectation and probability are according to $(x, y) \sim \nu$. For clarity, we will omit the subscript $(x, y) \sim \nu$ in $\mathbb{E}$ and $\mathbb{P}$. In some cases, if the classifier is random, for instance because it was build on the random data set (x_i, y_i) the expectation might also be taken with respect to the classifier itself. But it will be explicited.

Lemma 5.1. *For any (deterministic) classifier $f : \mathbb{R}^d \to \mathcal{Y}$,*

$$\mathcal{R}(f) = \mathbb{E}\big[\eta(x)\mathbb{1}_{f(x)=0} + (1-\eta(x))\mathbb{1}_{f(x)=1}\big] .$$

Proof. Let f be a classifier, then

$$\begin{aligned}
\mathcal{R}(f) &= \mathbb{E}\big[\mathbb{1}_{f(x)\neq y}\big] = \mathbb{E}\big[\mathbb{E}\big[\mathbb{1}_{f(x)\neq y}|x\big]\big] \\
&= \mathbb{E}\big[\mathbb{E}\big[\mathbb{1}_{f(x)\neq y}|x, y=1\big]\mathbb{P}\{y=1|x\} + \mathbb{E}\big[\mathbb{1}_{f(x)\neq y}|x, y=0\big]\mathbb{P}\{y=0|x\}\big] \\
&= \mathbb{E}\big[\mathbb{E}\big[\mathbb{1}_{f(x)\neq 1}|x, y=1\big]\eta(x) + \mathbb{E}\big[\mathbb{1}_{f(x)\neq 0}|x, y=0\big](1-\eta(x))\big] \\
&= \mathbb{E}\big[\mathbb{1}_{f(x)=0}\eta(x) + \mathbb{1}_{f(x)=1}(1-\eta(x))\big] .
\end{aligned}$$

□

The (optimal) Bayes classifier. It is worth to notice that a random classifier sampling $f(x) = 0$ and $f(x) = 1$ with probability $1/2$ has an expected risk $1/2$. Hence, we will only focus on non-trivial classifiers that outperform this expected error. If the function η was known, one could define the Bayes classifier as follows:

$$f^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geqslant 1/2 \\ 0 & \text{otherwise} \end{cases}$$

Lemma 5.2. *The risk of the Bayes classifier is*

$$\mathcal{R}^* := \mathcal{R}(f^*) = \mathbb{E}\big[\min\{\eta(x), 1-\eta(x)\}\big] .$$

Furthermore, for any classifier f we have

$$\mathcal{R}(f) - \mathcal{R}^* = \mathbb{E}\big[|2\eta(x)-1|\mathbb{1}_{f(x)\neq f^*(x)}\big] \geqslant 0.$$

The above lemma implies that the Bayes classifier is optimal and $\mathcal{R}^* = \min_{f:\mathbb{R}^d \mapsto \{0,1\}} \mathcal{R}(f)$. The goal of this lesson is to build a classifier that gets close to $\mathcal{R}^*$. We call such estimator consistent.

Definition 5.1 (Consistency)**.** *We say that an estimator $\widehat{f}_n$ is consistent if*

$$\mathbb{E}_{(x_i,y_i)\sim\nu}\big[\mathcal{R}(\widehat{f}_n)\big] \underset{n\to+\infty}{\longrightarrow} \mathcal{R}^*.$$

Proof. Applying Lemma 5.1, we get from the definition of f^*

$$\begin{aligned}
\mathcal{R}^* &= \mathbb{E}\big[\eta(x)\mathbb{1}_{f^*(x)=0} + (1-\eta(x))\mathbb{1}_{f^*(x)=1}\big] \\
&= \mathbb{E}\big[\eta(x)\mathbb{1}_{\eta(x)<1/2} + (1-\eta(x))\mathbb{1}_{\eta(x)\geqslant 1/2}\big] \\
&= \mathbb{E}\big[\min\{\eta(x), 1-\eta(x)\}\big], .
\end{aligned}$$

Furthermore, let $f : \mathbb{R}^d \to \mathcal{Y}$, then

$$\begin{aligned}
\mathcal{R}(f) - \mathcal{R}^* &= \mathbb{E}\big[\eta(x)(\mathbb{1}_{f(x)=0} - \mathbb{1}_{f^*(x)=0}) + (1-\eta(x))(\mathbb{1}_{f^*(x)=1} - \mathbb{1}_{f^*(x)=1})\big] \\
&= \mathbb{E}\big[(2\eta(x)-1)(\mathbb{1}_{f(x)=0} - \mathbb{1}_{f^*(x)=0})\big] \\
&= \mathbb{E}\big[(2\eta(x)-1)\mathbb{1}_{f(x)\neq f^*(x)}\operatorname{sign}(1 - 2\mathbb{1}_{f^*(x)=0})\big]
\end{aligned}$$

But $\operatorname{sign}(1-2\mathbb{1}_{f^*(x)=0}) = \operatorname{sign}(1-2\mathbb{1}_{\eta(x)\leqslant 1/2}) = \operatorname{sign}(2\eta(x)-1)$ which concludes the proof. □

Therefore, if η was known, one could compute the optimal classifier f^*. However, η is unknown and one should thus estimate it.

Plug-in estimator Let $\widehat{\eta}_n$ be an estimator of η, i.e., $\widehat{\eta}_n$ is a function of the training data $D_n = (x_i, y_i)_{1 \leqslant i \leqslant n}$ which takes values in the functions from $\mathbb{R}^d$ to $[0,1]$. We will omit in the following the dependence of $\widehat{\eta}_n$ in the data D_n. From $\widehat{\eta}_n$, we can build the plug-in estimator as follows:

$$\widehat{f}_n(x) = \begin{cases} 1 & \text{if } \widehat{\eta}_n(x) \geqslant 1/2 \\ 0 & \text{otherwise} \end{cases} . \tag{8}$$

Hopefully, if $\widehat{\eta}_n$ is close enough to η the estimator $\widehat{f}_n$ will be also close to f^* and will have a small risk. This is formalized be the following Lemma.

Lemma 5.3. *If $\widehat{f}_n$ is defined in* (8)*, then $\mathcal{R}(\widehat{f}_n) - \mathcal{R}^* \leqslant 2\mathbb{E}_{(x,y)\sim\nu}\big[|\eta(x) - \widehat{\eta}_n(x)| \, |D_n\big]$.*

Proof. From Lemma 5.2, we have

$$\mathcal{R}(\widehat{f}_n) - \mathcal{R}^* = 2\mathbb{E}\big[|\eta(x) - 1/2|\mathbb{1}_{\widehat{f}_n(x) \neq f^*(x)}|D_n\big].$$

Thus, to prove the Lemma it suffices to show that almost surely,

$$|\eta(x) - 1/2|\mathbb{1}_{\widehat{f}_n(x) \neq f^*(x)} \leqslant |\eta(x) - \widehat{\eta}_n(x)| .$$

We can assume that $\mathbb{1}_{\widehat{f}_n(x) \neq f^*(x)} \neq 0$ (otherwise the inequality is true). This implies that $\widehat{\eta}_n(x) - 1/2$ and $\eta(x) - 1/2$ have opposite sign. In particular it yields

$$|\eta(x) - 1/2| \leqslant |\eta(x) - 1/2| + |1/2 - \widehat{\eta}_n(x)| = |\eta(x) - \widehat{\eta}_n(x)|$$

which concludes the proof. □

The above Lemma shows first, if $\widehat{\eta}_n = \eta$, then the plug-in classifier $\widehat{f}_n$ is Bayes optimal. Second, if $\widehat{\eta} \approx \eta$, then $\widehat{f}_n$ is close to f^*. Therefore, if we could build from the data an estimator $\widehat{\eta}_n$ of η such that

$$\mathbb{E}_{(x_i,y_i)\sim\nu}\big[|\eta(x) - \widehat{\eta}_n(x)|\big] \underset{n\to+\infty}{\longrightarrow} 0$$

then the associated plugin classifier $\widehat{f}_n$ would be consistent (see Definition 5.1). The reverse if not true: estimating η is harder then estimating f^*. We will show that the k-nearest neighbors satisfy the above convergence if the number of neighbors grows appropriately. This is not the case for fixed numbers of neighbors.

5.2 The k-nearest neighbors classifier (kNN)

The kNN classifiers classifies a new input x with the majority class among its k-nearest neighbors (see Figure 5). More formally, we denote by $x_{(i)}(x)$ the i-th nearest neighbor of $x \in \mathbb{R}^d$ (using the Euclidean distance) among the inputs x_i, $1 \leqslant i \leqslant n$. We have for all $x \in \mathbb{R}^d$

$$\|x - x_{(1)}(x)\| \leqslant \|x - x_{(2)}(x)\| \leqslant \ldots \leqslant \|x - x_{(n)}(x)\|$$

and $x_{(i)}(x) \in \{x_1, \ldots, x_n\}$ for all $1 \leqslant i \leqslant n$. We denote by $y_{(i)}(x) \in \{0,1\}$ the label of the $i-th$ neighbor. We can then define

$$\widehat{\eta}_n^k(x) = \frac{1}{k}\sum_{i=1}^k y_{(i)}(x) = \frac{1}{k}\sum_{i=1}^n y_i \mathbb{1}_{x_i \in \{x_{(1)}(x), \ldots, x_{(k)}(x)\}}$$

and $\widehat{f}_n^k$ the kNN classifier is the plugin estimator defined in (8). We denote by

$$\mathcal{R}_{kNN} := \lim_{n\to\infty} \mathbb{E}_{(x_i,y_i)\sim\nu}\big[\mathcal{R}(\widehat{f}_n^k)\big]$$

the asympotic risk of the k-nearest neighbor classifier.

Example 5.1. *Let's consider a concrete example. Suppose we have two biased dice that we roll heads or tails. The feature x represents the die that we throw: $x = 1$ if we throw the first die and $x = 2$ if we throw the second. The player is asked to predict the outcome of a rolled die that was picked uniformly at random between the two. Let's say that the first die has a $3/4$ probability of turning up heads ($y = 1$) and the second has a $2/3$ probability of turning up tails ($y = 0$). We can formalize this as:*

$$\mathbb{P}(y = 1|x = 1) = \frac{3}{4} \qquad \textit{and} \qquad \mathbb{P}(y = 1|x = 0) = \frac{1}{3}\,.$$

These probabilities are unknown to the player, but the player has access to as many rolls of each of the two dice as he or she wishes before making a prediction.

In this example, the function $\eta : x \mapsto \mathbb{P}(y = 1|x)$ equals $\eta(1) = 3/4$ and $\eta(2) = 1/3$. The optimal prediction is thus to predict head ($y = 1$) if the first die $x = 1$ is rolled and tail ($y = 0$) if it is the second ($x = 2$). The optimal probability of error is thus

$$\begin{aligned}\mathcal{R}^* &= \mathbb{P}(\textit{ tail } |\textit{ 1st die is rolled })\mathbb{P}(\textit{ 1st die is rolled })\\ &\qquad\qquad + \mathbb{P}(\textit{ head } |\textit{ 2nd die is rolled })\mathbb{P}(\textit{ 2nd die is rolled })\\ &= \mathbb{P}(y = 0|x = 1)\mathbb{P}(x = 1) + \mathbb{P}(y = 1|x = 2)\mathbb{P}(x = 2) = \frac{1}{4}\times\frac{1}{2} + \frac{1}{3}\times\frac{1}{2} = \frac{7}{24} \approx 0.29\,.\end{aligned}$$

A player that would play the 1-Nearest neighbor would see what die is rolled, used a single past outcome of that die to make its prediction. For the first die, the player would thus predict tail with probability $3/4$ and head with probability $1/4$. Its probability of error is thus

$$\begin{aligned}\mathcal{R}_{\textit{1-NN}} &= \mathbb{P}(\textit{ head } |\textit{ 1st die is rolled })\mathbb{P}(\textit{ 1st die is rolled and player predicts tail })\\ &\qquad + \mathbb{P}(\textit{ tail } |\textit{ 1st die is rolled })\mathbb{P}(\textit{ 1st die is rolled and player predicts head })\\ &\qquad + \mathbb{P}(\textit{ head } |\textit{ 2nd die is rolled })\mathbb{P}(\textit{ 2nd die is rolled and player predicts tail })\\ &\qquad + \mathbb{P}(\textit{ tail } |\textit{ 2nd die is rolled })\mathbb{P}(\textit{ 2nd die is rolled and player predicts head })\\ &= \mathbb{P}(y = 0|x = 1)\mathbb{P}(x = 1)\frac{3}{4} + \mathbb{P}(y = 1|x = 0)\mathbb{P}(x = 1)\frac{1}{4}\\ &\qquad + \mathbb{P}(y = 1|x = 2)\mathbb{P}(x = 2)\frac{2}{3} + \mathbb{P}(y = 0|x = 2)\mathbb{P}(x = 2)\frac{2}{3}\\ &= \frac{1}{4}\times\frac{1}{2}\times\frac{3}{4} + \frac{3}{4}\times\frac{1}{2}\times\frac{1}{4} + \frac{1}{3}\times\frac{1}{2}\times\frac{2}{3} + \frac{2}{3}\times\frac{1}{2}\times\frac{2}{3}\\ &= \frac{59}{144} \approx 0.41\,.\end{aligned}$$

This is better than random guess but worth than optimal. If the player would use more samples of previous rolls, its error would decrease. This is what we show next.

5.3 The nearest neighbor classifier

Theorem 5.4 (Inconsistency of the 1-nearest neighbor)**.** *The asymptotic risk of the 1-nearest neighbor satisfies $\mathcal{R}^* \leqslant \mathcal{R}_{kNN} = \mathbb{E}\big[2\eta(x)(1-\eta(x))\big] \leqslant 2\mathcal{R}^*(1-\mathcal{R}^*)$.*

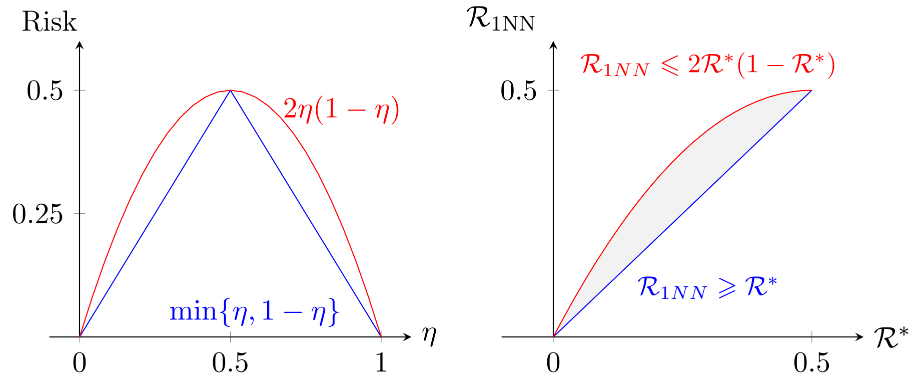


Figure 7: [left] Risk of the 1-nearest neighbor and optimal risk according to η. [right] The risk of the 1-nearest neighbor lies in the dotted area in-between the blue curve (optimal risk) and the red curve (upper-bound of Theorem 5.4).

Sketch of proof of Theorem 5.4. We do not provide the complete proof here but only a sketch with the main idea. We refer the curious reader to Devroye et al., 2013 for the rigorous argument. Let $(x, y) \sim \nu$ be some new input. From (7), knowing x the label y follows a Bernoulli distribution with parameter $\eta(x)$. When the number n of data points increases the nearest neighbor of x gets closer to x (this has to be made rigorous since x is a random variable). Thus by continuity of η, given x when $n \to \infty$, we also have $y_{(1)}(x) \sim \mathcal{B}(\eta(x))$. Therefore,

$$\lim_{n\to\infty} \mathbb{E}_{(x_i,y_i)\sim\nu}\big[\mathcal{R}(\widehat{f}_n^1)\big] = \mathbb{P}\big\{y_{(1)}(x) \neq y\big\}$$

where $y_{(1)}(x), y \sim \mathcal{B}(\eta(x))$ are independent given x. The probability of error is thus

$$\begin{aligned}
\mathbb{P}\big\{y_{(1)}(x) \neq y\big\} &= \mathbb{E}_{(x,y)\sim\nu}\big[\mathbb{P}\big\{y_{(1)}(x) \neq y|x\big\}\big] \\
&= \mathbb{E}_{(x,y)\sim\nu}\big[P\{y = 1, y_{(1)}(x) \neq 1|x\} + \mathbb{P}\big\{y \neq 1, y_{(1)}(x) = 1|x\big\}\big] \\
&= \mathbb{E}_{(x,y)\sim\nu}\big[\mathbb{P}\{y = 1|x\}\mathbb{P}\big\{y_{(1)}(x) \neq 1|x\big\} + \mathbb{P}\{y \neq 1|x\}\mathbb{P}\big\{y_{(1)}(x) = 1|x\big\}\big] \\
&= \mathbb{E}_{(x,y)\sim\nu}\big[2\eta(x)\big(1-\eta(x)\big)\big] .
\end{aligned}$$

This concludes the first equality of the Theorem. As for the second, denoting $\mathcal{R}(x) := \min\{\eta(x), 1-\eta(x)\}$, we have

$$\mathbb{E}\big[\eta(x)\big(1-\eta(x)\big)\big] = \mathbb{E}\big[\mathcal{R}(x)(1-\mathcal{R}(x)\big] \overset{\text{Concavity}}{\leqslant} \mathbb{E}[\mathcal{R}(x)]\big(1-\mathbb{E}[\mathcal{R}(x)]\big) = \mathcal{R}^*(1-\mathcal{R}^*) .$$

□

The 1-nearest neighbor is therefore not consistent as shown in Figure 7 as soon as the optimal risk is not trivial: $\mathcal{R}^* \notin \{0, 1/2\}$. This result was first proved by Cover and Hart, 1967 with assumptions on ν and η and by Stone, 1977 without any assumption. It is worth to stress that this result is completely distribution free (independent of ν and η). The smoothness of ν and η does not matter for the limit, it only changes the rate of convergence.

5.4 Inconsistency of the k-NN classifier (fixed k)

Therefore, a single neighbor is not sufficient to approach the optimal risk $\mathcal{R}^*$. Actually, we could prove a similar result for any fixed number of neighbors. It is convenient to let k be odd to avoid ties. We refer to Devroye et al., 2013 for the proof.

Theorem 5.5. *Let $k \geqslant 1$ be odd and fixed. Then, the asymptotic risk of the k-nearest neighbor satisfies*

$$\mathcal{R}_{kNN} = \mathbb{E}_X\left[\sum_{j=0}^{k}\binom{k}{j}\eta(x)^j(1-\eta(x))^{k-j}\big(\eta(x)\mathbb{1}_{j<k/2} + (1-\eta(x))\mathbb{1}_{j>k/2}\big)\right]$$
$$= \mathcal{R}^* + \mathbb{E}\Big[\big|2\eta(x)-1\big|\mathbb{P}\Big\{\text{Binomial}(k, \min\{\eta(x), 1-\eta(x)\}) > \frac{k}{2}\Big|x\Big\}\Big]\,.$$

Sketch of proof of Theorem 5.5. Similarly to Theorem 5.4, we only provide an idea of the proof. Let $(x,y) \sim \nu$ be a new data point. When the number of data goes to infinity, the nearest neighbors $x_{(1)}(x), \dots, x_{(k)}(x)$ of x get closer to x (to be proved rigorously) and given x their labels $y_{(1)}(x), \dots, y_{(k)}(x)$ are i.i.d. Bernoulli random variables with parameter $\eta(x)$. The k-NN classifier predicts

$$\widehat{f}_n^k = \left\{\begin{array}{ll} 1 & \text{if} \quad y_{(1)}(x) + \cdots + y_{(k)}(x) > \frac{k}{2} \\ 0 & \text{if} \quad y_{(1)}(x) + \cdots + y_{(k)}(x) < \frac{k}{2} \end{array}\right. .$$

The asymptotic probability of error of the k-NN classifier is thus

$$\begin{aligned}\mathcal{R}_{kNN} &= \lim_{n\to\infty}\mathbb{P}\{\widehat{f}_n^k \neq y\} \\ &= \mathbb{P}\left\{y_{(1)}(x) + \cdots + y_{(k)}(x) < \frac{k}{2}, y = 1\right\} + \mathbb{P}\left\{y_{(1)}(x) + \cdots + y_{(k)}(x) > \frac{k}{2}, y = 0\right\} \\ &= \mathbb{E}_X\Bigg[\underbrace{\mathbb{P}\{y=1|x\}}_{\eta(x)}\mathbb{P}\Bigg\{\underbrace{y_{(1)}(x) + \cdots + y_{(k)}(x)}_{\text{Binomial}(k,\eta(x))} > \frac{k}{2}|x\Bigg\} \\ &\qquad + \underbrace{\mathbb{P}\{y=0|x\}}_{1-\eta(x)}\mathbb{P}\Bigg\{\underbrace{y_{(1)}(x) + \cdots + y_{(k)}(x)}_{\text{Binomial}(k,\eta(x))} < \frac{k}{2}\Bigg\}\Bigg],\end{aligned}$$

where given x, $y_{(1)}(x) + \cdots + y_{(k)}(x), y$ are i.i.d. independent Bernoulli random variables with parameter $\eta(x)$. This proves the first equality.

$$\mathcal{R}_{kNN} = \mathbb{E}_X\big[\alpha(\eta(x))\big]$$

where

$$\alpha(p) := p\,\mathbb{P}\Big\{\text{Binomial}(k,p) < \frac{k}{2}\Big\} + (1-p)\,\mathbb{P}\Big\{\text{Binomial}(k,p) > \frac{k}{2}\Big\}\,.$$

If $p < 1/2$, then $p < 1-p$ and

$$\begin{aligned}\alpha(p) &= p\left(1 - \mathbb{P}\Big\{\text{Binomial}(k,p) > \frac{k}{2}\Big\}\right) + (1-p)\,\mathbb{P}\Big\{\text{Binomial}(k,p) > \frac{k}{2}\Big\} \\ &= p + (1-2p)\,\mathbb{P}\Big\{\text{Binomial}(k,p) > \frac{k}{2}\Big\}\,.\end{aligned}$$

Following the same calculation for $p > 1/2$ yields

$$\alpha(p) = \min\{p, 1-p\} + |2p-1|\mathbb{P}\Big\{\text{Binomial}\big(k, \min\{p,1-p\}\big) > \frac{k}{2}\Big\}$$

which concludes the proof using that $\mathcal{R}^* = \mathbb{E}_X\big[\min\{\eta(x), 1-\eta(x)\}\big]$. □

The previous Theorem may provide nice inequalities on $\mathcal{R}_{kNN}$ as shown by the next corollary.

Corollary 5.6. *We have $\mathcal{R}^* \leqslant \ldots \leqslant \mathcal{R}_{5NN} \leqslant \mathcal{R}_{3NN} \leqslant \mathcal{R}_{1NN} \leqslant 2\mathcal{R}^*(1-\mathcal{R}^*)$. Furthermore, let $k \geqslant 1$ be odd and fixed. Then, the asymptotic risk of the k-NN classifier satisfies*

$$\mathcal{R}_{kNN} \leqslant \mathcal{R}^* + \frac{1}{\sqrt{ke}}\,.$$

Proof. The first inequalities are because $\mathbb{P}\Big\{\text{Binomial}\big(k,p\big) > \frac{k}{2}\Big\}$ decreases in k for $p < 1/2$. Let $0 \leqslant p \leqslant 1/2$ and $B \sim \text{Binomial}(k,p)$. Then,

$$\begin{aligned}(1-2p)\,\mathbb{P}\Big\{B > \frac{k}{2}\Big\} &= (1-2p)\,\mathbb{P}\Big\{\frac{B-kp}{k} > \frac{1}{2} - p\Big\}\\ &\overset{(*)}{\leqslant} (1-2p)e^{-2k(1/2-p)^2}\\ &\leqslant \sup_{0\leqslant u\leqslant 1} ue^{-ku^2/2}\\ &= \frac{1}{\sqrt{ke}}\,,\end{aligned}$$

where $(*)$ is by the Okamoto-Hoeffding inequality that we recall below (see Devroye et al., 2013, Thm 8.1).

Lemma 5.7 (Okamoto-Hoeffding inequality)**.** *Let $x_1, \ldots, x_n$ be independant bounded random variables such that $x_i \in [a_i, b_i]$ almost surely. Then, for all $\varepsilon > 0$*

$$\mathbb{P}\left\{S_n - \mathbb{E}[S_n] \geqslant \varepsilon\right\} \leqslant e^{\frac{-2\varepsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}}\,,$$

where $S_n = \sum_{i=1}^n x_i$.

□

Therefore the asymptotic error of the k-NN classifier decreases with k but is not consistent: for any fixed k, it does not converge to the optimal risk $\mathcal{R}^*$. The idea is thus to make $k \to \infty$ when n grows.

5.5 Consistent nearest neighbors making $k \to \infty$

Theorem 5.8 (Stone 1964)**.** *If $k(n) \to \infty$ and $\frac{k(n)}{n} \to 0$ then the $k(n)$-NN classifier is universally consistent: for all distribution ν, we have*

$$\mathcal{R}_{k(n)NN} := \lim_{n\to\infty} \mathbb{E}_{(x_i,y_i)\sim\nu}\big[\mathcal{R}(\widehat{f}_n^k)\big] = \mathcal{R}^*\,.$$

Historically, this is the first universally consistent algorithm. The proof is not trivial and comes from a more general result (Stone's Theorem) on "Weighted Average Plug-in" classifiers (WAP).

Definition 5.2 (Weighted Average Plug-in classifier (WAP))**.** *Let $D_n = \{(x_1,y_1), \ldots, (x_n,y_n)\}$, a WAP classifier is a plug-in estimator $\widehat{f}_n$ associated to an estimator of the form*

$$\widehat{\eta}_n(x) = \sum_{i=1}^n w_{n,i}(x) y_i$$

where the weights $w_{n,i}(x) = w_{n,i}(x, x_1, \ldots, x_n)$ are non negative and sum to one.

This is the case of the k-NN classifier which satisfies

$$w_{n,i}(x) = \begin{cases} \frac{1}{k} & \text{if } x_i \text{ is a } k\text{NN of } x \\ 0 & \text{otherwise} \end{cases} .$$

Theorem 5.9 (Stone 1977). *Let $(f_n)_{n\geqslant 0}$ a WAP such that for all distribution ν the weights $w_{n,i}$ satisfy*

a) *it exists $c > 0$ s.t. for all non-negative measurable function f with $\mathbb{E}[f(x)] < \infty$,*

$$\mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x) f(x_i)\right] \leqslant c\mathbb{E}\big[f(x)\big] ;$$

b) *for all $a > 0$, $\mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x)\mathbb{1}_{\|x_i - x\| > a}\right] \underset{n\to+\infty}{\longrightarrow} 0$*

c) $\mathbb{E}\big[\max_{1\leqslant i\leqslant n} w_{n,i}(x)\big] \underset{n\to+\infty}{\longrightarrow} 0$

Then, the plug-in estimator associated with $\widehat{\eta}_n(x) = \sum_{i=1}^{n} w_{n,i}(x) y_i$ is universally consistent

$$\lim_{n\to\infty} \mathbb{E}\big[\mathcal{R}(\widehat{f}_n)\big] = \mathcal{R}^* .$$

Let us make some remarks about the conditions:

a) is a technical condition

b) says that the weights of points outside of a ball around x should vanish to zero. Only the x_i in a smaller and smaller neighborhood of x should contribute.

c) says that no point should have a too important weight. The number of points in the local neighborhood of x should increase to ∞.

Proof of Theorem 5.9. From Lemma 5.3 together with Cauchy-Schwarz, it suffices to show that $\mathbb{E}\big[(\eta(x) - \widehat{\eta}_n(x))^2\big] \underset{n\to+\infty}{\longrightarrow} 0$. Let us introduce

$$\tilde{\eta}_n(x) := \sum_{i=1}^{n} w_{n,i}(x) \underbrace{\eta(x_i)}_{\text{instead of } y_i \text{ in } \widehat{\eta}_n}$$

in which we replaced y_i in $\widehat{\eta}_n$ with $\eta(x_i)$ which we recall:

$$\widehat{\eta}_n(x) = \sum_{i=1}^{n} w_{n,i}(x) y_i \qquad \text{and} \qquad \eta(x) = \sum_{i=1}^{n} w_{n,i}(x)\eta(x) .$$

Using $(a + b)^2 \leqslant 2a^2 + 2b^2$, we have

$$\mathbb{E}\big[(\eta(x) - \widehat{\eta}_n(x))^2\big] \leqslant 2\underbrace{\mathbb{E}\big[(\eta(x) - \tilde{\eta}_n(x))^2\big]}_{(1)} + 2\underbrace{\mathbb{E}\big[(\tilde{\eta}_n(x) - \widehat{\eta}_n(x))^2\big]}_{(2)} .$$

We will upper-bound (1) and (2) independently.

(1) For simplicity, to bound this term we assume η to be absolutely continuous: let $\varepsilon > 0$, it exists $a > 0$ such that $\|x - x'\| \leqslant a \Rightarrow (\eta(x) - \eta(x'))^2 \leqslant \varepsilon$. Then,

$$\begin{aligned}
(1) &= \mathbb{E}\left[\left(\sum_{i=1}^{n} w_{n,i}(x)\big(\eta(x) - \eta(x_i)\big)\right)^2\right] \\
&\overset{\text{Jensen}}{\leqslant} \mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x)\big(\eta(x) - \eta(x_i)\big)^2\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x)\big(\eta(x) - \eta(x_i)\big)^2 \mathbb{1}_{\|x_i - x\| \leqslant \varepsilon}\right] + \mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x)\big(\eta(x) - \eta(x_i)\big)^2 \mathbb{1}_{\|x_i - x\| \geqslant \varepsilon}\right] \\
&\leqslant \varepsilon + \mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x)\big(\eta(x) - \eta(x_i)\big)^2 \mathbb{1}_{\|x_i - x\| \geqslant \varepsilon}\right] \\
&\leqslant \varepsilon + \underbrace{\mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x) \mathbb{1}_{\|x_i - x\| \geqslant \varepsilon}\right]}_{\underset{n \to +\infty}{\longrightarrow} 0 \quad \text{from Assumption (b)}} .
\end{aligned}$$

Therefore (1) converges to 0 as $n \to \infty$. If η is not absolutely continuous, the result still holds using Assumption (a) but the proof is harder (see Devroye et al., 2013, p99).

(2) For the second term, using that $\mathbb{E}[\eta(x_i)] = y_i$, only the diagonal terms in the sum remain

$$\begin{aligned}
(2) &= \mathbb{E}\left[\left(\sum_{i=1}^{n} w_{n,i}(x)\big(y_i - \eta(x_i)\big)\right)^2\right] \\
&= \sum_{i=1}^{n}\sum_{j \neq i} w_{n,i}(x) w_{n,j}(x) \underbrace{\mathbb{E}\Big[\big(y_i - \eta(x_i)\big)\big(y_j - \eta(x_j)\big)\Big]}_{=0} + \mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x)^2 \big(y_i - \eta(x_i)\big)^2\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x)^2 \big(y_i - \eta(x_i)\big)^2\right] \\
&\leqslant \mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x)^2\right] \\
&\leqslant \mathbb{E}\left[\underbrace{\sum_{i=1}^{n} w_{n,i}(x)}_{=1} \max_{1 \leqslant j \leqslant n} w_{n,i}(x)\right] \\
&\leqslant \mathbb{E}\left[\max_{1 \leqslant j \leqslant n} w_{n,i}(x)\right] \underset{n \to +\infty}{\longrightarrow} 0 \quad \text{from Assumption (c)} .
\end{aligned}$$

□

Let us now conclude with the proof of the consistency of the k nearest neighbors when $k \to \infty$.

Proof of Theorem 5.8. First, we recall the definition of the weights $w_{n,i}(x)$ for the kNN classifier:

$$w_{n,i}(x) = \frac{\mathbb{1}_{x_i \in x_{(1)}(x), \dots, x_{(k)}(x)}}{k} = \begin{cases} \frac{1}{k} & \text{if } x_i \text{ belong to the } k \text{ nearest neighbors of } x \\ 0 & \text{otherwise} \end{cases} .$$

It suffices to show that they satisfy the three assumptions of Theorem 5.9 (Stone's theorem):

c) for all x, $\max_{1 \leqslant i \leqslant n} w_{n,i}(x) = \frac{1}{k(n)} \underset{n \to +\infty}{\longrightarrow} 0$ so that assumption (c) holds.

b) let $a > 0$, recall that $x_{(k)}(x)$ is the k-th nearest neighbor of x. We use that almost surely the distance of the k-nearest neighbor of x with x goes to zero when $k/n \to 0$:$\|x - x_{(k)}\| \underset{n \to +\infty}{\longrightarrow} 0$ when $\frac{k}{n} \to 0$ (see Devroye et al., 2013 for details). This yields $\mathbb{P}\{\|x - x_{(k)}(x)\| > a\} \to 0$ which entails

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x) \mathbb{1}_{\|x_i - x\| > a}\right] \\
&\qquad \leqslant \mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x) \mathbb{1}_{\|x_i - x\| > a} \mathbb{1}_{\|x_i - x_{(k)}(x)\| > a}\right] + \mathbb{E}\left[\sum_{i=1}^{n} w_{n,i}(x) \mathbb{1}_{\|x_i - x\| > a} \mathbb{1}_{\|x_i - x_{(k)}(x)\| < a}\right] \\
&\qquad \leqslant 0 + \mathbb{P}\big\{\|x_i - x_{(k)}(x)\| < a\big\} \to 0
\end{aligned}$$

a) Technical. See Devroye et al., 2013, Lemma 5.3.

□

Conclusion The k-nearest neighbors are universally consistent if $k \to \infty$ and $k/n \to 0$. Stone's theorem is actually more general and applies to other rules such as histograms.

6 High-dimensional data and variable selection

Learning objectives: understand the main concepts of the curse of dimensionality and how to deal with it. Understand why the Lasso regularization induces sparsity and how to compute the Lasso estimator with coordinate gradient descent.

In statistics or machine learning, we often want to explain some output $Y \in \mathcal{Y}$ from input $X \in \mathcal{X} \subset \mathbb{R}^p$ by observing a data set $D_n = \{(X_i, Y_i)\}_{1 \leqslant i \leqslant n}$ of i.i.d. observations. In previous lessons, we saw methods such as Ordinary Least Square Regression, K-Nearest Neighbors, Logistic Regression, and Probabilist models. Today, we would like to deal with high-dimensional input spaces, i.e., large p (possibly $p \gg n$). We will have two motivations in mind:

- *prediction accuracy*: when $p \gg n$ classical models fail. Is it possible to have strong theoretical guarantees on the risk (i.e., generalization error)?
- *model interpretability*: by removing irrelevant features X_i (i.e, by setting the corresponding coefficients estimates to zero), the model will easier to understand.

Good references on this topic are Giraud, 2014 and Friedman et al., 2001.

Why high-dimensional data? The volume of available data is growing exponentially fast nowadays. According to IBM in 2017, 10^{18} bytes of data were created every day in the world and 90% of data is less than two years old. Many modern data record simultaneously thousands up to millions of features on each objects or individuals. In many applications, data is high-dimensional such as with DNA, images, video, cookies (data about consumer preferences) or in astrophysics.

The curse of dimensionality

- High-dimensional spaces are vast and data points are isolated in their immensity.
- The accumulation of small errors in many different directions can produce a large global error.
- An event that is an accumulation of rare events may be not rare in high-dimensional space.

Example 6.1. *In high-dimensional spaces, no point in you data set will be close from a new input you want to predict. Assume that your input space is $\mathcal{X} = [0,1]^p$. The number of points needed to cover the space at a radius ε in L2 norm is of order $1/\varepsilon^p$ which increases exponentially with the dimension. Therefore, in high dimension, it is unlikely to have a point in you data set that will be close to any new input.*

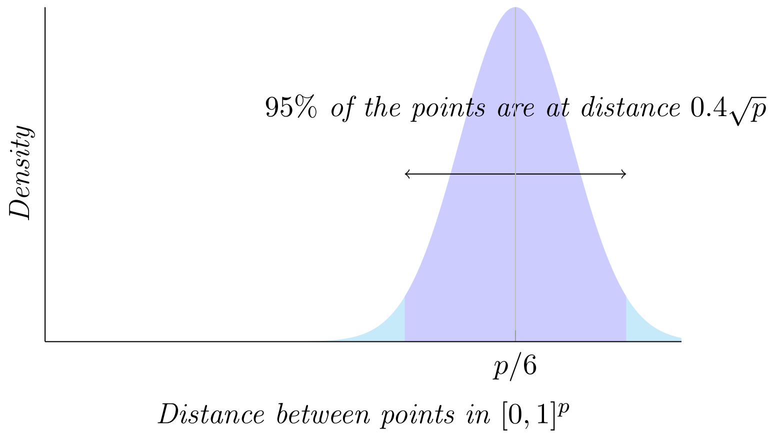
Example 6.2. *In high-dimensional spaces classical distances are often meaningless: all the points tends to be at similar distance from one another. Consider the following example to convince ourselves. Assume that X, X' follow uniform distribution on $[0,1]^p$. Then, the expected distance in square L2-norm between X and X' is*

$$\mathbb{E}\big[\|X - X'\|^2\big] = \sum_{i=1}^{p} \mathbb{E}\big[(X_i - X_i')^2\big] = p\mathbb{E}\big[(X_1 - X_1')^2\big] = p \int_0^1 \int_0^1 (x - x')dxdx' = \frac{p}{6}$$

Therefore, the average distance between the points increases with the dimension. Furthermore, the standard deviation of this square distance is

$$\sqrt{\mathrm{Var}\big(\|X - X'\|^2\big)} = \sqrt{\sum_{i=1}^{p} \mathrm{Var}\big((Xi - X_i')^2\big)} = \sqrt{p\mathrm{Var}\big((X_1 - X_1')^2\big)} = \frac{\sqrt{7p}}{6\sqrt{5}} \simeq 0.2\sqrt{p}\,.$$

Thus, if we plot the distribution of the square distance, we get something like:



Therefore, relatively to their distance, all points seem to be at similar distance from one another. The notion of nearest point distance vanishes. As a consequence, K-Nearest Neighbors gets poor performance in large dimension.

Example 6.3. *Let us consider another example in high-dimensional linear regression. We consider the ordinary least square estimator (OLS) for the linear model*

$$\widehat{\beta} \in \operatorname*{arg\,min}_{\beta\in\mathbb{R}^p} \big\|Y - X\beta\big\|^2 \qquad \textit{where} \quad Y_i = x_i^\top \beta^* + \varepsilon_i, \quad X = (x_1,\dots,x_n)^\top \in \mathbb{R}^{n\times p} \quad \textit{and} \quad \varepsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0,\sigma^2)\,.$$

If $rg(X) = p$ (i.e., $p \leqslant n$) then $\widehat{\beta} = (X \top X)^{-1} X^\top Y$ and as we saw in previous lecture the estimator satisfies

$$\mathbb{E}\big[\|\widehat{\beta} - \beta^*\|^2\big] = \mathrm{Tr}\big((X^\top X)^{-1}\big)\sigma^2\,.$$

In particular, in the very gentle case of an orthogonal design, we get $\mathbb{E}\big[\|\widehat{\beta} - \beta^*\|^2\big] = p\sigma^2$. Therefore, the variance of the estimator increases linearly with the dimension and the later gets unstable for high-dimensional data. Furthermore, OLS only works for $p \leqslant n$ because otherwise the matrix $X^\top X$ is not invertible and using pseudo-inverse would lead to highly unstable estimator and over-fitting. One needs to regularize.

The previous examples seem to show that the curse of dimensionality is unavoidable and we are doomed to poor estimators in large dimension. Hopefully, in many cases, data has an intrinsic low complexity (sparsity, low dimensional structure,...). This is the case of the data (for instance with images) or of the machine learning methods which is used (for instance Kernel regression).

What can we do with high-dimensional data? There are three classes of methods to deal with large dimensional input spaces:

- *Model selection*: we identify a subset of $s \ll p$ predictors that we believe to be related to the response. We then fit a model (for instance OLS) on the s variables only.
- *Regularization*: Ridge, Lasso,...
- *Dimension reduction*: the objective is to find a low-dimensional representation of the data. If we consider linear transformation, we may project the p predictors into a s-dimensional space with $s \ll p$. This is achieved by computing s different linear combination or projections of the variables. Then these projections are used as new features to fit a simple model for instance by least squares. Examples of such methods are PCA, PLS, ...

6.1 Model selection

The high level idea is to compare different statistical models corresponding to different possible hidden structure and select the best. This is theoretically very powerful, however the computational complexity is often prohibitive. Here, we will consider the example of the sparse linear model

$$Y = X\beta^* + \varepsilon\,, \quad Y = (y_1, \dots, y_n) \in \mathbb{R}^n, quadX \in \mathbb{R}^{n\times p}, \varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)\,. \tag{9}$$

We consider $p \gg n$ but we assume that β^* has only $s < p$ non-zero coordinates.

If we knew in advance the non-zero coordinates of β^* say $m^* \subset \{1, \dots, p\}$, we could consider the simpler linear regression problem $y_i = \sum_{j\in m^*} \beta^*_j X_{i,j} + \varepsilon_i$ and use the estimator

$$\widehat{\beta}_m \in \underset{\substack{\beta \in \mathbb{R}^p \\ \beta_j = 0 \forall j \notin m}}{\arg\min} \quad \big\|Y - X\beta\big\|^2 \tag{10}$$

for the correct choice $m = m^*$. More generally, this would work if we know that β belongs to some vectorial space of dimension $s < p$. We then get a risk which is scaling with s instead of p and the estimator has good statistical properties.

If we do not know m^* in advance, assuming the algorithmic complexity is not a problem, we can

1. consider a collection $\mathcal{M}$ of possible models $m \subset \{1, \dots, p\}$;
2. compute $\widehat{\beta}_m$ for each $m \in \mathcal{M}$ as defined in (10);
3. estimate β^* by the best estimator among the collection $\widehat{\beta}_m$.

A natural candidate for the best model is the minimizer of the empirical risk:

$$\widehat{\beta}_{\widehat{m}} \qquad \text{with} \quad \widehat{m} \in \underset{m\in\mathcal{M}}{\arg\min} \left\{ \big\|Y - X\widehat{\beta}_m\big\|^2 \right\}$$

The issue is that larger models $m \supset m'$ will always get smaller empirical risk because of overfitting. One needs to penalize models according to their complexity and choose the penalized estimator

$$\widehat{\beta}_{\widehat{m}} \qquad \text{with} \quad \widehat{m} \in \underset{m\in\mathcal{M}}{\arg\min} \left\{ \big\|Y - X\widehat{\beta}_m\big\|^2 + \mathrm{pen}(m) \right\} \tag{11}$$

There are several well known penalization criteria.

The Akaike Information Criterion (AIC) It defines the penalization

$$\mathrm{pen}(m) = 2|m|\sigma^2\,.$$

The AIC criterion is motivated by the following lemma.

Lemma 6.1. *In least square linear regression with Gaussian model (see* (9)*),* $\|Y - \widehat{X\beta}_m\|^2 + (2|m| - n)\sigma^2$ *is an unbiased estimator of the risk* $R(\widehat{\beta}_m) := \mathbb{E}\big[\|X\beta^* - X\widehat{\beta}_m\|^2\big]$.

Proof. We show that in least square regression the risk equals

$$R(\widehat{\beta}_m) := \mathbb{E}\big[\|X\beta^* - X\widehat{\beta}_m\|^2\big] = \mathbb{E}\big[\|Y - X\widehat{\beta}_m\|^2\big] + (2|m| - n)\sigma^2\,.$$

Let us first give some useful notation an equalities. For each $m \subset \{1, \dots, p\}$, we define the sub-vectorial space $S_m := \{X\beta \in \mathbb{R}^n : \beta \in \mathbb{R}^p, \beta_j = 0\ \forall j \notin m\}$ and $\Pi_{S_m} \in \mathbb{R}^{n\times n}$ the orthogonal projection matrix on S_m. Then, by definition of $\widehat{\beta}_m$, we have $X\widehat{\beta}_m = \Pi_{S_m} Y$ and we recall that $Y = X\beta^* + \varepsilon$. Furthermore, we will also use that:

$$\begin{aligned}\mathbb{E}\big[\|\Pi_{S_m}\varepsilon\|^2\big] = \mathbb{E}\big[\varepsilon^\top \Pi_{S_m}^\top \Pi_{S_m}\varepsilon\big] = \mathbb{E}\big[\varepsilon^\top \Pi_{S_m}\varepsilon\big] &= \mathbb{E}\big[\mathrm{Tr}(\varepsilon^\top \Pi_{S_m}\varepsilon)\big] \\ &= \mathbb{E}\big[\mathrm{Tr}(\Pi_{S_m}\varepsilon\varepsilon^\top)\big] = \sigma^2 \mathrm{Tr}(\Pi_{S_m}) = |m|\sigma^2\,. \end{aligned} \tag{12}$$

Similarly, $\mathbb{E}\big[\|(I-\Pi_{S_m})\varepsilon\|^2\big] = (n-|m|)\sigma^2$. From the decomposition $Y - X\widehat{\beta}_m = (I-\Pi_{S_m})(X\beta^* + \varepsilon)$, we have

$$\begin{aligned}
\mathbb{E}\big[\|Y - X\widehat{\beta}_m\|^2\big] &= \mathbb{E}\big[\|(I-\Pi_{S_m})X\beta^*\|^2 + \cancel{2\varepsilon^\top(I-\Pi_{S_m})X\beta^*} + \|(I-\Pi_{S_m})\varepsilon\|^2\big] \\
&= \|(I-\Pi_{S_m})X\beta^*\|^2 + (n-|m|)\sigma^2\,. \\
&= \|(I-\Pi_{S_m})X\beta^*\|^2 + \mathbb{E}\big[\|\Pi_{S_m}\varepsilon\|^2\big] + (n-2|m|)\sigma^2 \\
&= \mathbb{E}\big[\|(I-\Pi_{S_m})X\beta^* - \Pi_{S_m}\varepsilon\|^2\big] + (n-2|m|)\sigma^2 \qquad \leftarrow \text{Pythagore's theorem} \\
&= \mathbb{E}\big[\|X\beta^* - \Pi_{S_m}(X\beta^* + \varepsilon)\|^2\big] + (n-2|m|)\sigma^2 \\
&= \mathbb{E}\big[\|X\beta^* - X\widehat{\beta}_m\|^2\big] + (n-2|m|)\sigma^2\,.
\end{aligned}$$

□

Prior-based penalization Another popular penalization is to assign a prior weight π_m for each $m \in \mathcal{M}$, choose a regularization parameter $K > 1$ and select

$$\mathrm{pen}(m) = K\sigma^2\big(\sqrt{|m|} + \sqrt{2\log(1/\pi_m)}\big)^2\,. \tag{13}$$

Theorem 6.2 (Thm. 2.2, Giraud, 2014)**.** *Under the model 9, there exists some constant $C_K > 1$ depending only on K such that the penalized estimator $\widehat{\beta}_{\widehat{m}}$ defined in* (11) *with penalty* (13) *satisfies*

$$R(\widehat{\beta}_{\widehat{m}}) := \mathbb{E}\big[\|X\beta^* - X\widehat{\beta}_{\widehat{m}}\|^2\big] \leqslant C_K \min_{m\in\mathcal{M}} \Big\{\mathbb{E}\big[\|X\beta^* - X\widehat{\beta}_m\|^2\big] + \sigma^2 \log\frac{1}{\pi_m} + \sigma^2\Big\}\,.$$

A possible choice motivated by minimum description length (see lecture on PAC-Learning with infinite number of models) for the prior is $\log(1/\pi_m) \approx 2|m|\log p$, i.e., the number of bits needed to encode $m \subset \{1,\dots,p\}$. Remark that this choice of prior leads up to the $\log p$ to a similar criterion that for *AIC*. Yet, it is worth pointing out that the previous theorem is valid for general models $m \in \mathcal{M}$ (it is not restricted to the estimators (10)) and priors π_m. Other priors can promote different types of assumptions such as group sparsity.

Computational issues The estimator (11) has very nice statistical properties even when $p \gg n$. However we need to compute $\widehat{\beta}_m$ for all models $m \in \mathcal{M}$. This is often prohibitive. We can understand it by rewriting it as an optimization problem of the form

$$\widehat{\beta}_{\widehat{m}} \in \operatorname*{arg\,min}_{\beta\in\mathbb{R}^p} \Big\{\, \|Y - X\beta\|^2 + \lambda\|\beta\|_0 \,\Big\} \tag{14}$$

which is non-convex because of the $\|\cdot\|_0$. The estimator of AIC corresponds to the choice $\lambda = 2\sigma^2$. In some cases, such as orthogonal design, we can approximate efficiently the solution or find an efficient implementation. However, this is not true in general. A approximate implementation which is sometimes used to solve (11) is the *forward-backward algorithm.* It consists in alternatively trying to add or remove variables in the model one by one. It quickly converges in practice, but there is no theoretical guarantees.

6.2 The Lasso

The high-level idea of the Lasso is to transform the non-convex optimization problem (14) into a convex problem. This is done by replacing the ℓ_0-norm $\|\beta\|_0 = \sum_{j=1}^m \mathbb{1}_{\beta_j \neq 0}$ with the ℓ_1-norm $\|\beta\|_1 = \sum_{j=1}^p |\beta|_j$ which is convex. We define the LASSO estimator

$$\widehat{\beta}_\lambda \in \operatorname*{arg\,min}_{\beta \in \mathbb{R}^p} \Big\{ \|Y - X\beta\|^2 + \lambda \|\beta\|_1 \Big\} . \tag{LASSO}$$

The solution $\widehat{\beta}_\lambda$ may not be unique but the prediction $X\widehat{\beta}_\lambda$ is.

6.2.1 Geometric insight

By convex duality, the Lasso is also the solution of

$$\widehat{\beta}_\lambda \in \operatorname*{arg\,min}_{\beta \in \mathbb{R}^p : \|\beta\|_1 \leqslant R_\lambda} \Big\{ \|Y - X\beta\|^2 \Big\} ,$$

for some radius $R_\lambda > 0$. The non-smoothness of the ℓ_1-norm puts some coefficients to zero. In Figure 8, we can see that because of the corners of the ℓ_1-ball, the solution $\widehat{\beta}_\lambda$ gets zero coefficients which is not the case when regularizing with the ℓ_2-norm (on the right).

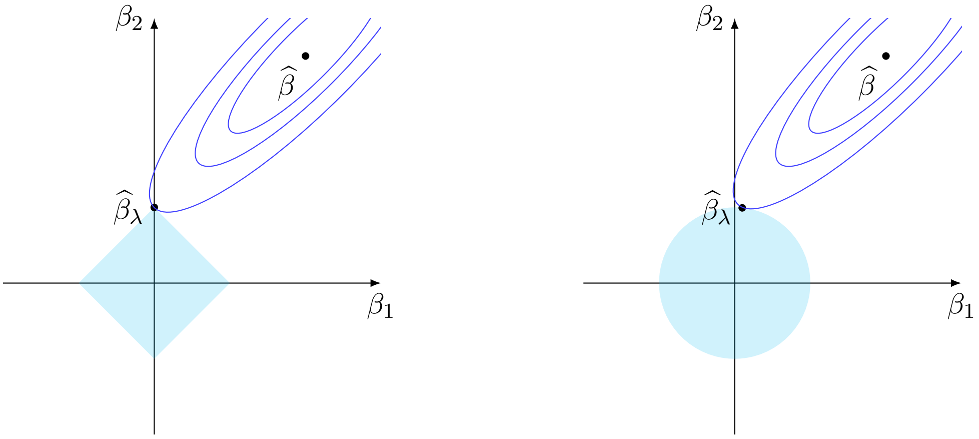


Figure 8: $\widehat{\beta}$ denotes the minimizer of the empirical risk and the blue lines denote level lines of the empirical risk [left] Regularization with a ℓ_1-ball [right] Regularization with a ℓ_2-ball.

6.2.2 What does the solution of the Lasso looks like?

To solve the problem of Lasso, if the objective function $\mathcal{L} : \beta \mapsto \|Y - X\beta\|^2 + \lambda\|\beta\|_1$ was differentiable, one would cancel the gradient. However, because of the ℓ_1-norm the latter is not differentiable and one needs to generalize the notion of gradient to convex functions which are not necessarily differentiable. This is done with the following definition.

Definition 6.1 (Subdifferential). *A subgradient of a convex function $f : \mathbb{R}^p \to \mathbb{R}$ at a point $\beta_0 \in \mathbb{R}^p$ is a vector $z \in \mathbb{R}^p$ such that for any $\beta \in \mathbb{R}^p$ the convex inequality holds*

$$f(\beta) - f(\beta_0) \geqslant z^\top (\beta - \beta_0) .$$

The set of all subgradients of f at β_0 is denoted $\partial f(\beta_0)$ and is called the subdifferential of f at β_0.

The subdifferential of the ℓ_1-norm is

$$\partial\|\beta\|_1 = \Big\{ z \in [-1,1]^p \text{ s.t. for all } 1 \leqslant j \leqslant p \quad z_j = \operatorname{sign}(\beta_j) \text{ if } \beta_j \neq 0 \Big\}$$

and the subdifferential of the objective funtion of the Lasso is

$$\partial \mathcal{L}(\beta) = \big\{ -2X^\top (Y - X\beta) + \lambda z : \ z \in \partial\|\beta\|_1 \big\}\,.$$

Any solution of the Lasso should cancel the subdifferential. Therefore, if $\widehat{\beta}_\lambda$ is a solution of the Lasso, it exists $\widehat{z} \in \partial\|\widehat{\beta}_\lambda\|_1$ (i.e., $\widehat{z}_j = \operatorname{sign}(\widehat{\beta}_\lambda(j))$ if $\widehat{\beta}_\lambda(j) \neq 0$ and $\widehat{z}_j \in [-1,1]$ otherwise) such that

$$-2X^\top (Y - X\beta) + \lambda \widehat{z} = 0 \quad \Rightarrow \quad X^\top X \widehat{\beta}_\lambda = X^\top Y - \frac{\lambda}{2}\widehat{z}\,. \tag{15}$$

If the gram matrix $X^\top X$ is general, it is not possible to solve the later in close form. To get some insights about the solution of the Lasso, let us assume the orthonormal setting $X^\top X = I_p$. Then, from (15), we get for all $j \in \{1, \dots, p\}$ such that $\widehat{\beta}_\lambda(j) \neq 0$

$$\widehat{\beta}_\lambda(j) = X_j^\top Y - \frac{\lambda}{2}\operatorname{sign}(\widehat{\beta}_\lambda(j))\,.$$

Therefore, $X_j^\top Y = \widehat{\beta}_\lambda(j) + \operatorname{sign}(\widehat{\beta}_\lambda(j))$ and $\widehat{\beta}_\lambda(j)$ have same sign and we obtain for all $1 \leqslant j \leqslant p$

$$\widehat{\beta}_\lambda(j) = \begin{cases} X_j^\top Y - \frac{\lambda}{2}\operatorname{sign}(X_j^\top Y) & \text{if } |X_j^\top Y| \geqslant \frac{\lambda}{2} \\ 0 & \text{if } |X_j^\top Y| \leqslant \frac{\lambda}{2} \end{cases}$$

In the orthonormal setting, the Lasso performs thus a soft threshold of the coordinates of the OLS.

Statistical property of the Lasso estimator For λ large enough $\lambda \simeq \sigma\sqrt{\log p}$, under some additional condition on the design (relaxed version of orthonormal design), it is possible to show that the Lasso does not assign any weight to coefficients that are not in m^*. If λ is properly chosen, it recovers exactly the coefficients of β^* and its risk is controlled with high probability as

$$R(\widehat{\beta}_\lambda) = \big\|X\beta^* - X\widehat{\beta}_\lambda\big\|^2 \leqslant \inf_{\beta \in \mathbb{R}^p \setminus \{0\}} \Big\{ \|X\beta - X\beta^*\|^2 + \square_X \lambda^2 \|\beta\|_0 \Big\}\,,$$

where $\lambda^2 \simeq \sigma^2 \log p$ and $\square_X$ is the compatibility constant depending on the design X. It can be bad for non-orthogonal design. We recover a similar result than the one obtained for model selection in Theorem 6.2 but with $\square_X$ and with an efficient procedure. It can be shown that it is not possible to avoid $\square_X$ for efficient (polynomial time) procedures.

6.2.3 Computing the Lasso estimator

Since this is the solution of a convex optimization problem, the solution of the Lasso can be obtained efficiently. There are three main algorithms used by the community.

Coordinate descent The idea is to repeatedly minimize the objective function $\mathcal{L}(\beta)$ with respect to each coordinate. It converges thanks to the convexity of $\mathcal{L}$. As we saw in Equation (15), the solution of the Lasso satisfies

$$X^\top X \widehat{\beta}_\lambda = X^\top Y - \lambda \widehat{z}$$

where $\widehat{z} \in \partial \|\beta\|_1$. We saw that the solution equals $\widehat{\beta}_\lambda(j) = S_\lambda(X_j^\top Y)$ when $X^\top X = I_p$. This equation has however no closed-form solution in general. The idea of coordinate descent is to solve this equation only for one coordinate, fixing all the other coordinates.

Let $1 \leqslant i \leqslant n$ and fix coordinates $\beta_j \in \mathbb{R}$ for $j \neq i$. Solving the i-th coordinate optimisation problem given by

$$\min_{\beta_i} \mathcal{L}(\beta) = \min_{\beta_i \in \mathbb{R}} \left\{ \frac{1}{2} \|Y - X\beta\|^2 + \lambda \|\beta\|_1 \right\} ,$$

we get that the i-th partial sub-derivative of $\mathcal{L}$ should cancel, which gives similarly to previously

$$X_i^\top X \beta = X_i^\top Y - \lambda z_i \,,$$

where $z_i \in \partial|\beta_i|$. This can be rewritten as

$$X_i^\top X_i \beta_i + X_i^\top X_{-i} \beta_{-i} = X_i^\top Y - \lambda z_i \,,$$

where $X_{-i} \in \mathbb{R}^n \times (p-1)$ is the input matrix without column i and $\beta_{-i} \in \mathbb{R}^{p-1}$ is the fixed parameter vector without coordinate i.

Assume $\beta_i \neq 0$, then $z_i = \operatorname{sign}(\beta_i)$ and

$$X_i^\top X_i \beta_i + \lambda \operatorname{sign}(\beta_i) = X_i^\top (Y - X_{-i}\beta_{-i}) \,,$$

Since $X_i^\top X_i > 0$, we have $z_i = \operatorname{sign}\big(X_i^\top (Y - X_i^\top X_{-i}\beta_{-i})\big)$ which implies

$$\beta_i = \frac{S_\lambda\big(X_i^\top (Y - X_i^\top X_{-i}\beta_{-i})\big)}{X_i^\top X_i} . \tag{16}$$

where S_λ is the soft-threshold function:

$$S_\lambda(x) = \begin{cases} 0 & \text{if} \quad |x| \leqslant \lambda \\ x - \lambda \operatorname{sign}(x) & \text{otherwise} \end{cases} .$$

The algorithm of coordinate descent consists in sequentially repeating the update (16) for $i = 1, \dots, p, 1 \dots, p, \dots$ minmizing the objective function with respect to each coordinate at a time.

Fista (fast iterative shrinkage thresholding algorithmn) It uses the explicit formula in the orthogonal design setting for computing recursively an approximation of the solution

LARS The insight of the algorithm comes from equation (15): $X^\top X \widehat{\beta}_\lambda = X^\top Y - \frac{\lambda}{2}\widehat{z}$. We then consider the function $\lambda \mapsto \widehat{\beta}_\lambda$. For non-zero coefficients, $\widehat{z}_j = \operatorname{sign}(\widehat{\beta}_\lambda(j))$ and is constant while $\lambda \mapsto \widehat{\beta}_\lambda(j)$ does not change sign. Therefore, the function $\lambda \mapsto \widehat{\beta}_\lambda$ is piecewise linear in λ with a change when for some coordinate $\widehat{\beta}_\lambda(j)$ changes sign. LARS computes the sequence $\{\widehat{\beta}_{\lambda_1}, \widehat{\beta}_{\lambda_2}, \dots\}$ of the Lasso estimator corresponding to the break points of the path $\lambda \mapsto \widehat{\beta}_\lambda$. At each break point, the model $m_\lambda = \{i \in \{1, \dots, p\} : \widehat{\beta}_\lambda(i) \neq 0\}$ is updated and we solve the linear equation

$$X_{m_\lambda}^\top X_{m_\lambda} \widehat{\beta}_\lambda(m_\lambda) = X_{m_\lambda}^\top Y - \frac{\lambda}{2} \operatorname{sign}(\widehat{\beta}_\lambda(m)) \,,$$

until the next break point. This algorithm is slower than the other two algorithms but it provides the full regularization path $\lambda \mapsto \widehat{\beta}_\lambda$ (see Figure 9).

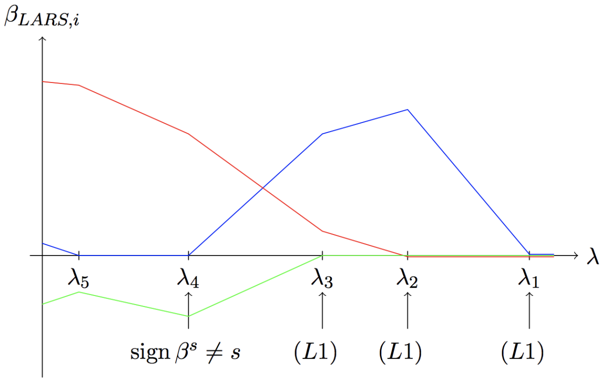


Figure 9: Lasso regularization path computed with LARS

6.2.4 Final remarks and variants

Removing the bias of the Lasso The Lasso estimator $\widehat{\beta}_\lambda$ is biased. Often one might want to remove the bias for instance by first computing $\widehat{\beta}_\lambda$ to select to good model $\widehat{m}_\lambda$ and then solve the OLS or Ridge on the model $\widehat{m}_\lambda$ only.

No penalization of the intercept In practice, the intercept is often no penalized and the Lasso solves

$$\widehat{\beta}_\lambda \in \operatorname*{arg\,min}_{\beta \in \mathbb{R}^p} \left\{ \sum_{i=1}^{n} (Y_i - \beta_0 - \beta^\top X_i)^2 + \lambda \|\beta\|_1 \right\}.$$

Group Lasso It is an extension when coordinates are sparse by groups. In other words, we have some groups $G_k \subset \{1, \dots, p\}$ and we assume that all coordinates β_i for $i \in G_k$ are either all zero or all non-zero.

Elastic net It is a mix of ℓ_1 and ℓ_2 regularization

$$\widehat{\beta} \in \operatorname*{arg\,min}_{\beta \in \mathbb{R}^p} \left\{ \|Y - X\beta\|^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2 \right\}.$$

It also selects variables thanks to sharp corners and it is heavily used in practice.

Calibration of λ It is a crucial point in practice. A common solution is to perform K-fold cross validation. There are a few other techniques such as the slopes heuristic.

7 Multilayer Perceptron

Learning objectives: understand the main concepts of multilayer perceptron, given the weights and biases for a neural net, be able to compute its output from its input. Approximation properties of shallow networks.

Neural networks are a particular approach to machine learning, inspired by the way the brain processes information. A neural network is composed of a large number of units, each of which performs very simple operations, but produces sophisticated behaviors as a whole. Neural networks are increasingly used in many applications. They are the basis for speech recognition, translation, search result ranking, face recognition, sentiment analysis, image retrieval and many other applications. There are powerful software packages such as Caffe, Theano, Torch, and TensorFlow, which allow us to quickly implement sophisticated learning algorithms.

7.1 Single processing unit

In biology, the neuron is the basic processing unit of the brain. It has a large branching tree of dendrites, which receive chemical signals from other neurons at junctions called synapses, and convert them into electrical signals. In machine learning, we eliminate most of the complexity and use a simplified model of a neuron, shown in Figure 10. This neuron has a set of incoming connections from other neurons, each with an associated strength or weight. It computes a value, called pre-activation, which is the sum of the incoming signals x_j multiplied by their weights w_j: $b + \langle w, x\rangle$. An additional bias (or intercept) b determines the activation of the neuron in the absence of inputs. The neuron then applies an activation function ϕ to form its output:

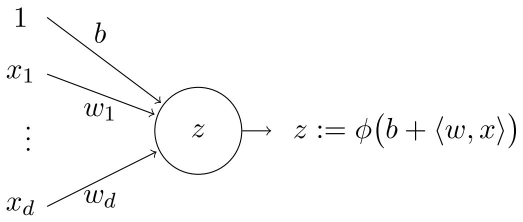


Figure 10: Single processing unit and its components. The activation function is denoted by ϕ and the output z. The vector $x = (x_1, \dots, x_d)$ represents the inputs from other units within the network; b is called bias and represents an external input to the unit.

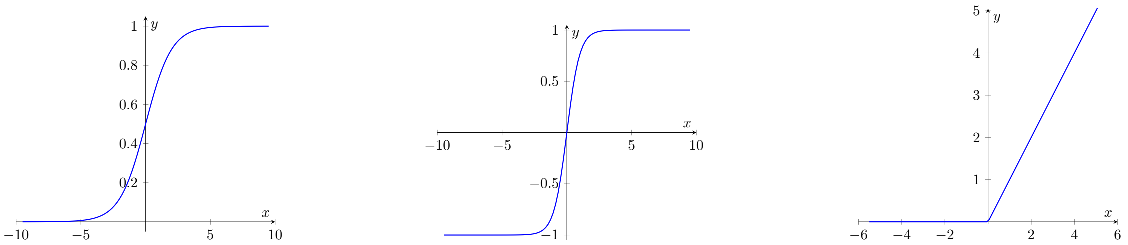
$$z = \phi(b + \langle w, x\rangle)\,.$$

Examples of activation functions include:

- the identity function $\phi(x) = x$. This is used for regression $y \in \mathbb{R}$ and corresponds to performing *linear regression*;
- the threshold function $\phi(x) = \mathbb{1}\{x \geqslant 0\}$ or the sign $\phi(x) = \text{sign}(x)$. This is used for binary classification $y \in \{0, 1\}$ or $y \in \{-1, 1\}$. It corresponds to binary linear classifier or perceptron;
- the logistic sigmoid $\phi(x) = (1 + e^{-x})^{-1}$. This is used for binary classification and corresponds to logistic regression;
- the linear rectification (ReLu) $\phi(x) = x\mathbb{1}\{x \geqslant 0\}$;
- the hyperbolic tangent function $\phi(x) = (e^x - e^{-x})/(e^x + e^{-x})$ for binary classification.
- ramp functions,. . .

The Perceptron algorithm Consider a binary classification problem with input $x \in \mathbb{R}^d$ and ouptut $y \in \{-1, 1\}$. Perceptron will make predictions of the form:

$$z = \text{sign}(b + \langle w, x\rangle)$$



(a) Sigmoide (b) Hyperbolic tangent (c) ReLU

Figure 11: Three common activation functions for neural networks.

which corresponds to the non-linear activation function $\phi(x) = \text{sign}(x)$. While logistic regression provides probabilitic outputs, perceptron only provides outputs in $\{-1, 1\}$. A prediction z for a data (x, y) makes an error if $z \neq y$, which is equivalent to $zy < 0$. One may thus consider the loss function

$$\ell(y, z) = -yz\,\mathbb{1}\{yz < 0\}$$

which equals 1 in case of error and 0 otherwise. Similarly, Perceptron considers the loss

$$\ell\big(y, (b, w)\big) = -y(b + \langle w, x\rangle)\mathbb{1}\{y(b + \langle w, x\rangle) < 0\}\,.$$

The partial derivatives in b and w are 0 if the data is correctly classified and

$$\nabla_b \ell(y, (b, w)) = -y \quad \text{and} \quad \nabla_w \ell(y, (b, w)) = -yx\,,$$

for misclassified data. The Perceptron algorithm learns the parameters (b, w) by applying stochastic gradient descent (without resampling) to the training data with that loss. The algorithm is described in Algorithm 1.

Algorithm 1 Perceptron

```
Input:  $\gamma \in (0, 1)$                                ▷ Learning rate
Init:  $b_1 \leftarrow 0$ and $w_1 \leftarrow 0$
for $i = 1, \dots, n$ do
    $z_i \leftarrow b_i + w_i^\top x_i$                     ▷ Prediction of data $i$
    if $z_i y_i < 0$ then                                  ▷ if the data point is misclassified
        $b_{i+1} \leftarrow b_i + \gamma y_i$
        $w_{i+1} = w_i + \gamma y_i x_i$
    else                                                   ▷ if the data is well classified
        $b_{i+1} \leftarrow b_i$
        $w_{i+1} \leftarrow w_i$
    end if
end for
Return: $(b_{n+1}, w_{n+1})$
```

7.2 Multilayer perceptron

A neural network is a combination of several of these units. Each of them is very simple, but together they can approximate complex functions. Here, we focus on feedforward neural networks, in which the units are organized in a graph without any cycles, so that all computations

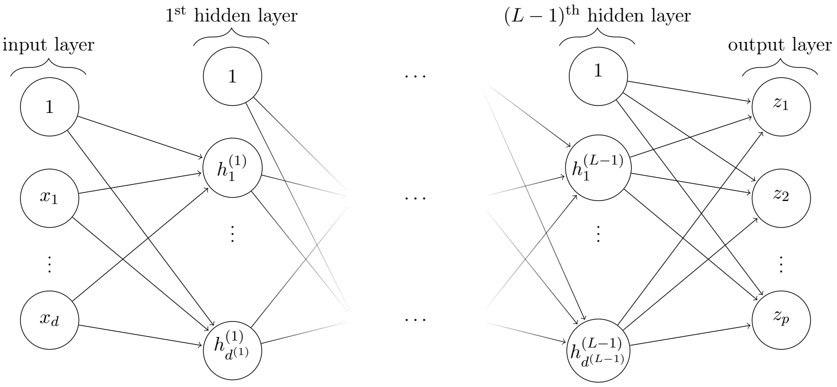


Figure 12: Network graph of a L-layer perceptron with d input units and p output units. The k^{th} hidden layer contains $d^{(k)}$ hidden units.

can be performed sequentially. In contrast, in recurrent neural networks, the graph can have cycles. The most basic type of feedforward network is the multilayer perceptron (MLP), shown in Figure 12. Here, the units are organized into a set of layers. Each unit in one layer is connected to each unit in the next layer; the network is said to be fully connected. The first layer is the input layer, and its units take the values of the input features. The last layer is the output layer, and it has one unit for each value that the network produces (i.e., a single unit in the case of regression or binary classification, or p units in the case of p-class classification). All intermediate layers are called hidden layers, because we do not know in advance what these units are to compute, and this must be discovered during learning. The number of layers is referred as the depth and the number of neurons within a layer as the width. Deep Learning refers to neural nets with many hidden layers.

Denote by

- L the number of layers (the depth of the network); There are thus $L-1$ hidden layers and the output layer correponds to the L-th layer of the network;
- $x \in \mathbb{R}^d$ the input vector;
- $z \in \mathbb{R}^p$ the final output.
- $d^{(k)}$ the dimension (number of units) of the k-th layer. Then, $d^{(0)} = d$ and $d^{(L)} = p$;
- $h^{(k)} \in \mathbb{R}^{d^{(k)}}$ the output of the k-th hidden layer;
- $W^{(k)} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$ the weight matrix where each row is the weight vector of a unit of the k-th layer;
- $b^{(k)} \in \mathbb{R}^{d^{(k)}}$ the bias vector of the k-th layer;
- $\phi^{(k)}$ the activation function of the k-th layer; Note that activation functions may vary from one layer to another;

The MLP computations may be written as:

$$\begin{aligned} h^{(1)} &= \phi^{(1)}\big(W^{(1)}x + b^{(1)}\big) \\ h^{(k)} &= \phi^{(k)}\big(W^{(k)}h^{(k-1)} + b^{(k)}\big) \qquad \forall k = 2, \dots, L-1 \\ z &= \phi^{(L)}\big(W^{(L)}h^{(L-1)} + b^{(L)}\big), \end{aligned} \tag{17}$$

where by abuse of notation $\phi(u)$ for a vector $u \in \mathbb{R}^d$ denotes ϕ applied to each component of u, i.e., $(\phi(u)_i = \phi(u_i)$. Note that the parameters of the network are all weight matrices $W^{(k)}$ and bias $b^{(k)}$, for $k = 1, \dots, L$. A MLP has thus $m := \sum_{k=1}^{L} d^{(k)}(d^{(k-1)} + 1)$ parameters: each unit of the k-th layer has one bias parameter and $d^{(k-1)}$ weights.

Matrix notation When learning the parameters of a network, one has access to learning data $\{(x_i, y_i)\}_{1 \leqslant i \leqslant n}$. Similarly to linear regression, it is useful to write the above equations in matrix form. Denote by $X = (x_1, \dots, x_n)^\top \in \mathbb{R}^{n \times d}$ the input matrix, by $H^{(k)} \in \mathbb{R}^{n \times d^{(k)}}$ the outputs of the k-th layer for all training data points, and $Z \in \mathbb{R}^{n \times p}$ the matrix of outputs, we can write

$$\begin{aligned} H^{(1)} &= \phi^{(1)}\big(XW^{(1)\top} + \mathbf{1}b^{(1)\top}\big) \\ H^{(k)} &= \phi^{(k)}\big(H^{(k-1)}W^{(k)\top} + \mathbf{1}b^{(k)\top}\big) \qquad \forall k = 2, \dots, L-1 \\ Z &= \phi^{(L)}\big(H^{(L-1)}W^{(L)\top} + \mathbf{1}b^{(L)\top}\big)\,. \end{aligned} \tag{18}$$

These equations can be directly transposed to python librairies to efficiently compute predictions over the whole dataset simultaneously. Note that it is hard to remember where the transpose $\top$ are. To write it properly, the best solution is to pay attention to the dimensions.

Learning MLP As we did for supervised learning, we must first choose a loss function to evaluate performance. For regression, we can use the squared loss $\ell(y, z) = (y - z)^2$, while for binary classification, we can use the logistic loss. Then, denote by $\theta \in \mathbb{R}^m$ the vector of parameters of the network (i.e., that contains all $W_{ij}^{(k)}$ and $b_i^{(k)}$) and by $f_\theta(x) \in \mathbb{R}$ the prediction z obtained by applying the network computations (17) to the input x. Then, we can write the empirical risk of the network:

$$\widehat{\mathcal{R}}(\theta) = \frac{1}{n} \sum_{i=1}^n \ell\big(y_i, f_\theta(x_i)\big)\,.$$

The parameter vector θ is the trained through empirical risk minimization by seeking for $\widehat{\theta} \in \arg\min_{\theta \in \mathbb{R}^m} \widehat{\mathcal{R}}(\theta)$. This is generally approximately solved by applying a (stochastic) gradient descent type algorithm

$$\widehat{\theta}_{i+1} \leftarrow \widehat{\theta}_i - \gamma \widehat{g}_i\,,$$

where $\widehat{g}_i$ is an estimation of $\nabla \widehat{\mathcal{R}}(\widehat{\theta}_i)$. The parameter θ_0 is usually initialized with small random values distributed around zero.

Of course, many interesting aspects, which we will not have time to address in details in this class, arise from this optimization problem. We list some below.

- In deep learning, the dimension m may be very large and computing full gradients directly is prohibitive. This problem is somewhat solved by *back-propagation* that cleverly uses chain rules to efficiently compute partial derivatives and propagate errors through the network.
- Large parameter sizes m can also cause over-parametrisation (when $m > n$) and hence overfitting problems. Therefore, similarly to linear regression, one needs to use regularization (ℓ_2-penalty, dropout, early stopping,...).
- The empirical risk $\widehat{\mathcal{R}}$ is generally non-convex in θ. Gradient descent has thus no theoretical guarantee to converge to a global minimum and often yield to local minima.

7.3 Universal approximation properties

In previous lectures, we saw how to represent complex non-linear functions by using features. For example, linear regression can represent a cubic polynomial if we use the feature map $\phi(x) = (1, x, x^2)$. Yet, this is not fully satisfactory because:

- The features must be specified in advance, which may require a lot of engineering work.

- It may require a very large number of features to represent a certain set of functions; for example, the feature representation for cubic polynomials is cubic in the number of input features.

Kernel methods that you will see latter in this class partially solve these issues by providing rich feature representations of the inputs.

In contrast, Multilayer Perceptron, or neural nets in general, do not require any feature transformation of the inputs and take a different approach to model complex functions. MLP connects many simple units into a network and together these units can compute surprisingly complex functions, when using non-linear activation functions. As it turns out, even a shallow MLP (i.e., with one single hidden layer) can approximate any continous function when given appropriate weights. The site http://playground.tensorflow.org/ provides a beautiful interactive visualization of neural networks that allows to see the role and power of hidden units.

Theorem 7.1 (Universal approximation theorem, Cybenko'89, Hornik'89)**.** *Any continuous fonction on a compact can be approximated arbirarily well by a Multilayer Perceptron with one hidden layer and large enough width, as soon as the activation function is not polynomial.*

Note that the same theorem holds true for greater depth (by taking the same network with one hidden layer and adding other layers that approximate the identity function). More recent approximation theorems also hold for fixed width when making the depth goes to infinity.

⚠ This approximation result is a nice property but there are a few caveats.

- First, the network can be arbitrarily large. In practice, for computational and statistical purposes, compact networks (i.e., with few parameters) are desirable. This need of compactness explains partially the success of deep learning (otherwise why would we need deep networks if shallow MLP are enough): deep networks are often much more compact with the same approximation power.
- Second, the weights of the network can be arbitrarily large, which can lead to large variance.
- Finally, the approximation theorem provides only the existence but not the exact values of the weights and the training of neural networks can be sophisticated with non-convex objective functions.

Example 7.1. *Let us prove universality in the case of binary inputs: $\mathcal{X} = \{-1, 1\}^d$ and arbitrary output space $\mathcal{Y}$. Let $f : \mathcal{X} \to \mathcal{Y}$ and let us design a shallow network that equals f. We consider the hard-threshold $\phi(x) = \mathbb{1}\{x \geqslant 0\}$ as activation function for all hidden units, and the identity function for the output unit. For any $x \in \mathcal{X}$, we associate one hidden unit with the weight vector $w_x^{(1)} = x$ and bias $b_x = d - 1/2$. Then, for any $x' \in \mathcal{X}$, we have for this unit*

$$w_x^\top x' + b_x \geqslant 0 \qquad \textit{iff.} \qquad x' = x\,,$$

which yields,

$$h_x^{(1)} = \phi(w_x^{(1)\top} x' + b_x) = \begin{cases} 1 & \textit{if } x' = x \\ 0 & \textit{otherwise} \end{cases}$$

The highlevel idea is that each hidden unit is asked to identify one pattern in $\{-1, 1\}^d$. Then, it suffices to assign the weight $w_x^{(2)} = f(x)$ to relate that hidden unit to the output unit. The final prediction of the is then for any $x' \in \mathcal{X}$

$$\sum_{x \in \mathcal{X}} w_x^{(1)} \phi(w_x^\top x' + b_x) = \sum_{x \in \mathcal{X}} f(x) \mathbb{1}\{x = x'\} = f(x')\,.$$

Therefore, the network exactly matches the function f. Note that the depth of the network is $L = 2$ and the width is $|\mathcal{X}| = 2^d$.

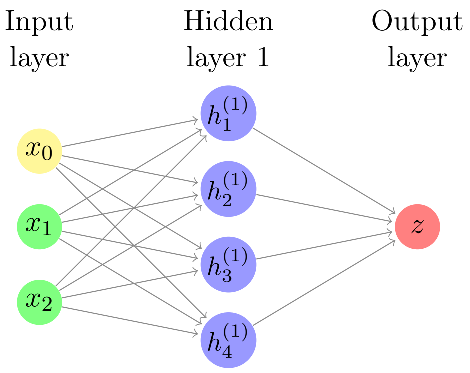


Figure 13: Architecture of a multilayer perceptron with one hidden layer that may approximate any function with inputs in $\{-1, 1\}^2$.

References

Bach, F. (2022). “Learning Theory from First Principles”.

Charalambos, D. A. and K. C. Border (2006). *Infinite dimensional analysis: a hitchhiker’s guide.* Springer.

Cornillon, P.-A. and E. Matzner-Løber (2011). “La régression linéaire simple”. In: *Régression avec R*, pp. 1–28.

Cover, T. and P. Hart (1967). “Nearest neighbor pattern classification”. In: *IEEE transactions on information theory* 13.1, pp. 21–27.

Devroye, L., L. Györfi, and G. Lugosi (2013). *A probabilistic theory of pattern recognition.* Vol. 31. Springer Science & Business Media.

Friedman, J., T. Hastie, and R. Tibshirani (2001). *The elements of statistical learning.* Vol. 1. 10. Springer series in statistics New York, NY, USA:

Giraud, C. (2014). *Introduction to high-dimensional statistics.* Chapman and Hall/CRC.

Hastie, T. J., R. J. Tibshirani, and J. H. Friedman (2001). *The Elements of Statistical Learning.* Springer Series in Statistics. New York, NY, USA: Springer New York Inc.

Rivoirard, V. and G. Stoltz (2012). *Statistique mathématique en action.*

Stone, C. J. (1977). “Consistent nonparametric regression”. In: *The annals of statistics*, pp. 595–620.