Chapter 15

Credit Scoring Models

Credit scoring refers to statistical models to measure the creditworthiness of a person or a company. They have progressively replaced judgemental systems and are now widely used by financial and banking institutions that check the credit rating and capacity of the borrower before to approve a loan. Therefore, credit scoring is at the heart of the decision-making system for granting credit. This is particularly true for consumer credit (mortgage, credit card, personal loan, etc.). Credit scoring models are also used for commercial firms, but their final outputs are generally not sufficient for making a decision. For instance, they can be completed with the knowledge of the relationship manager on the company.

Credit scoring first emerged in the United States. For instance, one of the oldest credit scores is the FICO score that was introduced in 1989 by Fair Isaac Corporation. The FICO score is based on consumer credit files of consumer credit reporting agencies such as Experian, Equifax and TransUnion. It remains today the best-known and most-used external scoring system in the world. In thirty years, credit scoring models have evolved considerably, and financial institutions have generally built their own internal credit scoring system. In particular, the development of credit scoring techniques has speeded up in the 2000s with the introduction of the IRB formula in the Basel II Accord. For instance, they are now used for estimating the probability of default or the loss given default, while validation and backtesting procedures are better defined. The estimation of credit scores has also benefitted from the massive development of marketing scores, big data and machine learning.

15.1 The method of scoring

15.1.1 The emergence of credit scoring

15.1.1.1 Judgmental credit systems versus credit scoring systems

The underlying idea of credit valuation is to use the experience in order to approve or deny the credit of a (new) customer. In the case of judgmental credit analysis, the decision is made by a credit analyst or the relationship manager, and is based on the character, the capacity and the capital of the borrower. Past experience of the credit analyst is then fundamental, and two credit analysts may give two different answers. Moreover, it takes many years to build a track record, because it is not an industrial process. Indeed, the credit analyst can analyze only a limited number of requests per week. Because of the high costs, financial institutions have sought to automate credit decisions.

In 1941, Durand presented a statistical analysis of credit valuation. He showed that credit analysts uses similar factors, and proposed a credit rating formula based on nine factors: (1) age, (2) sex, (3) stability of residence, (4) occupation, (5) industry, (6) stability of employment, (7) bank account, (8) real estate and (9) life insurance. The score is additive and can take values between 0 and 3.46. For instance, 0.40 is added to the score if the applicant is a woman, 0.30 if the applicant is 50 years old or more, etc. Durand's formula is the first credit scoring model that has been published. Such credit scoring models become more and more popular in financial institutions in the 1950s and 1960s, but the real turning point is the development of the credit card business in the 1970s (Thomas, 2000). From an industrial point of view, a credit scoring system has two main advantages compared to a judgmental credit system:

1. it is cost efficient, and can treat a huge number of applicants;
2. decision-making process is rapid and consistent across customers.

Generally, financial institutions also consider that credit scoring systems are more efficient and successful than judgmental credit systems. However, comparing track records is always a difficult exercise since it depends on many factors. Some credit analysts may have a very good track record, while the live performance of some statistical credit models may be worse than their backtest performance. Nevertheless, the case of credit cards has demonstrated that credit scoring models are far better than judgmental credit systems. The main reason is the large amount of data that can be analyzed by a statistical model. While experience is essential for a credit analyst, the efficiency of credit scoring depends on the quality and amount of data.

15.1.1.2 Scoring models for corporate bankruptcy

These models appear with the research of Tamari (1966), who proposed to combine several financial ratios for assessing the financial health of corporate firms. Nevertheless, the weight of each ratio was assumed to be fixed and has been arbitrary calibrated. The empirical work of Beaver (1966) was more interesting since he estimated the univariate statistical relationship between financial ratios and the failure. However, the seminal paper for the evaluation of creditworthiness is the publication of Altman (1968). Using a small dataset and the statistical method of discriminant analysis, he introduced the concept of z-score for predicting bankruptcy of commercial firms. The score was equal to:

$$Z = 1.2 \cdot X_1 + 1.4 \cdot X_2 + 3.3 \cdot X_3 + 0.6 \cdot X_4 + 1.0 \cdot X_5$$

where the variables X_j represent the following financial ratios:

X_j	Ratio
X_1	Working capital / Total assets
X_2	Retained earnings / Total assets
X_3	Earnings before interest and tax / Total assets
X_4	Market value of equity / Total liabilities
X_5	Sales / Total assets

If we note Z_i the score of the firm i, we can calculate the normalized score $Z_i^\star = (Z_i - m_z)/\sigma_z$ where m_z and σ_z are the mean and standard deviation of the observed scores. $Z_i^\star$ can then be compared to the quantiles of the Gaussian distribution or the empirical distribution. A low value of $Z_i^\star$ (for instance $Z_i^\star < 2.5$) indicates that the firm has a high probability of default. Today, the technique of z-score, which consists of normalizing a score, is very popular and may be found in many fields of economics, finance, marketing and statistics.

15.1.1.3 New developments

Since the publication of Durand (1941) and Altman (1968), the research on credit scoring can be split into three main categories:

- The first category concerns the default of corporate firms. It appears that the choice of financial ratios and relevant metrics as explanatory variables are more important than the model itself (Hand, 2006). Other factors such as the business cycle, economic conditions or market prices (Hillegeist *et al.*, 2004) may be taken into account. Moreover, the one-size-fits-all approach is not appropriate and credit scoring models are different for stock-listed companies, medium-sized companies, financial companies or industrial companies (Altman *et al.*, 2010).

- The second category focuses on consumer credit and retail debt management (credit cards, mortgages, etc.). Sample sizes are larger than for corporate credit (Thomas, 2000) and may justify the use of more sophisticated techniques that include the behavior of the customer (Thomas *et al.*, 2017).

- The third research direction concerns statistical methods. Besides discriminant analysis, new approaches have been proposed, in particular logit or probit models (Ohlson, 1980; Lennox, 1999) and survival models (Shumway, 2001). Moreover, with the availability of more personal data, machine learning techniques such as neural networks (West, 2000) are also used and tested in credit scoring and are not reserved for only marketing scores.

15.1.2 Variable selection

15.1.2.1 Choice of the risk factors

Variables used to determine the creditworthiness of a borrower are generally based on 5 risk factor categories, also called the five Cs:

1. **Capacity** measures the applicant's ability to meet the loan payments. For example, lenders may look at the debt-to-income or the job stability of the applicant. In the case of corporate firms, the cash flow dynamics is a key element.

2. **Capital** is the size of assets that are held by the borrower. In the case of consumer credit, it corresponds to the net wealth of the borrower. For a corporate firm, it can be machinery, equipment, buildings, investment portfolio, etc.

3. **Character** measures the willingness to repay the loan. For example, the lender can investigate the payment history of the applicant. If the applicant has children, the applicant may have more incentive than if he/she is single.

4. **Collateral** concerns additional forms of security that the borrower can provide to the lender. This item is particularly important in the case of corporate credit.

5. **Conditions** refer to the characteristics of the loan and the economic conditions that might affect the borrower. For example, the score is generally a decreasing function of the maturity and the interests paid by the borrower. For corporate firms, some sectors are more dependent on the economic cycle than others.

In Table 15.1, we report some variables that are used when building a consumer credit score. This type of score is generally used by banks, since they may include information that is related to the banking relationship.

Scores are developed by banks and financial institutions, but they can also be developed by consultancy companies. This is the case of the FICO$^{®}$ scores, which are the most widely

TABLE 15.1: An example of risk factors for consumer credit

Character	Age of applicant
	Marital status
	Number of children
	Educational background
	Time with bank
	Time at present address
Capacity	Annual income
	Current living expenses
	Current debts
	Time with employer
Capital	Purpose of the loan
	Home status
	Saving account
Condition	Maturity of the loan
	Paid interests

used credit scoring systems in the world[1]. They are based on 5 main categories: payment history (35%), amount of debt (30%), length of credit history (15%), new credit (10%) and credit mix (10%).They generally range from 300 to 850, while the average score of US consumers is 695. These scores are generally classified as follows: exceptional (800+), very good (740-799), good (670-739), fair (580-669) and poor (580−).

Corporate credit scoring systems use financial ratios:

1. **Profitability**: gross profit margin, operating profit margin, return-on-equity (ROE), etc.

2. **Solvency**: debt-to-assets ratio, debt-to-equity ratio, interest coverage ratio, etc.

3. **Leverage**: liabilities-to-assets ratio (financial leverage ratio), long-term debt/assets, etc.

4. **Liquidity**: current assets/current liabilities (current ratio), quick assets/current liabilities (quick or cash ratio), total net working capital, assets with maturities of less than one year, etc.

Liquidity and solvency ratios measure the company's ability to satisfy its short-term and long-term obligations, while profitability ratios measure its ability to generate profits from its resources. High profitability, high solvency and high liquidity reduces the probability of default, but a high leverage increases the credit risk of the company. The score may also include non-financial variables: firm age[2], size (number of employees), quality of accounting information, management quality, etc. For instance, we generally consider that large firms default less often than small firms. Like retail scores, corporate scores are built by banks but also by consulting firms and credit agencies. For example, Moody's proposes the RiskCalc model (Falkenstein *et al.*, 2000).

[1]The FICO scores are developed since 1989 by Fair Isaac Corporation, which is a Californian-based firm. There are more than 20 scores that are commonly used for auto lending, credit card decisioning, mortgage lending, etc. In the US, FICO scores are used in over 90% of lending decisions (source: `https://www.myfico.com`).

[2]Recent firms may be penalized.

15.1.2.2 Data preparation

Of course data quality is essential for building a robust credit scoring. However, data preparation is not limited to check the data and remove outliers or fill missing values. Indeed, a '*one-size-fits-all*' approach is generally not appropriate, because a scoring model is generally more a decision tree system than a parsimonious econometric model. This is why credit scoring is work-intensive on data mining. Once the data is clean, we can begin the phase of exploratory data analysis, which encompasses three concurrent steps: variable transformation, slicing-and-dicing segmentation and potential interaction research. The first step consists in applying a non-linear transformation, for example by computing the logarithm, while the second and third steps are the creation of categorical/piecewise and interaction variables.

Piecewise and dummy variables Let b be a $p \times 1$ vector of bounds. We assume that b is sorted in ascending order. We note $b^{(1)} = (-\infty, b)$, $b^{(2)} = (b, +\infty)$, $b^{(3)} = (b_1, b)$ and:

$$b^{(4)} = (0, b_2 - b_1, b_3 - b_2, \ldots, b_p - b_{p-1}, 0)$$

It follows that $b^{(1)}$, $b^{(2)}$, $b^{(3)}$ and $b^{(4)}$ are four vector of dimension $(p+1) \times 1$. From the vector b, we can then create $(p+1)$ piecewise variables which are defined by:

$$PW_j = \left(X - b_j^{(3)}\right) \cdot \mathbb{1}\left\{X > b_j^{(1)}\right\} \cdot \mathbb{1}\left\{X \leq b_j^{(2)}\right\} + b_j^{(4)} \cdot \mathbb{1}\left\{X > b_j^{(2)}\right\}$$

The underlying idea is to have an affine function if the original variable takes its values in the interval $]b_{j-1}, b_j]$. For instance, Figure 15.1 represents the fourth piecewise variables which are obtained from $b = (-0.5, 0, 1)$. In a similar way, we define dummy variables as follows:

$$D_j = \mathbb{1}\left\{X > b_j^{(1)}\right\} \cdot \mathbb{1}\left\{X \leq b_j^{(2)}\right\}$$

In this case, D_j takes a value of 1 if $X \in]b_{j-1}, b_j]$. Using $b = (-0.5, 0, 1)$, we obtain Figure 15.2.

Optimal slicing An important point is the choice of the bound $b = (b_1, b_2, \ldots, b_K)$. It is obvious that the optimal values depend on the response variable Y. For that, we introduce the contingency table of the random vector (Y, X), which corresponds to a table of counts with p rows and q columns:

Y/X	$X \in \mathcal{I}_1^{(X)}$	$\cdots$	$X \in \mathcal{I}_j^{(X)}$	$\cdots$	$X \in \mathcal{I}_q^{(X)}$
$Y \in \mathcal{I}_1^{(Y)}$	$n_{1,1}$		$n_{1,j}$		$n_{1,q}$
$\vdots$					
$Y \in \mathcal{I}_i^{(Y)}$	$n_{i,1}$		$n_{i,j}$		$n_{i,q}$
$\vdots$					
$Y \in \mathcal{I}_p^{(Y)}$	$n_{p,1}$		$n_{p,j}$		$n_{p,q}$

where $n_{i,j}$ is the number of observations such that $Y \in \mathcal{I}_i^{(Y)}$ and $X \in \mathcal{I}_j^{(X)}$. We assume that the set are disjoints: $\mathcal{I}_{j_1}^{(X)} \cap \mathcal{I}_{j_2}^{(X)} = \emptyset$ for $j_1 \neq j_2$ and $\mathcal{I}_{i_1}^{(Y)} \cap \mathcal{I}_{i_2}^{(Y)} = \emptyset$ for $i_1 \neq i_2$. We introduce the following notations:

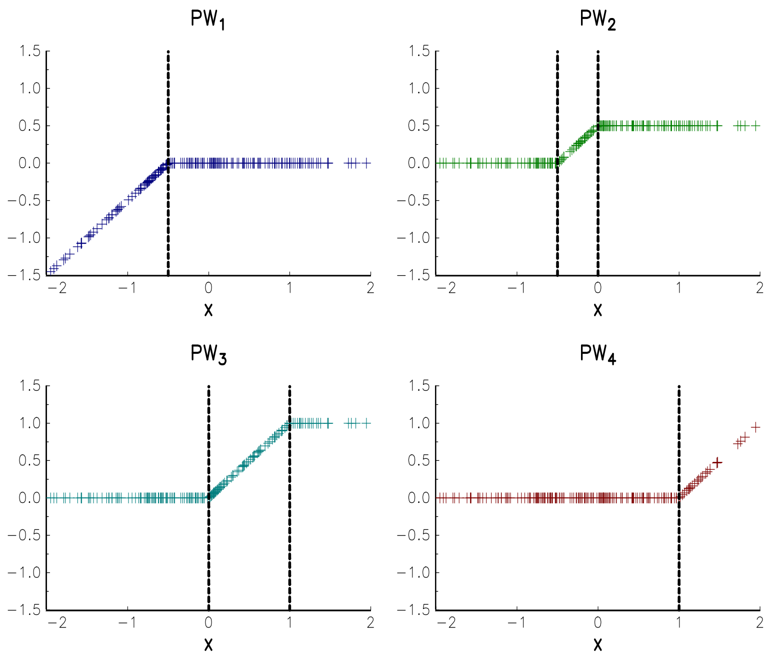

FIGURE 15.1: Piecewise variables

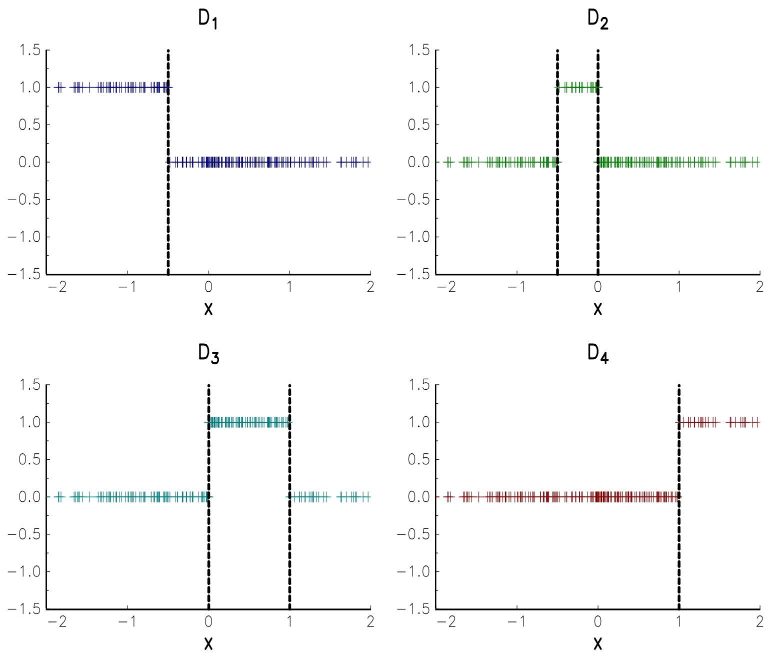

FIGURE 15.2: Dummy variables

- $n_{i,\cdot} = \sum_{j=1}^{q} n_{i,j}$ is the number of observations such that $Y \in \mathcal{I}_i^{(Y)}$;
- $n_{\cdot,j} = \sum_{i=1}^{p} n_{i,j}$ is the number of observations such that $X \in \mathcal{I}_j^{(X)}$;
- $n = \sum_{i=1}^{p} \sum_{j=1}^{q} n_{i,j}$ is the total number of observations[3].

If we assume that X and Y are independent (null hypothesis $\mathcal{H}_0$), the expected number of observations such that $Y \in \mathcal{I}_i^{(Y)}$ and $X \in \mathcal{I}_j^{(X)}$ must be equal to:

$$\bar{n}_{i,j} = \frac{n_{i,\cdot} \times n_{\cdot,j}}{n}$$

Under $\mathcal{H}_0$, we can prove that the Pearson's statistic χ has a chi-squared limit distribution:

$$\chi = \sum_{i=1}^{p} \sum_{j=1}^{q} \frac{\left(n_{i,j} - \bar{n}_{i,j}\right)^2}{\bar{n}_{i,j}} \sim \chi^2(\nu)$$

where $\nu = (p-1)(q-1)$. If we apply the Pearson's chi-squared statistic to the previous scoring problem, the contingency table becomes:

X	$X \le b_1$	$b_1 < X \le b_2$	$\cdots$	$b_{p-1} < X \le b_p$	$X > b_p$
$Y = 0$	$n_{0,1}$	$n_{0,2}$	$\cdots$	$n_{0,p}$	$n_{0,p+1}$
$Y = 1$	$n_{1,1}$	$n_{1,2}$	$\cdots$	$n_{1,p}$	$n_{1,p+1}$

We assume here that Y is a binary random variable: $Y = 0$ indicates a good credit and $Y = 1$ corresponds to a bad credit. We note $\chi(b)$ the value of the chi-squared statistic that depends on the slicing vector b:

$$\chi(b) = \sum_{i=0}^{1} \sum_{j=1}^{p+1} \frac{\left(n_{i,j} - \bar{n}_{i,j}\right)^2}{\bar{n}_{i,j}}$$

The optimal value of b is defined by:

$$b^\star = \arg\max \chi(b) \tag{15.1}$$

Indeed, if X and Y are independent, we have $\chi(b) = 0$. In this case, the variable X does not help to predict the variable Y. Maximizing the chi-squared statistic is equivalent to finding the slicing setup that deviates the most from the independent case.

In order to solve the maximization problem (15.1), we may use the dynamic programming principle, whose objective function is to solve this problem:

$$\begin{aligned} \left\{c^\star(k)\right\}_{k=1}^{K-1} &= \arg\max \sum_{k=1}^{K-1} f(k, s(k), c(k)) + f(K, s(K)) \qquad (15.2) \\ \text{s.t.} &\quad \left\{ \begin{array}{l} s(k+1) = g(k, s(k), c(k)) \\ s(k) \in \mathcal{S}(k) \\ c(k) \in \mathcal{C}(k) \\ s(1) = s \end{array} \right. \end{aligned}$$

The underlying idea is to initialize the algorithm[4] with a predetermined slice $\{b_1, b_2, \ldots, b_p\}$, to aggregate the knots in order to find the optimal slice $\left\{b_1^\star, b_2^\star, \ldots, b_{p^\star}^\star\right\}$ for a given value

[3]We also have:

$$n = \sum_{i=1}^{p} n_{i,\cdot} = \sum_{j=1}^{q} n_{\cdot,j}$$

[4]The algorithm is described on page 1049.

of $p^\star$. For that, we note $n_{i,j}(b_{j_1}, b_{j_2}) = \#(Y = i, b_{j_1} < X \leq b_{j_2})$. The chi-squared marginal contribution is defined by:

$$\chi(b_{j_1}, b_{j_2}) = \sum_{i=0}^{1} \frac{(n_{i,j}(b_{j_1}, b_{j_2}) - \bar{n}_{i,j}(b_{j_1}, b_{j_2}))^2}{\bar{n}_{i,j}(b_{j_1}, b_{j_2})} \qquad \text{for } j_1 < j_2$$

$\chi(b_{j_1}, b_{j_2})$ can be viewed as the Pearson's statistic when we only consider the observations such that $b_{j_1} < X \leq b_{j_2}$. The gain function is equal to:

$$f(k, s(k), c(k)) = \begin{cases} -\infty & \text{if } c(k) \leq s(k) \\ \chi(b_{s(k)+1}, b_{c(k)}) & \text{otherwise} \end{cases}$$

If $k = 1$, we have:

$$f(1, s(1), c(1)) = \begin{cases} -\infty & \text{if } c(1) \leq s(1) \\ \chi(b_0, b_{s(1)}) + \chi(b_{s(1)+1}, b_{c(1)}) & \text{otherwise} \end{cases}$$

The transfer function is defined as follows:

$$s(k+1) = g(k, s(k), c(k)) = c(k)$$

The state variable $s(k)$ and the control variable $c(k)$ take their values in the set $\{1, 2, \ldots, p\}$. The number K of iterations is exactly equal to $p^\star$ and we have:

$$f(K, s_j) = \chi(b_{s_j+1}, b_p)$$

In the case where $p^\star = 1$, the dynamic programming algorithm reduces to the brute force algorithm:

$$j^\star = \underset{j \in \{b_1, b_2, \ldots, b_p\}}{\arg\max} \; \chi(-\infty, b_j) + \chi(b_j, \infty)$$

In this case, the optimal slice is composed of two classes: $X \leq b_{j^\star}$ and $X > b_{j^\star}$.

Example 163 *We consider 40 observations of the random vector (Y, X). Below, we indicate the values taken by X when $Y = 0$ and $Y = 1$:*

- $Y = 0$: *−2.0, −1.1, −1.0, −0.7, −0.5, −0.5, −0.4, −0.3, −0.2, −0.2, 0.0, 0.7, 0.8, 0.9, 1.0, 1.4, 1.9, 2.8, 3.2, 3.7.*
- $Y = 1$: *−5.2, −4.3, −3.6, −2.7, −1.8, −1.5, −1.2, −1.0, −0.8, −0.1, 0.0, 0.2, 0.2, 0.3, 0.5, 0.5, 0.5, 0.7, 0.8, 1.9.*

If we consider the following grid $b = (-5, -4, -3, -2, -1, 0, 1, 2, 3)$, we obtain the following contingency table:

X	$\mathcal{I}_1^{(X)}$	$\mathcal{I}_2^{(X)}$	$\mathcal{I}_3^{(X)}$	$\mathcal{I}_4^{(X)}$	$\mathcal{I}_5^{(X)}$	$\mathcal{I}_6^{(X)}$	$\mathcal{I}_7^{(X)}$	$\mathcal{I}_8^{(X)}$	$\mathcal{I}_9^{(X)}$	$\mathcal{I}_{10}^{(X)}$
$Y = 0$	0	0	0	0	2	8	4	3	1	2
$Y = 1$	1	1	1	1	3	3	9	1	0	0

where $\mathcal{I}_1^{(X)} = \{X \leq -5\}$, $\mathcal{I}_2^{(X)} = \{-5 < X \leq -4\}$, ..., $\mathcal{I}_{10}^{(X)} = \{X > -4\}$. If we would like to slice X into two classes, we use the brute force algorithm. If we group the intervals $\left\{\mathcal{I}_2^{(X)}, \ldots, \mathcal{I}_{10}^{(X)}\right\}$, the contingency table becomes:

X	$X \leq -5$	$X > -5$	$n_{i,\cdot}$
$Y = 0$	0	20	20
$Y = 1$	1	19	20
$n_{\cdot,j}$	1	39	$n = 40$

We deduce that:

$$\begin{aligned}\chi &= \frac{(0-0.5)^2}{0.5} + \frac{(20-19.5)^2}{19.5} + \frac{(1-0.5)^2}{0.5} + \frac{(19-19.5)^2}{19.5} \\ &= 1.02564\end{aligned}$$

If we now consider the two groups $\left\{\mathcal{I}_1^{(X)}, \mathcal{I}_2^{(X)}\right\}$ and $\left\{\mathcal{I}_3^{(X)}, \ldots, \mathcal{I}_{10}^{(X)}\right\}$, we obtain the following contingency table:

X	$X \leq -4$	$X > -4$	$n_{i,\cdot}$
$Y = 0$	0	20	20
$Y = 1$	2	18	20
$n_{\cdot,j}$	2	38	$n = 40$

The associated Pearson's chi-squared statistic is then equal to:

$$\begin{aligned}\chi &= \frac{(0-1.0)^2}{1.0} + \frac{(20-19.0)^2}{19.0} + \frac{(2-1.0)^2}{1.0} + \frac{(18-19.0)^2}{19.0} \\ &= 2.10526\end{aligned}$$

We can proceed in the same way with the other values of b and we obtain the following values of χ when the cut-off is b_j:

X	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
χ	1.03	2.11	3.24	4.44	3.58	0.00	4.33	3.24	2.11

Since the maximum is reached for b_4 ($\chi = 4.44$), the optimal slicing is the following:

X	$X \leq -2$	$X > -2$
$Y = 0$	0	20
$Y = 1$	4	16

If we prefer to slice X into three classes, the dynamic programming algorithm finds that the optimal cut-offs are $b^\star = (-2, 1)$. In the case of four classes, the optimal slicing is:

X	$X \leq -1$	$-1 < X \leq 0$	$0 < X \leq 1$	$X > 1$
$Y = 0$	2	8	4	6
$Y = 1$	7	3	9	1

and the optimal value $\chi^\star$ is equal to 10.545. In order to understand how does the dynamic programming algorithm work, we report the $\boldsymbol{J}$ and $\boldsymbol{C}$ matrices in Table 15.2. We notice that the optimal value is $\mathcal{J}(1, s^\star(1)) = 10.545$ where $s^\star(1)$ is the 5th state. Moreover, the optimal controls are $c^\star(1) = 6$ and $c^\star(2) = 7$ implying that $s^\star(2) = c^\star(1)$ and $s^\star(3) = c^\star(2)$ are the 6th and 7th states. This is why the optimal cut-offs are $b^\star = (-1, 0, 1)$, that is the 5th, 6th and 7th elements of the initial vector b.

Remark 178 *We notice that the optimal slice $b^\star$ depends on the initial grid b. This implies that another grid b will not necessarily give the same optimal slice. For instance, we have used a step of 1 in the previous example. If we use a step of 0.2, we obtain the optimal solution $b^\star = (-0.8, -0.2, 0.6)$. We have reported the corresponding slicing in Figure 15.3. In this case, the Pearson's chi-squared statistic is equal to 18.444, which is better than the value 10.545 obtained previously. This is why it is better to use a small step than a large step. The risk is that the dynamic programming algorithm produces some classes with a low number of observations. To prevent this possible overfitting, we can impose that $\chi(b_{j_1}, b_{j_2}) = -\infty$ when the number of observations is below a given threshold ($\#(b_{j_1} < X \leq b_{j_2}) \leq n_{\min}$). This ensures that each optimized class has at least $n_{\min}$ observations.*

TABLE 15.2: Dynamic programming matrices $\boldsymbol{J}$ and $\boldsymbol{C}$

state	$\mathcal{J}(1,s(1))$	$\mathcal{J}(1,s(2))$	$\mathcal{J}(1,s(3))$	$c(1)$	$c(2)$
1	7.6059	4.0714	0.0256	4	7
2	7.7167	3.8618	0.1053	6	7
3	9.0239	3.7048	0.2432	6	7
4	10.4945	3.6059	0.4444	6	7
5	10.5450	3.5714	0.8065	6	7
6	5.9231	5.4945	0.0000	7	7
7	4.7576	4.0000	3.5714	8	8
8	$-\infty$	3.0000	3.0000	1	9
9	$-\infty$	$-\infty$	2.0000	1	1

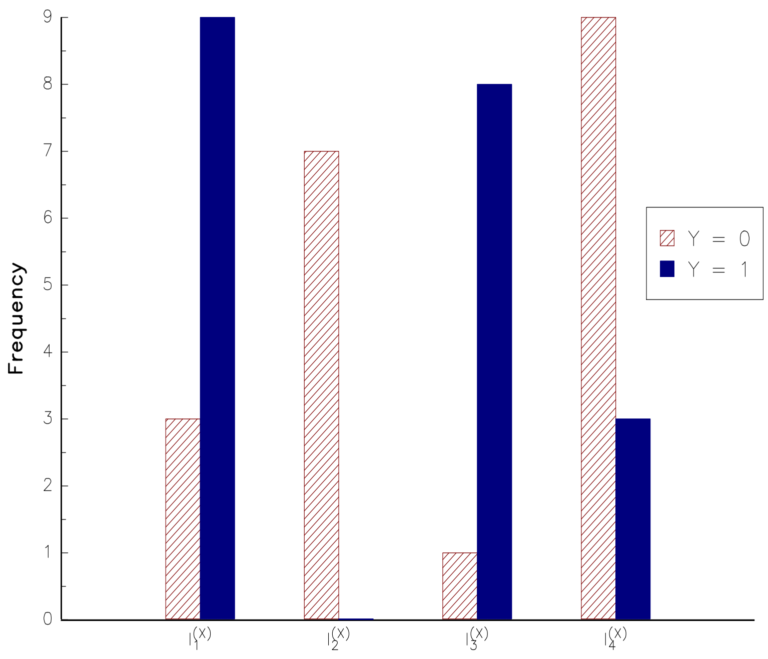

FIGURE 15.3: Optimal slicing with four classes

15.1.2.3 Variable selection

In practice, one may have many candidate variables $X = (X_1, \ldots, X_m)$ for explaining the variable Y. The variable selection problem consists in finding the best set of optimal variables. Let us assume the following statistical model:

$$Y = f(X) + u$$

where $u \sim \mathcal{N}\left(0, \sigma^2\right)$. We denote the prediction by $\hat{Y} = \hat{f}(X)$. By assuming the standard statistical hypotheses, we obtain:

$$\begin{aligned}
\mathbb{E}\left[\left(Y - \hat{Y}\right)^2\right] &= \mathbb{E}\left[\left(f(X) + u - \hat{f}(X)\right)^2\right] \\
&= \mathbb{E}\left[\left(f(X) - \hat{f}(X)\right)^2\right] + \mathbb{E}\left[u^2\right] \\
&= \left(\mathbb{E}\left[\hat{f}(X)\right] - f(X)\right)^2 + \mathbb{E}\left[\left(\hat{f}(X) - \mathbb{E}\left[\hat{f}(X)\right]\right)^2\right] + \sigma^2 \\
&= \text{Bias}^2 + \text{Variance} + \text{Error}
\end{aligned}$$

Hastie *et al.* (2009) decompose the mean squared error of $\hat{f}(X)$ into three terms: a bias component, a variance component and an irreducible error. This bias-variance decomposition depends on the complexity of the model. When the model complexity is low (*i.e.* when there is a low number of regressors), the estimator $\hat{f}(X)$ generally presents a high bias but a low variance. When the model complexity is high (*i.e.* when there is a high number of regressors), the estimator $\hat{f}(X)$ generally presents a low bias but a high variance. The underlying idea of variable selection is then to optimize the bias-variance trade-off.

Best subset selection A first approach is to find the best subset of size k for $k \in \{1, \ldots, m\}$ that gives the smallest residual sum of squares. It follows that the search is performed through 2^m possible subsets, meaning that we rapidly face a combinatorial explosion. Moreover, minimizing the residual sum of squares is equivalent to consider the largest subset $(1, \ldots, m)$. This is why we prefer to consider an information criterion that penalizes the degree of freedom of the model. For instance, the Akaike criterion is defined as follows:

$$\text{AIC}(\alpha) = -2\boldsymbol{\ell}_{(k)}\left(\hat{\theta}\right) + \alpha \cdot \text{df}_{(k)}^{(\text{model})}$$

where $\boldsymbol{\ell}_{(k)}\left(\hat{\theta}\right)$ and $\text{df}_{(k)}^{(\text{model})}$ are the log-likelihood and the degree of freedom of the k^{th} model[5]. Therefore, the best model corresponds to the model that minimizes the Akaike criterion. In practice, the penalization parameter is generally set to $\alpha = 2$. In the case of the previous model, we deduce that:

$$\text{AIC}(2) = n \ln\left(\frac{\text{RSS}\left(\hat{\theta}\right)}{n}\right) + 2\,\text{df}_{(k)}^{(\text{model})}$$

Stepwise approach Another way for selecting variables is to use sequential approaches: forward selection, backward selection and forward/backward combined selection. In the case of forward selection, we start with the intercept and include one variable by one variable. At each step, we select the model of dimension $k+1$ with the most significant F-value with respect to the previous optimal model of dimension k:

$$F = \frac{\text{RSS}\left(\hat{\theta}_{(k)}\right) - \text{RSS}\left(\hat{\theta}_{(k+1)}\right)}{\text{RSS}\left(\hat{\theta}_{(k+1)}\right) / \text{df}_{(k+1)}^{(\text{residual})}}$$

[5] $\text{df}_{(k)}^{(\text{model})}$ is a complexity measure of the model, and corresponds to the number of estimated parameters. It is sometimes called the '*model degree of freedom*' whereas the classical measure used in linear regression t-statistics $\text{df}_{(k)}^{(\text{residual})}$ is called the '*residual degree of freedom*'. We have the following relationship $\text{df}_{(k)}^{(\text{residual})} = n - \text{df}_{(k)}^{(\text{model})}$ where n is the number of observations.

We stop when no model produces a significant F-value at the 95% confidence level. In the case of backward selection, we start with all the variables and remove one variable by one variable. At each step, we select the model of dimension k with the smallest significant F-value with respect to the previous optimal model of dimension $k+1$. The forward/backward combined procedure consists in using a forward step followed by a backward step, and to iterate this loop until the convergence criterion is reached. The convergence criterion can be expressed as a maximum number of loops[6].

Lasso approach The lasso method consists in adding a L_1 penalty function to the optimization function in order to obtain a sparse parameter vector θ:

$$L_1(\theta) = \|\theta\|_1 = \sum_{k=1}^{K} |\theta_k|$$

For example, the lasso regression model is specified as follows (Tibshirani, 1996):

$$y_i = \sum_{k=1}^{K} \beta_k x_{i,k} + u_i \quad \text{s.t.} \quad \sum_{k=1}^{K} |\beta_k| \le \tau$$

where τ is a scalar to control the sparsity. Using the notations introduced on page 604, we have:

$$\begin{aligned} \hat{\beta}(\tau) &= \arg\min (\mathbf{Y} - \mathbf{X}\beta)^\top (\mathbf{Y} - \mathbf{X}\beta) \\ \text{s.t.} \quad & \|\beta\|_1 \le \tau \end{aligned} \tag{15.3}$$

This problem is equivalent to the Lagrange optimization program $\hat{\beta}(\lambda) = \arg\min \mathcal{L}(\beta; \lambda)$ where[7]:

$$\begin{aligned} \mathcal{L}(\beta;\lambda) &= \frac{1}{2}(\mathbf{Y} - \mathbf{X}\beta)^\top (\mathbf{Y} - \mathbf{X}\beta) + \lambda \|\beta\|_1 \\ &\propto \frac{1}{2}\beta^\top \left(\mathbf{X}^\top \mathbf{X}\right)\beta - \beta^\top \left(\mathbf{X}^\top \mathbf{Y}\right) + \lambda \|\beta\|_1 \end{aligned}$$

The solution $\hat{\beta}(\lambda)$ can be found by solving the augmented QP program where $\beta = \beta^+ - \beta^-$ under the constraints $\beta^+ \ge \mathbf{0}$ and $\beta^- \ge \mathbf{0}$. We deduce that:

$$\begin{aligned} \|\beta\|_1 &= \sum_{k=1}^{K} \left|\beta_k^+ - \beta_k^-\right| \\ &= \sum_{k=1}^{K} \left|\beta_k^+\right| + \sum_{k=1}^{K} \left|\beta_k^-\right| \\ &= \mathbf{1}^\top \beta^+ + \mathbf{1}^\top \beta^- \end{aligned}$$

Since we have:

$$\beta = \begin{pmatrix} I_K & -I_K \end{pmatrix} \begin{pmatrix} \beta^+ \\ \beta^- \end{pmatrix}$$

the augmented QP program is specified as follows:

$$\begin{aligned} \hat{\theta} &= \arg\min \frac{1}{2}\theta^\top Q \theta - \theta^\top R \\ \text{s.t.} \quad & \theta \ge \mathbf{0} \end{aligned}$$

[6]The algorithm also stops when the variable to be added is the same as the last deleted variable.
[7]τ and λ are related by the relationship $\tau = \left\|\hat{\beta}(\lambda)\right\|_1$.

where $\theta = (\beta^+, \beta^-)$, $\tilde{\mathbf{X}} = \begin{pmatrix} \mathbf{X} & -\mathbf{X} \end{pmatrix}$, $Q = \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$ and $R = \tilde{\mathbf{X}}^\top \mathbf{Y} - \lambda \cdot \mathbf{1}$. If we denote $A = \begin{pmatrix} I_K & -I_K \end{pmatrix}$, we obtain:

$$\hat{\beta}(\lambda) = A\hat{\theta}$$

Remark 179 *If we consider Problem (15.3), we can also solve it using another augmented QP program:*

$$\begin{array}{rcl} \hat{\theta} & = & \arg\min \frac{1}{2}\theta^\top Q\theta - \theta^\top R \\ & & \text{s.t.} \left\{ \begin{array}{l} C\theta \geq D \\ \theta \geq \mathbf{0} \end{array} \right. \end{array}$$

where $Q = \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$, $R = \tilde{\mathbf{X}}^\top \mathbf{Y}$, $C = -\mathbf{1}^\top$ and $D = -\tau$. We again have $\hat{\beta}(\tau) = A\hat{\theta}$.

We have:

$$\begin{array}{rcl} \mathrm{RSS}(\beta) & = & (\mathbf{Y} - \mathbf{X}\beta)^\top (\mathbf{Y} - \mathbf{X}\beta) \\ & = & \left(\mathbf{Y} - \mathbf{X}\left(\hat{\beta}^{\mathrm{ols}} + \beta - \hat{\beta}^{\mathrm{ols}}\right)\right)^\top \left(\mathbf{Y} - \mathbf{X}\left(\hat{\beta}^{\mathrm{ols}} + \beta - \hat{\beta}^{\mathrm{ols}}\right)\right) \\ & = & \left(\mathbf{Y} - \mathbf{X}\hat{\beta}^{\mathrm{ols}}\right)^\top \left(\mathbf{Y} - \mathbf{X}\hat{\beta}^{\mathrm{ols}}\right) + 2\left(\mathbf{Y} - \mathbf{X}\hat{\beta}^{\mathrm{ols}}\right)^\top \mathbf{X}\left(\beta - \hat{\beta}^{\mathrm{ols}}\right) + \\ & & \left(\beta - \hat{\beta}^{\mathrm{ols}}\right)^\top \mathbf{X}^\top \mathbf{X}\left(\beta - \hat{\beta}^{\mathrm{ols}}\right) \end{array}$$

We notice that:

$$\begin{array}{rcl} (*) & = & \left(\mathbf{Y} - \mathbf{X}\hat{\beta}^{\mathrm{ols}}\right)^\top \mathbf{X}\left(\beta - \hat{\beta}^{\mathrm{ols}}\right) \\ & = & \left(\mathbf{Y}^\top - \left(\hat{\beta}^{\mathrm{ols}}\right)^\top \mathbf{X}^\top\right) \mathbf{X}\left(\beta - \hat{\beta}^{\mathrm{ols}}\right) \\ & = & \left(\mathbf{Y}^\top - \left(\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{Y}\right)^\top \mathbf{X}^\top\right) \mathbf{X}\left(\beta - \hat{\beta}^{\mathrm{ols}}\right) \\ & = & \left(\mathbf{Y}^\top\mathbf{X} - \left(\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{Y}\right)^\top \mathbf{X}^\top\mathbf{X}\right) \left(\beta - \hat{\beta}^{\mathrm{ols}}\right) \\ & = & \left(\mathbf{Y}^\top\mathbf{X} - \mathbf{Y}^\top\mathbf{X}\right)\left(\beta - \hat{\beta}^{\mathrm{ols}}\right) \\ & = & 0 \end{array}$$

Finally, we obtain:

$$\mathrm{RSS}(\beta) = \mathrm{RSS}\left(\hat{\beta}^{\mathrm{ols}}\right) + \left(\beta - \hat{\beta}^{\mathrm{ols}}\right)^\top \mathbf{X}^\top \mathbf{X}\left(\beta - \hat{\beta}^{\mathrm{ols}}\right)$$

If we consider the equation $\mathrm{RSS}(\beta) = c$, we distinguish three cases:

1. if $c < \mathrm{RSS}\left(\hat{\beta}^{\mathrm{ols}}\right)$, there is no solution;

2. if $c = \mathrm{RSS}\left(\hat{\beta}^{\mathrm{ols}}\right)$, there is one solution $\beta^\star = \hat{\beta}^{\mathrm{ols}}$;

3. if $c > \mathrm{RSS}\left(\hat{\beta}^{\mathrm{ols}}\right)$, we have:

$$\left(\beta - \hat{\beta}^{\mathrm{ols}}\right)^\top A\left(\beta - \hat{\beta}^{\mathrm{ols}}\right) = 1$$

where:

$$A = \frac{\mathbf{X}^\top \mathbf{X}}{c - \operatorname{RSS}\left(\hat{\beta}^{\text{ols}}\right)}$$

The solution $\beta^\star$ is an ellipsoid, whose center is $\hat{\beta}^{\text{ols}}$ and principal axes are the eigenvectors of the matrix A.

If we add the lasso constraint $\sum_{k=1}^{K} |\beta_k| \leq \tau$, the lasso estimator $\hat{\beta}(\tau)$ corresponds to the tangency between the diamond shaped region and the ellipsoid that corresponds to the possible maximum value of c. The diamond shape region due to the lasso constraint ensures that the lasso estimator is sparse:

$$\exists\, \eta > 0 : \forall\, \tau < \eta, \ \min\left(\hat{\beta}_1(\tau), \ldots, \hat{\beta}_K(\tau)\right) = 0$$

For example, the two-dimensional case is represented in Figure 15.4. We notice that $\hat{\beta}_1(\tau)$ is equal to zero if $\tau < \eta$. This sparsity property is central for understanding the variable selection procedure.

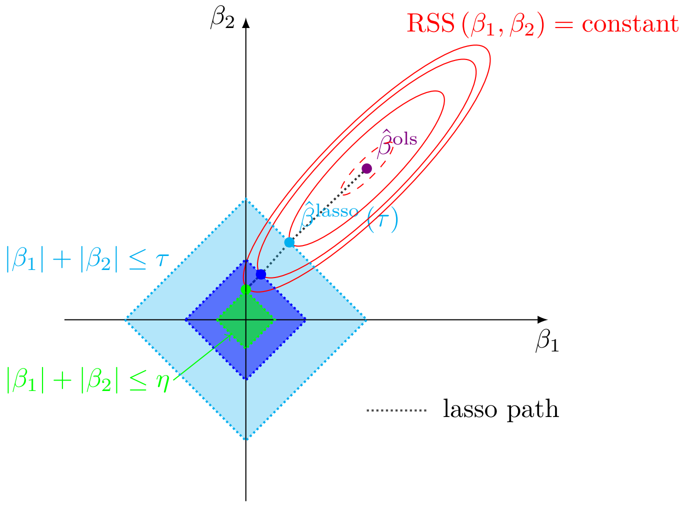
β_2
$\operatorname{RSS}(\beta_1, \beta_2) = \text{constant}$
$\hat{\beta}^{\text{ols}}$
$\hat{\beta}^{\text{lasso}}(\tau)$
$|\beta_1| + |\beta_2| \leq \tau$
β_1
$|\beta_1| + |\beta_2| \leq \eta$
lasso path

FIGURE 15.4: Interpretation of the lasso regression

Example 164 *Using the data given in Table 15.3, we consider the linear regression model:*

$$y_i = \beta'_0 + \sum_{k=1}^{5} \beta'_k x_{i,k} + u_i \tag{15.4}$$

The objective is to determine the importance of each variable.

The lasso method can be used for ranking the variables. For that, we consider the following linear regression:

$$\tilde{y}_i = \sum_{k=1}^{5} \beta_k \tilde{x}_{i,k} + u_i$$

TABLE 15.3: Data of the lasso regression problem

i	y	x_1	x_2	x_3	x_4	x_5
1	3.1	2.8	4.3	0.3	2.2	3.5
2	24.9	5.9	3.6	3.2	0.7	6.4
3	27.3	6.0	9.6	7.6	9.5	0.9
4	25.4	8.4	5.4	1.8	1.0	7.1
5	46.1	5.2	7.6	8.3	0.6	4.5
6	45.7	6.0	7.0	9.6	0.6	0.6
7	47.4	6.1	1.0	8.5	9.6	8.6
8	−1.8	1.2	9.6	2.7	4.8	5.8
9	20.8	3.2	5.0	4.2	2.7	3.6
10	6.8	0.5	9.2	6.9	9.3	0.7
11	12.9	7.9	9.1	1.0	5.9	5.4
12	37.0	1.8	1.3	9.2	6.1	8.3
13	14.7	7.4	5.6	0.9	5.6	3.9
14	−3.2	2.3	6.6	0.0	3.6	6.4
15	44.3	7.7	2.2	6.5	1.3	0.7

where $\tilde{y}_i$ and $\tilde{x}_{i,k}$ are the standardized data[8]:

$$\frac{y_i - \bar{y}}{s_y} = \sum_{k=1}^{5} \beta_k \left(\frac{x_{i,k} - \bar{x}_k}{s_{x_k}} \right) + u_i \tag{15.5}$$

Linear regressions (15.4) and (15.5) are related by the following equation:

$$y_i = \left(\bar{y} - \sum_{k=1}^{5} \frac{s_y \beta_k}{s_{x_k}} \bar{x}_k \right) + \sum_{k=1}^{5} \frac{s_y \beta_k}{s_{x_k}} x_{i,k} + s_y u_i$$

We deduce that $\beta'_0 = \bar{y} - \sum_{k=1}^{5} \left(s_y / s_{x_k} \right) \beta_k \bar{x}_k$ and $\beta'_k = \left(s_y / s_{x_k} \right) \beta_k$. When performing lasso regression, we always standardize the data in order to obtain comparable beta's. Otherwise, the penalty function $\|\beta\|_1$ does not make a lot of sense. In Table 15.4, we have estimated the lasso coefficients $\beta_k(\lambda)$ for different values of the shrinkage parameter λ. When $\lambda = 0$, we obtain the OLS estimate, and the lasso regression selects all the available variables. When $\lambda \to \infty$, the solution is $\hat{\beta}(\infty) = \mathbf{0}$, and the lasso regression selects no explanatory variables. In Table 15.4, we verify that the number of selected variables is a decreasing function of λ. For instance, the lasso regression selects respectively four and three variables when λ is equal to 0.9 and 2.5. It follows that the most important variable is the third one, followed by the first, second, fourth and fifth variables.

In Figure 15.5, we have reported the path of the lasso estimate $\hat{\beta}(\lambda)$ with respect to the scaling factor $\tau^\star \in [0,1]$, which is defined as follows:

$$\tau^\star = \frac{\tau}{\tau_{\max}} = \frac{\left\| \hat{\beta}(\lambda) \right\|_1}{\left\| \hat{\beta}(0) \right\|_1}$$

[8] The notations $\bar{x}_k$ and s_{x_k} represent the mean and the standard deviation of the data $\left\{ x_{i,k}, i = 1, \ldots, n \right\}$.

$\tau^{\star}$ is equal to zero when $\lambda \to \infty$ (no selected variable) and one when $\lambda = 0$, which corresponds to the OLS case. From this path, we verify the lasso ordering:

$$x_3 \succ x_1 \succ x_2 \succ x_4 \succ x_5$$

TABLE 15.4: Results of the lasso regression

λ	0.0	0.9	2.5	5.5	7.5
$\hat{\beta}_1(\lambda)$	0.4586	0.4022	0.3163	0.1130	
$\hat{\beta}_2(\lambda)$	−0.1849	−0.2005	−0.1411		
$\hat{\beta}_3(\lambda)$	0.8336	0.7265	0.5953	0.3951	0.2462
$\hat{\beta}_4(\lambda)$	−0.1893	−0.1102			
$\hat{\beta}_5(\lambda)$	0.0931				
$\left\Vert\hat{\beta}(\lambda)\right\Vert_1$	1.7595	1.4395	1.0527	0.5081	0.2462
$\mathrm{RSS}\left(\hat{\beta}(\lambda)\right)$	0.0118	0.0304	0.1180	0.4076	0.6306
R_c^2	0.9874	0.9674	0.8735	0.5633	0.3244
$\mathrm{df}^{(\mathrm{model})}$	5	4	3	2	1

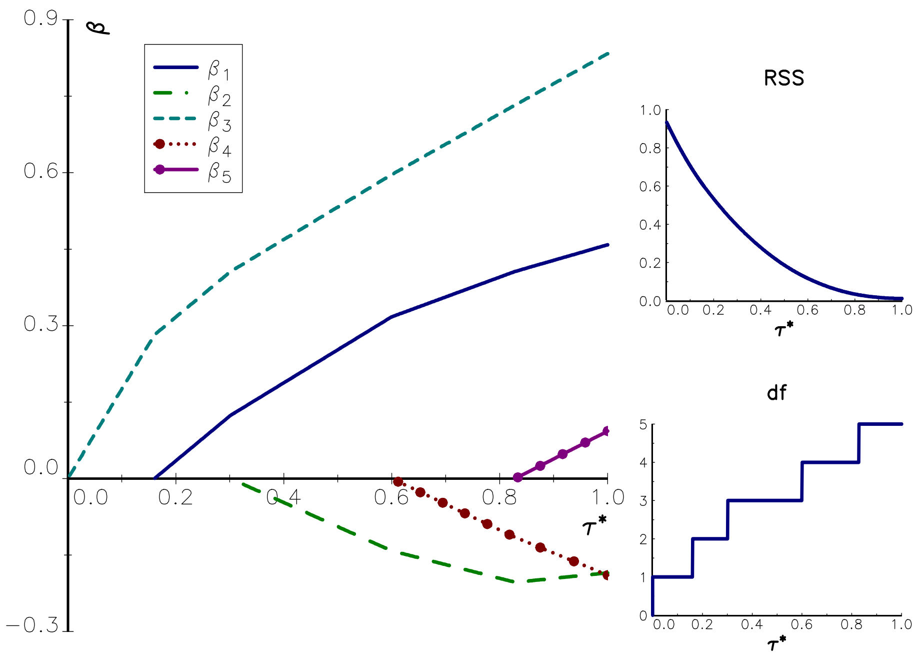

FIGURE 15.5: Variable selection with the lasso regression

15.1.3 Score modeling, validation and follow-up

15.1.3.1 Cross-validation approach

In order to avoid overfitting, we can also split the dataset into a training set and a validation set. The training set is used to estimate the model, for example the vector θ in the case of a parametric model, while the validation set is used to compute the prediction error and the residual sum of squares. This approach can be generalized for model selection. In this case, the training set is used to fit the several models, while the validation set is used to select the right model (Hastie *et al.*, 2009). We generally distinguish two types of cross-validation.

1. In exhaustive cross-validation methods, learning and testing are based on all possible ways to divide the original sample into a training set and a validation set. For example, leave-p-out cross-validation (LpOCV) assumes that the validation set is composed of p observations, while the training set corresponds to the remaining observations. Since the number of training and validation sets is equal to C_p^n, this approach may be computationally intensive. In order to reduce the complexity, we can choose $p = 1$. This approach is called the leave-one-out cross-validation (LOOCV).

2. Non-exhaustive cross-validation methods split the original sample into training and validation sets. For instance, the k-fold approach randomly divides the dataset into k (almost) equally sized subsamples. At each iteration, one subsample is choosen as a validation set, while the $k-1$ remaining subsamples form the training set. This means that the model is fitted using all but the j^{th} group of data, and the j^{th} group of data is used for the test set. We repeat the procedure k times, in such a way that each subsample is tested exactly once. In the case of a linear regression, the k-fold cross validation error is generally computed as:

$$\mathcal{E}_{\text{cv}} = \frac{1}{n}\sum_{j=1}^{k}\sum_{i\in\mathcal{G}_j}\left(y_i - x_i^\top \hat{\beta}\left(j\right)\right)^2$$

where $i \in \mathcal{G}_j$ denotes the observations of the j^{th} subsample and $\hat{\beta}\left(j\right)$ the estimate of β obtained by leaving out the j^{th} subsample. Even in simple cases, it cannot be guaranteed that the function $\mathcal{E}_{\text{cv}}$ has a unique minimum. The simple grid search approach is probably the best approach. The exhaustive leave-one-out cross validation (LOOCV) is a particular case when k is equal to the size of the dataset. Moreover, we can show that LOOCV is asymptotically equivalent to the AIC criterion (Stone, 1977).

In order to illustrate the principle of cross-validation, we consider the ridge estimator:

$$\hat{\beta} = \arg\min \frac{1}{2}\left(\mathbf{Y} - \mathbf{X}\beta\right)^\top\left(\mathbf{Y} - \mathbf{X}\beta\right) + \frac{\lambda}{2}\beta^\top\beta$$

where $\mathbf{Y}$ is a $n \times 1$ vector, $\mathbf{X}$ is a $n \times K$ matrix and β is a $K \times 1$ vector. The ridge model is then a regularized linear regression model with a L_2-norm penalty (Hoerl and Kennard, 1970). It follows that the expression of $\hat{\beta}$ is equal to:

$$\hat{\beta} = \left(\mathbf{X}^\top\mathbf{X} + \lambda I_K\right)^{-1}\mathbf{X}^\top\mathbf{Y}$$

In the case of the leave-one-out cross validation, Allen (1971, 1974) showed that the function $\mathcal{E}_{\text{cv}}$ has an explicit expression known as the predicted residual error sum of squares (or PRESS) statistic:

$$\mathcal{P}\text{ress} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_{i,-i}\right)^2$$

where $\hat{y}_{i,-i}$ is the estimate of y_i based on the ridge model when leaving out the i^{th} observation. Indeed, we have[9]:

$$\mathcal{P}\text{ress} = \frac{1}{n}\sum_{i=1}^{n} \frac{\hat{u}_i^2}{\left(1 - h_i\right)^2}$$

where $\hat{u}_i = y_i - x_i^\top \hat{\beta}$ and $h_i = x_i^\top \left(\mathbf{X}^\top \mathbf{X} + \lambda I_K\right)^{-1} x_i$. With this formula, we don't need to estimate the n estimators $\hat{\beta}_{-i}$, where $\hat{\beta}_{-i}$ is the ridge estimator when leaving out the i^{th} observation.

TABLE 15.5: Data of the ridge regression problem

i	y	x_1	x_2	x_3	x_4	x_5
1	−23.0	−8.0	6.0	−12.7	9.5	−7.5
2	−21.0	−6.5	11.1	5.4	6.6	6.7
3	−5.0	−14.4	−13.3	−3.2	0.8	1.0
4	−39.6	−6.7	26.0	11.5	15.5	6.5
5	5.8	2.3	−7.1	−4.6	7.0	−0.6
6	13.6	2.0	−13.0	−13.3	−0.9	−8.6
7	14.0	10.7	−4.9	−23.1	2.5	19.0
8	−5.2	−8.5	1.0	4.2	−11.5	12.9
9	6.9	3.4	4.9	9.5	−12.8	11.0
10	−5.2	0.0	5.1	−14.3	−3.8	−10.0
11	0.0	1.0	4.0	14.1	−3.5	−23.6
12	3.0	2.4	1.6	−1.2	−4.8	−9.2
13	9.2	−0.1	−10.6	16.0	7.5	5.8
14	26.1	15.2	2.5	5.3	−18.0	10.4
15	−6.3	−19.2	−20.7	−5.1	3.9	−13.8
16	11.5	10.1	1.7	−12.1	−2.7	13.9
17	4.8	3.8	0.8	2.7	1.0	14.4
18	35.2	23.1	1.2	−5.0	−16.1	3.3
19	14.0	13.1	6.6	1.6	−7.4	−3.5
20	−21.4	−19.0	0.7	0.8	−2.7	11.3

Example 165 *Using the data given in Table 15.5, we consider the linear regression model:*

$$y_i = \sum_{k=1}^{5} \beta_k x_{i,k} + u_i$$

The objective is to determine the ridge parameter λ by cross-validation.

In order to estimate the optimal value of λ, we calculate the PRESS function and find its minimum:

$$\lambda^\star = \arg\min \mathcal{P}\text{ress}\left(\lambda\right)$$

In Figure 15.6, we have represented the PRESS function for several values of λ. Using a bisection, we deduce that the optimal value is $\lambda^\star = 3.36$.

Remark 180 *The ridge regression is a good example where we can obtain an analytical formula for the cross-validation error $\mathcal{E}_{\text{cv}}$. In most of statistical models, this is not the case*

[9]See Exercise 15.4.2 on page 1022.

and we have to use a grid search for selecting the optimal model. This approach may be time-consuming. However, since the calibration of credit scoring models is done once per year, it is not an issue. Nevertheless, the computational time of cross-validation may be prohibitive with on-the-fly or real time statistical models.

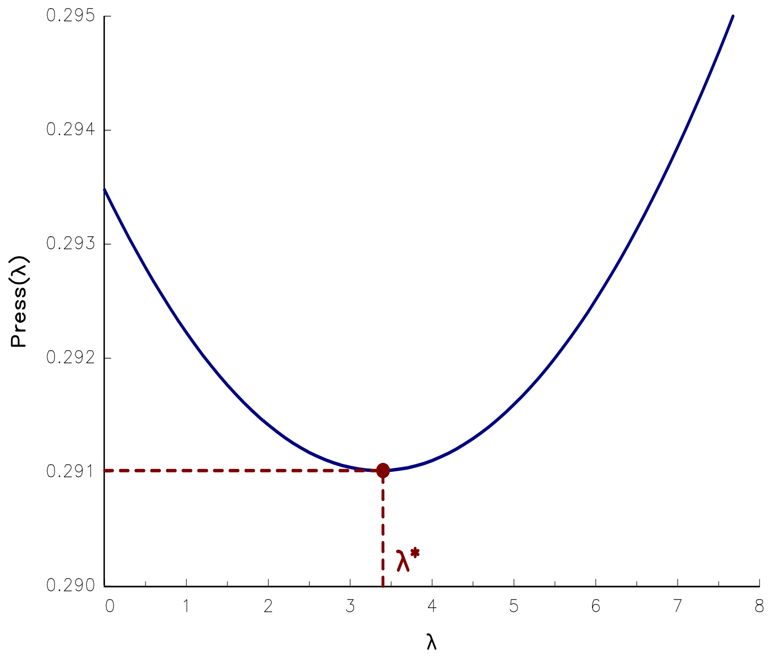

FIGURE 15.6: Selection of the ridge parameter using the PRESS statistic

15.1.3.2 Score modeling

Score modeling is the backbone of credit scoring. This is why it is extensively studied in the next two sections of this chapter. However, we present here some elements in order to understand the main challenges. The score is generally a (non-linear) function of exogenous variables X and parameters θ: $S = f(X;\theta)$. We assume that θ has been already estimated by a statistical inference method and the model $S = f\left(X;\hat{\theta}\right)$ has been validated. For example, $f\left(X;\hat{\theta}\right)$ may be a ridge regression model where $\hat{\theta} = \left(\hat{\beta},\hat{\lambda}\right)$ and $\hat{\lambda}$ has been calibrated by a cross-validation method. The score estimation $S = f\left(X;\hat{\theta}\right)$ is the preliminary part of the decision rule. Indeed, we have now to decide if we select or not the applicant. It can be done using the following rule:

$$\left\{\begin{array}{l} S < s \Longrightarrow Y = 0 \Longrightarrow \text{reject} \\ S \geq s \Longrightarrow Y = 1 \Longrightarrow \text{accept} \end{array}\right.$$

The difficulty lies in the choice of the cut-off s. For instance, if the model is a logit model, the score is a probability between 0 and 1:

$$\Pr\{Y = 1\} = \Pr\{\text{having a good risk}\} = f\left(X;\hat{\theta}\right)$$

At first sight, a natural cut-off is $s = 50\%$:

$$\begin{cases} S < 50\% \Longrightarrow Y = 0 \Longrightarrow \text{reject} \\ S \geq 50\% \Longrightarrow Y = 1 \Longrightarrow \text{accept} \end{cases}$$

However, we will see in Section 15.3 on page 1008 that $s = 50\%$ is not necessarily the optimal cut-off, in particular when the population is heterogenous[10]. Moreover, the decision rule may be influenced by other factors that are not driven by s statistical point of view. For example, if the loss associated to the selection of a bad risk is larger than the gain associated to the selection of a bad risk, the optimal cut-off may be larger than 50%.

15.1.3.3 Score follow-up

Once we have built a scoring system, we begin to collect new information about the selected applicants. We can then backtest the score in order to check its robustness. Let us consider a rating system, whose annual probability of default is given by the following table:

Rating	A	B	C	D	E	F
Probability of default	0.5%	1%	2%	5%	15%	25%

Each year, we can calculate for each grade the default frequency and adjust the decision rule in order to obtain a coherent scoring system. Below, we have reported two examples of default frequencies:

Rating	A	B	C	D	E	F
Year 1	0.05%	2.3%	2.8%	7.5%	22.6%	35.1%
Year 2	0.5%	2.7%	1.3%	2.0%	15.1%	25.1%

It is obvious that Year 2 produces closer figures to the expected result than Year 1. However, Year 2 raises more concerns than Year 1 in terms of coherency. Indeed, the default frequencies are not increasing between ratings B, C and D. On the contrary, we observe a coherent ranking for Year 1, which faces an average default rate larger than predicted.

Besides the coherency issue, the stability of the scoring system is another important key element of the follow-up. Two axes of analysis can be conducted. The first one concerns the structure of the population with respect to the score. In the table below, we report the observed frequencies of each class:

Rating	A	B	C	D	E	F
Year 0	25%	20%	20%	20%	10%	5%
Year 1	15%	20%	25%	17%	13%	10%
Year 2	15%	15%	30%	15%	15%	10%

We notice a change in the population distribution, implying that the original scoring system may be no longer valid. The second axis of analysis concerns the exogenous variables that compose the score. In this case, the analysis consists in comparing the structure of the population with respect to each variable.

Another issue is the status of the rejected applicants. Indeed, there is an asymmetry between applicants that are accepted for credit and the others. We know what accepted applicants will become in terms of good/bad risk, but we don't know what the good/bad status of rejected applicants would have been (Hand and Henley, 1997):

[10]For example when the number of good risks is larger than the number of bad risks.

> "*The behavior of those who have been rejected, if instead they had been accepted, is unknown. If one estimates a model using data only on accepted applicants, those estimated parameters may be biased when applied to all applicants. In addition, if cut-off scores are chosen to equalize the actual and predicted number of defaulting applicants then a sample of accepted applicants is likely to yield inappropriate cut-offs for the population of all applicants*" (Crook and Banasik, 2004, page 857).

The statistical study of these rejected applicants is called '*reject inference*' and can be viewed as a missing data problem[11] (Little and Rubin, 2014). Except when selected and rejected populations are perfectly coherent with the scoring decision rule, the fact that we do not observe the rejected population introduces a bias. Let us consider the example of a tight decision rule implying that we never observe a bad risk. It is obvious that the calibrated statistical model does not reflect the entire population, but only the selected population. The issue is even high because a credit scoring model does not reduce to a statistical problem, but it is used from a business point of view. Questions about the market share and the other competitors are also essential. We have reported below an illustration:

Choice	Number of selected applicants	Default rate	Total profit (in $ mn)	Per-unit profit (in $)
#1	1 000 000	5%	100	100
#2	2 000 000	7%	150	75
#3	5 000 000	10%	180	36

What is the optimal choice? If the goal is to minimize the default rate, the best choice is #1. If the goal is to maximize the total profit, the third choice is optimal. There are several statistical approaches to perform reject inference (extrapolation, augmentation, reweighting, reclassification, etc.). However, they are not satisfactory because they focus on the default rate and ignore business issues. Nevertheless, they can help to test if the credit scoring model is biased (Banasik and Crook, 2007).

15.2 Statistical methods

Unsupervised learning is a branch of statistical learning, where test data does not include a response variable. It is opposed to supervised learning, whose goal is to predict the value of the response variable Y given a set of explanatory variables X. In the case of unsupervised learning, we only know the X-values, because the Y-values do not exist or are not observed. Supervised and unsupervised learning are also called '*learning with/without a teacher*' (Hastie *et al.*, 2009). This metaphor means that we have access to the correct answer provided by the supervisor (or the teacher) in supervised learning. In the case of unsupervised learning, we have no feedback on the correct answer. For instance, the linear regression is a typical supervised learning model, whereas the principal component analysis is an approach of unsupervised learning.

[11] We generally distinguish three types of missing value problems: missing completely at random or MCAR, missing at random or MAR, and missing not at random or MNAR. Credit scoring models generally face MAR or MNAR situation.

15.2.1 Unsupervised learning

In the following paragraphs, we focus on cluster analysis and dimension reduction, which are two unsupervised approaches for detecting commonalities in data.

15.2.1.1 Clustering

Cluster analysis is a method for the assignment of observations into groups or clusters. It is then an exploratory data analysis which allows to group similar observations together. As a result, the objective of clustering methods is to maximize the proximity between observations of a same cluster and to maximize the dissimilarity between observations which belong to different clusters. In what follows, we consider two popular cluster methods: K-means and hierarchical clustering.

K-means clustering It is a special case of combinatorial algorithms. This kind of algorithm does not use a probability distribution but works directly on observed data. We consider n observations with K attributes $x_{i,k}$ ($i = 1, \ldots, n$ and $k = 1, \ldots, K$). We note x_i the $K \times 1$ vector $(x_{i,1}, \ldots, x_{i,K})$. We would like to build n_C clusters $\mathcal{C}_j$ defined by the index j where $j = 1, \ldots, n_C$ with the following properties:

1. clusters must be disjoint: $\mathcal{C}_j \cap \mathcal{C}_{j'} = \emptyset$ for $j \neq j'$;
2. clusters must describe the entire dataset: $\mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_{n_C} = \{1, \ldots, n\}$;
3. observations assigned to a cluster are statistically similar.

Let $\mathcal{C}$ be the mapping function which permits to assign an observation to a cluster, meaning that $\mathcal{C}(i) = j$ assigns the i^{th} observation to the j^{th} cluster $\mathcal{C}_j$ – j is also called the corresponding label. The principle of combinatorial algorithms is to adjust the mapping function $\mathcal{C}$ in order to minimize the following loss function (Hastie *et al.*, 2009):

$$\mathcal{L}(\mathcal{C}) = \frac{1}{2} \sum_{j=1}^{n_C} \sum_{\mathcal{C}(i)=j} \sum_{\mathcal{C}(i')=j} d(x_i, x_{i'})$$

where $d(x_i, x_{i'})$ is the dissimilarity measure between the observations i and i'. As a result, the optimal mapping function is denoted $\mathcal{C}^\star = \arg\min \mathcal{L}(\mathcal{C})$.

In the case of the K-means algorithm, the dissimilarity measure is the Frobenius distance (or Euclidean norm):

$$d(x_i, x_{i'}) = \sum_{k=1}^{K} (x_{i,k} - x_{i',k})^2 = \|x_i - x_{i'}\|^2$$

Therefore, the loss function becomes[12]:

$$\mathcal{L}(\mathcal{C}) = \sum_{j=1}^{n_C} n_j \sum_{\mathcal{C}(i)=j} \|x_i - \bar{x}_j\|^2$$

[12] In Exercise 15.4.3 on page 1023, we show that:

$$\sum_{\mathcal{C}(i)=j} \frac{1}{2} \sum_{\mathcal{C}(i')=j} \|x_i - x_{i'}\|^2 = \sum_{\mathcal{C}(i)=j} n_j \|x_i - \bar{x}_j\|^2$$

where $\bar{x}_j = (\bar{x}_{1,j}, ..., \bar{x}_{K,j})$ is the $(K \times 1)$ mean vector associated with the j^{th} cluster and $n_j = \sum_{i=1}^{n} \mathbb{1}\{\mathcal{C}(i) = j\}$ is the corresponding number of observations. If we note $\mu_j^\star = \arg\min \sum_{\mathcal{C}(i)=j} \|x_i - \mu_j\|^2$, the previous minimization problem is equivalent to:

$$\left\{\mathcal{C}^\star, \mu_1^\star, \ldots, \mu_{n_C}^\star\right\} = \arg\min \sum_{j=1}^{n_C} n_j \sum_{\mathcal{C}(i)=j} \|x_i - \mu_j\|^2$$

where μ_j is called the centroid of cluster $\mathcal{C}_j$. This minimization problem may be solved by the Lloyd's iterative algorithm:

1. we initialize cluster centroids $\mu_1^{(0)}, \ldots, \mu_{n_C}^{(0)}$;
2. at the iteration s, we update the mapping function $\mathcal{C}^{(s)}$ using the following rule:

$$\mathcal{C}^{(s)}(i) = \arg\min_j \left\| x_i - \mu_j^{(s-1)} \right\|^2$$

3. we then compute the optimal centroids of the clusters $\left\{\mu_1^{(s)}, \ldots, \mu_{n_C}^{(s)}\right\}$:

$$\mu_j^{(s)} = \frac{1}{n_j} \sum_{\mathcal{C}^{(s)}(i)=j} x_i$$

4. we repeat steps 2 and 3 until convergence, that is when the assignments do not change: $\mathcal{C}^\star = \mathcal{C}^{(s)} = \mathcal{C}^{(s-1)}$.

We can show that the algorithm converges to a local minimum, implying that the main issue is to determine if the solution is also a global minimum. The answer depends on the initial choice of centroids. Generally, the algorithm is initialized with random centroids. In this case, we can run the algorithm many times and choose the clusters that give the smallest value of the function $\mathcal{L}(\mathcal{C})$. We also notice that the number of clusters is an hyperparameter of the clustering model[13]. This implies that we have to test different values of n_C in order to find the '*optimal*' partition.

TABLE 15.6: Data of the clustering problem

i	X_1	X_2	X_3	X_4	X_5
1	17.6	19.6	19.8	20.4	28.8
2	13.2	17.5	17.5	17.4	24.2
3	35.9	25.4	32.4	25.0	40.7
4	28.1	24.0	25.1	28.7	26.7
5	23.5	23.6	23.7	14.3	18.1
6	36.5	30.3	29.5	32.0	29.5
7	14.0	23.9	18.3	19.2	17.2
8	36.7	29.0	30.3	21.1	28.7
9	31.2	19.4	29.9	33.3	23.8
10	17.0	20.5	23.8	16.0	19.7

[13]Originally, the K-means method defines K clusters by their means (or centroids). In this book, K is the number of explanatory variables. This is why we prefer to use the notation j for cluster labeling, while n_C represents the number of classes.

TABLE 15.7: Optimal centroids $\mu_j^\star$ for 2 and 3 clusters

$\mu_j^\star$	X_1	X_2	X_3	X_4	X_5
			$n_C = 2$		
$\mu_1^\star$	17.06	21.02	20.62	17.46	21.60
$\mu_2^\star$	33.68	25.62	29.44	28.02	29.88
			$n_C = 3$		
$\mu_1^\star$	17.06	21.02	20.62	17.46	21.60
$\mu_2^\star$	36.37	28.23	30.73	26.03	32.97
$\mu_3^\star$	29.65	21.70	27.50	31.00	25.25

Example 166 *We consider the clustering problem of* 10 *observations with five variables* X_1 *to* X_5. *The data are reported in Table 15.6. We would like to know if two clusters are sufficient or if we need more clusters to analyze the similarity.*

By setting n_C equal to 2, we obtain the following optimal clustering: $\mathcal{C}_1^\star = \{1, 2, 5, 7, 10\}$ and $\mathcal{C}_2^\star = \{3, 4, 6, 8, 9\}$. Optimal centroids are reported in Table 15.7. It follows that $\mathcal{L}(\mathcal{C}_1^\star, \mathcal{C}_2^\star) = 3\,390.32$. In the case $n_C = 3$, the optimal clustering becomes $\mathcal{C}_1^\star = \{1, 2, 5, 7, 10\}$, $\mathcal{C}_2^\star = \{3, 6, 8\}$ and $\mathcal{C}_3^\star = \{4, 9\}$ while the loss function $\mathcal{L}(\mathcal{C}_1^\star, \mathcal{C}_2^\star, \mathcal{C}_3^\star)$ is equal to $1\,832.94$. We notice that the K-means algorithm has split the second cluster into two new clusters. The loss function does not help to determine the optimal number of clusters, because $\mathcal{L}(\mathcal{C})$ tends to zero when n_C increases. The most popular approach is the Elbow method, which consists in drawing the percentage of variance explained as a function of the number of clusters and detecting when the marginal gain is small. However, there is no good solution to estimate n_C, because they generally overestimate the number of clusters[14]. This is why it is better to fix the minimum number of observations by cluster. It is obvious that two clusters are sufficient in our example, because $n_C = 3$ leads to having a cluster with only two observations.

Hierarchical clustering The K-means clustering method presents several weak points. First, it requires many iterations when the number of observations and the number of clusters are large. Second, the solution highly depends on the cluster initialization, implying that we need to run many times the Lloyd's algorithm in order to find the optimal clustering. Third, the number of clusters is definitively an issue.

The idea of hierarchical clustering is to create a tree structure in order to model the relationships between the different clusters. Unlike the K-means algorithm, this algorithm does not depend on the number of clusters or the initialization assignment. However, it depends on the dissimilarity measure between two clusters. In Figure 15.7, we have represented an example of tree structure (or dendrogram) obtained by hierarchical clustering. The 1st and 3rd observations are grouped in order to obtain a first cluster. This cluster is then merged with the 5th observation in order to define a new cluster. In a similar way, the 6th and 7th observations are grouped to obtain a first cluster. This cluster is then merged with the 10th observation in order to define a new cluster. The tree structure indicates how two clusters are merged into a new single cluster. The lowest level of the tree corresponds to the individual observations. In this case, each cluster contains one observation. The highest level of the tree corresponds to the entire dataset. In this case, there is only one cluster that contains all the observations.

[14]For instance, we obtain $\mathcal{C}_1^\star = \{1, 2\}$, $\mathcal{C}_2^\star = \{5, 7, 10\}$, $\mathcal{C}_3^\star = \{3, 6, 8\}$ and $\mathcal{C}_4^\star = \{4, 9\}$ when $n_C = 4$.

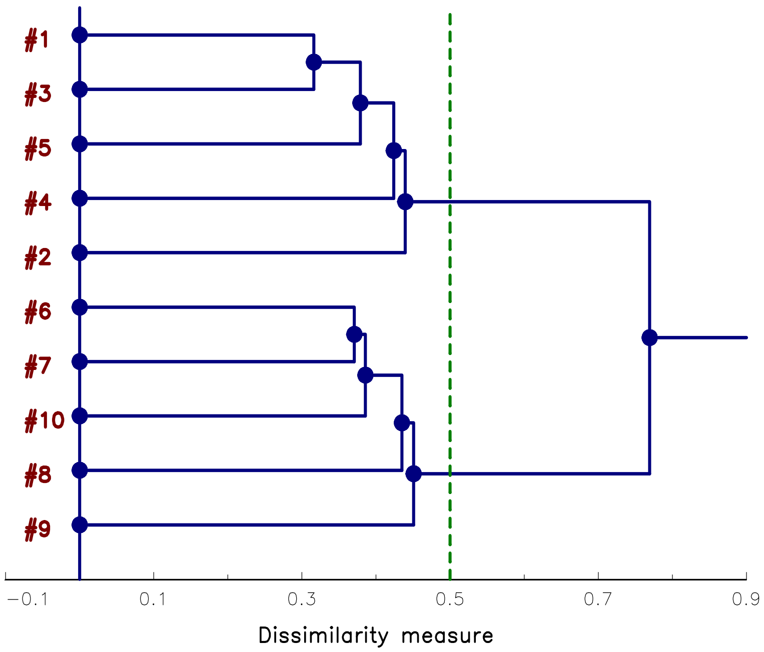

FIGURE 15.7: An example of dendrogram

Remark 181 *There is a difference between a basic tree and a dendrogram. Indeed, the x-axis of the dendrogram corresponds to the dissimilarity measure. Therefore, we can easily see which merge creates small or large dissimilarity.*

We generally distinguish two approaches for hierarchical clustering:

- in the agglomerative method (also called bottom-up clustering), the algorithm starts with the individual clusters and recursively merge the closest pairs of clusters into one single cluster;
- in the divisive method (also called the top-down clustering), the algorithm starts with the single cluster containing all the observations and recursively splits a cluster into two new clusters, which present the maximum dissimilarity.

Let $\mathcal{C}_j$ and $\mathcal{C}_{j'}$ be two clusters. The objective function of the agglomerative method is to minimize the dissimilarity measure $D(\mathcal{C}_j, \mathcal{C}_{j'})$ while we maximize the dissimilarity measure $D(\mathcal{C}_j, \mathcal{C}_{j'})$ in the divisive method. In what follows, we only consider the agglomerative method, because it is more efficient in terms of computational time and it is more widespread used.

The dissimilarity measure $D(\mathcal{C}_j, \mathcal{C}_{j'})$ is defined as a linkage function of pairwise dissimilarities $d(x_i, x_{i'})$ where $\mathcal{C}(i) = j$ and $\mathcal{C}(i') = j'$. Therefore, the agglomerative method requires defining two dissimilarity measures: the linkage function between two clusters $D(\mathcal{C}_j, \mathcal{C}_{j'})$ and the distance between two observations $d(x_i, x_{i'})$. For this last one, we generally consider the Mahalanobis distance:

$$d(x_i, x_{i'}) = \sqrt{(x_i - x_{i'})^\top \hat{\Sigma} (x_i - x_{i'})}$$

where $\hat{\Sigma}$ is the sample covariance matrix or the Minkowski distance:

$$d(x_i, x_{i'}) = \left(\sum_{k=1}^{K} |x_{i,k} - x_{i',k}|^p\right)^{1/p}$$

where $p > 1$. The case $p = 2$ corresponds to the Euclidean distance. For the linkage function, we generally consider three approaches. The single linkage (or nearest neighbor) is the smallest distance between the clusters:

$$D(\mathcal{C}_j, \mathcal{C}_{j'}) = \min_{\{\mathcal{C}(i)=j, \mathcal{C}(i')=j'\}} d(x_i, x_{i'})$$

The complete linkage (or furthest neighbor) is the largest distance between the clusters:

$$D(\mathcal{C}_j, \mathcal{C}_{j'}) = \max_{\{\mathcal{C}(i)=j, \mathcal{C}(i')=j'\}} d(x_i, x_{i'})$$

Finally, the average linkage is the average distance between the clusters:

$$D(\mathcal{C}_j, \mathcal{C}_{j'}) = \frac{1}{n_j n_{j'}} \sum_{\mathcal{C}(i)=j} \sum_{\mathcal{C}(i')=j'} d(x_i, x_{i'})$$

At each iteration, we search the clusters j and j' which minimize the dissimilarity measure and we merge them into one single cluster. When we have merged all the observations into one single cluster, the algorithm is stopped. It is also easy to perform a segmentation by considering a particular level of the tree. Indeed, we notice that the algorithm exactly requires $n - 1$ iterations. The level $L^{(s)} = s$ is then associated to the s^{th} iteration and we note $D^{(s)} = D(\mathcal{C}_{j^\star}, \mathcal{C}_{j'^\star})$ the minimum value of $D(\mathcal{C}_j, \mathcal{C}_{j'})$.

In Figure 15.7, the dendrogram was based on simulated data using the single linkage rule and the Euclidean distance. We have considered 10 observations divided into two groups. The attributes of the first (resp. second) one correspond to simulated Gaussian variates with a mean of 20% (resp. 30%) and a standard deviation of 5% (resp. 5%). The intra-group cross-correlation is set to 80% whereas the inter-group correlation is equal to 0%. We obtain satisfactory results. Indeed, if we would like to consider two clusters, the first cluster is composed of the first five observations, whereas the second cluster is composed of the last five observations. In practice, hierarchical clustering may produce concentrated segmentation as illustrated in Figure 15.8. We use the same simulated data as previously except that the standard deviation for the second group is set to 25%. In this case, if we would like to consider two clusters, we obtain a cluster with 9 elements and another cluster with only one element (the 6th observation).

Let us consider Example 166 on page 946. By using the Euclidean distance, we obtain the dendrograms in Figure 15.9. If we would like to split the data into two clusters, we find for the three methods the solution $\{1, 2, 5, 7, 10\}$ and $\{3, 4, 6, 8, 9\}$, which also corresponds to the solution given by the K-means analysis. In the case of the single linkage method, we have reported in Table 15.8 for each level $L^{(s)}$ the distance $D^{(s)}$, the two nearest neighbours $i^\star$ and $i'^\star$ and the created cluster $\mathcal{C}^{(s)}$. We notice that the solution for 3 clusters differs from the K-means solution. Indeed, we find $\{1, 2, 5, 7, 10\}$, $\{4, 6, 8, 9\}$ and $\{3\}$ for the single linkage method.

15.2.1.2 Dimension reduction

We now turn to the concept of dimension reduction, which consists in finding some common patterns in order to better explain the data. For instance, we might want to reduce

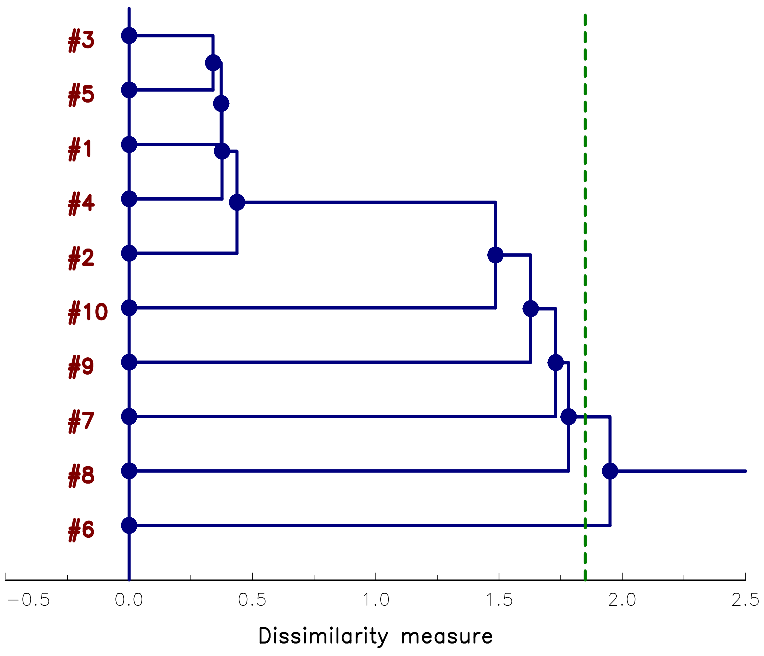

FIGURE 15.8: Unbalanced clustering

TABLE 15.8: Agglomerative hierarchical clustering (single linkage)

$L^{(s)}$	$D^{(s)}$	$(i^\star, i'^\star)$	$\mathcal{C}^{(s)}$
1	7.571	$(5,10)$	$\{5,10\}$
2	7.695	$(1,2)$	$\{1,2\}$
3	8.204	$(5,7)$	$\{5,7,10\}$
4	9.131	$(4,9)$	$\{4,9\}$
5	9.238	$(1,5)$	$\{1,2,5,7,10\}$
6	11.037	$(6,8)$	$\{6,8\}$
7	12.179	$(4,6)$	$\{4,6,8,9\}$
8	13.312	$(3,4)$	$\{3,4,6,8,9\}$
9	15.199	$(1,3)$	$\{1,2,3,4,5,6,7,8,9,10\}$

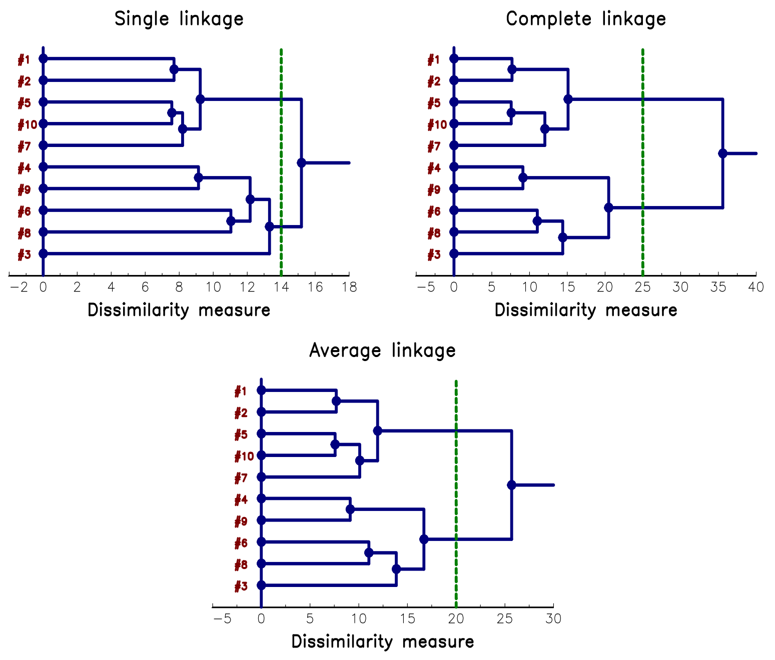

FIGURE 15.9: Comparison of the three dendrograms

a dataset with 1 000 variables to the two or three most important patterns. In machine learning, dimension reduction is also known as feature extraction, which is defined as the process to build new variables or features that are more informative and less redundant than the original variables.

Principal component analysis Let X be a $K \times 1$ random vector, whose covariance matrix is equal to Σ. We consider the linear transform $Z = B^\top X$ where $B = (\beta_1, \ldots, \beta_K)$ is a $K \times K$ matrix and the β's are $K \times 1$ vectors. The j^{th} element of Z is denoted by Z_j and we have $Z_j = \beta_j^\top X = \sum_{k=1}^{K} \beta_{k,j} X_k$. Z_j is also called the j^{th} principal component. The idea of PCA is to find a first linear function $\beta_1^\top X$ such that the variance of Z_1 is maximum and then a j^{th} linear function $\beta_j^\top X$ such that the variance of Z_j is maximum and Z_j is uncorrelated with $Z_1, \ldots, Z_{j-1}$ for all $j \geq 2$ (Jolliffe, 2002). We can show that B is the matrix of eigenvectors[15] of the covariance matrix Σ:

$$\Sigma B = B \Lambda$$

where $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_K)$ is the diagonal matrix of eigenvalues with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_K$. Since Σ is a symmetric and positive define matrix, we also have:

$$\Sigma = B \Lambda B^{-1} = B \Lambda B^\top$$

[15]See Exercise 15.4.4 on page 1024 for the derivation of this result.

and B is an orthonormal matrix. By construction, we have[16]:

$$\begin{aligned}\operatorname{var}(Z_j) &= \operatorname{var}\left(\beta_j^\top X\right) \\ &= \beta_j^\top \Sigma \beta_j \\ &= \Lambda_{j,j} \\ &= \lambda_j\end{aligned}$$

and:

$$\begin{aligned}\operatorname{cov}(Z_j, Z_{j'}) &= \beta_j^\top \Sigma \beta_{j'} \\ &= \Lambda_{j,j'} \\ &= 0\end{aligned}$$

We deduce the spectral decomposition of the covariance matrix:

$$\Sigma = B\Lambda B^\top = \sum_{j=1}^{K} \lambda_j \beta_j \beta_j^\top$$

We note $B_{(1:j)}$ and $B_{(K-j+1:K)}$ the matrices that contains the first and last j columns of B. We consider the random vector $\tilde{Z} = \left(\tilde{Z}_1, \ldots, \tilde{Z}_j\right) = \tilde{B}^\top X$ of dimension j. Here are some properties of the PCA (Jolliffe, 2002):

1. the trace of $\operatorname{cov}\left(\tilde{Z}\right)$ is maximized if $\tilde{B} = B_{(1:j)}$ corresponds to the j first eigenvectors;

2. the trace of $\operatorname{cov}\left(\tilde{Z}\right)$ is minimized if $\tilde{B} = B_{(K-j+1:K)}$ corresponds to the j last eigenvectors;

3. the covariance of X given $\tilde{Z}$ is:

$$\begin{aligned}\operatorname{cov}\left(X \mid \tilde{Z}_1, \ldots, \tilde{Z}_j\right) &= \Sigma_{X,X} - \Sigma_{X,\tilde{Z}} \Sigma_{\tilde{Z},\tilde{Z}}^{-1} \Sigma_{\tilde{Z},X} \\ &= \sum_{k=j+1}^{K} \lambda_k \beta_k \beta_k^\top\end{aligned}$$

4. we consider the following linear regression model:

$$X = A\tilde{Z} + U$$

where A is a $K \times j$ matrix and $U = (U_1, \ldots, U_K)$ is the vector of residuals; if we note $\Omega = \operatorname{diag}\left(\sigma_1^2, \ldots, \sigma_K^2\right)$ the covariance matrix of U, the trace of Ω is minimized if $\tilde{B} = B_{(1:j)}$.

Remark 182 *The principal component analysis can be performed with correlation matrices instead of covariance matrices. Jolliffe (2002) presents different arguments for justifying this choice. Indeed, PCA makes more sense when the variables are comparable. Otherwise, the principal components are dominated by the variables with the largest variances.*

[16]Because of the following equality:

$$\Lambda = B^{-1}\Sigma B = B^\top \Sigma B$$

Example 167 *We consider the random vector* $X = (X_1, X_2, X_3, X_4)$*, whose individual variances are equal to* 1, 2, 3 *and* 4*. The correlation matrix is:*

$$\rho = \begin{pmatrix} 1.00 & & & \\ 0.30 & 1.00 & & \\ 0.50 & 0.10 & 1.00 & \\ 0.20 & 0.50 & -0.50 & 1.00 \end{pmatrix}$$

We have reported the eigendecomposition of Σ and ρ in Tables 15.9 and 15.10. We observe some differences. For instance, the first principal component of the covariance matrix Σ is:

$$Z_1 = 0.18 \cdot X_1 + 0.33 \cdot X_2 + 0.53 \cdot X_3 + 0.76 \cdot X_4$$

whereas the first principal component of the correlation matrix ρ is:

$$Z_2 = 0.48 \cdot X_1 + 0.44 \cdot X_2 + 0.53 \cdot X_3 + 0.55 \cdot X_4$$

We verify that the sum of eigenvalues is equal to the sum of variances for the covariance matrix, and the number of variables for the correlation matrix[17].

TABLE 15.9: Eigendecomposition of the covariance matrix

	β_1	β_2	β_3	β_4
X_1	0.18	−0.20	−0.57	0.77
X_2	0.33	0.58	−0.63	−0.40
X_3	0.53	−0.73	−0.13	−0.41
X_4	0.76	0.31	0.50	0.27
λ_j	5.92	2.31	1.31	0.46

TABLE 15.10: Eigendecomposition of the correlation matrix

	β_1	β_2	β_3	β_4
X_1	0.48	−0.44	−0.65	−0.40
X_2	0.44	0.67	−0.40	0.45
X_3	0.53	−0.51	0.38	0.57
X_4	0.55	0.33	0.53	−0.56
λ_j	2.06	0.97	0.73	0.23

We now develop the interpretation tools of PCA. The quality of representation is defined as the percentage of total variance that is explained by the j^{th} principal component (or PC):

$$\mathbb{Q}_j = \frac{\lambda_j}{\sum_{k=1}^{K} \lambda_k}$$

We have $0 \leq \mathbb{Q}_j \leq 1$. The cumulative quality of representation is just the cumulative sum of the quality values:

$$\mathbb{Q}_j^{\star} = \sum_{k=1}^{j} \mathbb{Q}_k = \frac{\sum_{k=1}^{j} \lambda_k}{\sum_{k=1}^{K} \lambda_k}$$

[17]We have $\operatorname{trace}(\Sigma) = \sum_{k=1}^{K} \sigma_k^2$ and $\operatorname{trace}\left(B \Lambda B^{-1}\right) = \operatorname{trace}\left(\Lambda B^{-1} B\right) = \operatorname{trace}(\Lambda) = \sum_{j=1}^{K} \lambda_j$. For the correlation matrix, we deduce that $\sum_{j=1}^{K} \lambda_j = K$.

$\mathbb{Q}_j^{\star}$ is also called the quality of representation of the j^{th} principal plane[18]. The correlation between the variable X_k and the factor Z_j is given by:

$$\operatorname{cor}(X_k, Z_j) = \beta_{k,j}\sqrt{\lambda_j}$$

It follows that the quality of representation of the variable X_k with respect to the j^{th} PC is[19]:

$$\mathbb{Q}_{k,j} = \operatorname{cor}^2(X_k, Z_j) = \beta_{k,j}^2 \lambda_j$$

We can also define the contribution of the variable X_k to the j^{th} PC[20]:

$$\mathbb{C}_{k,j} = \beta_{k,j}^2$$

In order to the understand the association between variables, we generally plot the correlation circle between two principal components that corresponds to the scatterplot of $\operatorname{cor}(X_k, Z_j)$ and $\operatorname{cor}(X_k, Z_{j'})$.

Remark 183 *In practice, we estimate the covariance or the correlation matrix using a sample. Let $x_i = (x_{i,1}, \ldots, x_{i,K})$ be the i^{th} observation. We note $z_{i,j} = \beta_j^\top x_i$ the projection of x_i onto the j^{th} principal component. The quality of representation and the contribution of an observation to a principal component are then equal to:*

$$\mathbb{Q}_{i,j} = \frac{z_{i,j}^2}{\sum_{k=1}^K z_{i,k}^2}$$

and:

$$\mathbb{C}_{i,j} = \frac{z_{i,j}^2}{\sum_{i=1}^n z_{i,j}^2}$$

We consider again Example 166 on page 946. In Table 15.11, we have reported the results of the PCA applied to the correlation matrix of data. The first PC explains 68.35% of the variance, while the quality of representation of the second PC is equal to 14.54%. This means that we can explain 82.89% with only two factors. For the first factor, each variable has a positive loading. This is not the case of the second factor, where the factor loadings of X_1, X_2 and X_3 are negative. We notice that X_1 and X_3 are well represented by Z_1 (95.81% and 86.95%). For the second PC, the second variable X_2 is the most represented (41.40%). If we consider the last PC, the quality of representation is poor (less than 1%). This indicates that the last PC has a very low explanation power. We notice that the rationale of the fourth PC is to model X_3 because the second and third PCs do not explain this variable. The contribution values $\mathbb{C}_{k,j}$ are also interesting to confirm the previous results. For instance, X_1 does not contribute to Z_2. It follows that the second PC represents the opposition of X_2

[18]That is the plane composed of the first j principal components.

[19]We verify that the sum of $\mathbb{Q}_{k,j}$ is equal to the variance of the j^{th} PC:

$$\sum_{k=1}^K \mathbb{Q}_{k,j} = \sum_{k=1}^K \beta_{k,j}^2 \lambda_j = \lambda_j \sum_{k=1}^K \beta_{k,j}^2 = \lambda_j$$

[20]We verify that the sum of $\mathbb{C}_{k,j}$ is equal to 100%:

$$\sum_{k=1}^K \mathbb{C}_{k,j} = \sum_{k=1}^K \beta_{k,j}^2 = 1$$

TABLE 15.11: Principal component analysis of Example 166

Factor	Z_1	Z_2	Z_3	Z_4	Z_5
λ_j	3.4173	0.7271	0.5548	0.2783	0.0226
$\mathbb{Q}_j$	68.35%	14.54%	11.10%	5.57%	0.45%
$\mathbb{Q}_j^{\star}$	68.35%	82.89%	93.98%	99.55%	100.00%
Matrix B of eigenvectors					
X_1	0.5295	−0.1015	−0.0567	−0.2554	0.8006
X_2	0.3894	−0.7546	−0.0500	0.4855	−0.2019
X_3	0.5044	−0.0188	−0.0247	−0.6650	−0.5499
X_4	0.3952	0.5318	−0.6238	0.3995	−0.1107
X_5	0.3967	0.3702	0.7775	0.3120	−0.0609
Correlation between X_k and Z_j					
X_1	97.88%	−8.66%	−4.22%	−13.47%	12.03%
X_2	71.98%	−64.35%	−3.72%	25.61%	−3.03%
X_3	93.25%	−1.60%	−1.84%	−35.08%	−8.27%
X_4	73.06%	45.35%	−46.46%	21.07%	−1.66%
X_5	73.34%	31.57%	57.91%	16.46%	−0.92%
Quality of representation of each variable $\mathbb{Q}_{k,j}$					
X_1	95.81%	0.75%	0.18%	1.82%	1.45%
X_2	51.81%	41.40%	0.14%	6.56%	0.09%
X_3	86.95%	0.03%	0.03%	12.31%	0.68%
X_4	53.38%	20.57%	21.59%	4.44%	0.03%
X_5	53.78%	9.96%	33.54%	2.71%	0.01%
Contribution of each variable $\mathbb{C}_{k,j}$					
X_1	28.04%	1.03%	0.32%	6.52%	64.09%
X_2	15.16%	56.94%	0.25%	23.57%	4.08%
X_3	25.44%	0.04%	0.06%	44.22%	30.24%
X_4	15.62%	28.29%	38.91%	15.96%	1.23%
X_5	15.74%	13.70%	60.46%	9.73%	0.37%

with respect to X_4 and X_5. Clearly, the third PC mainly concerns X_4 and X_5. Figure 15.10 represents the scatterplot of the factor values $z_{i,j}$ for the first two principal components. We notice that the second component classifies the observations in the same way than the K-means algorithm or the agglomerative hierarchical clustering. Indeed, we retrieve the two clusters $\{1, 2, 5, 7, 10\}$ and $\{3, 4, 6, 8, 9\}$. This is not the case of the first component, which operates the following classification $\{1, 2, 3, 4, 9\}$ and $\{5, 6, 7, 8, 10\}$. In Figure 15.11, we have reported the correlation circle between different PCs. If we consider the first two PCs, the variables X_1 and X_2 are clearly opposed to the variables X_4 and X_5. The second panel confirms the competition between X_4 and X_5 due to the third PC.

Non-negative matrix factorization There are several alternative approaches to principal component analysis. For instance, independent component analysis (ICA) estimates additive factors that are maximally independent. Another popular method is the non-negative matrix factorization (NMF). Let A be a non-negative matrix $m \times p$. We define the NMF decomposition of A as follows:

$$A \approx BC$$

where B and C are two non-negative matrices with respective dimensions $m \times n$ and $n \times p$. Compared to classic decomposition algorithms, we remark that BC is an approximation

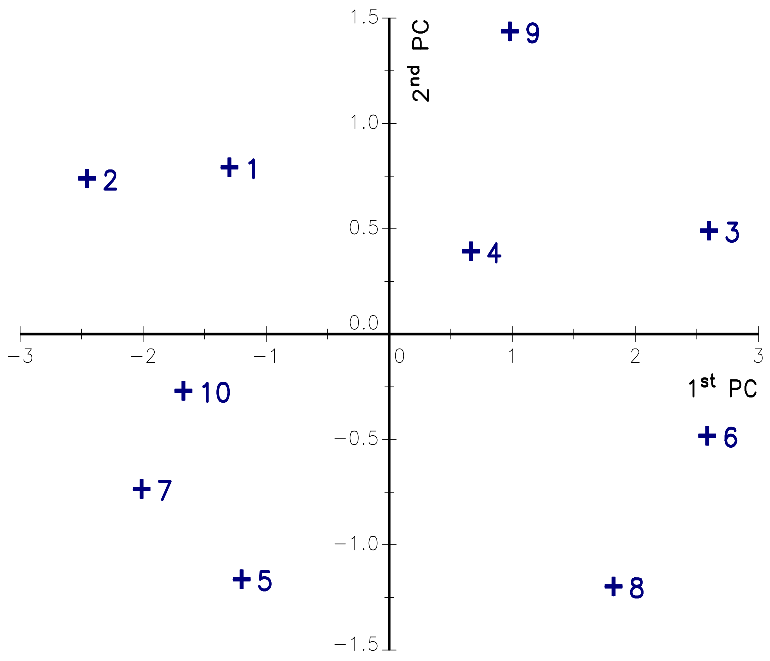

FIGURE 15.10: Scatterplot of the factor values $z_{i,1}$ and $z_{i,2}$

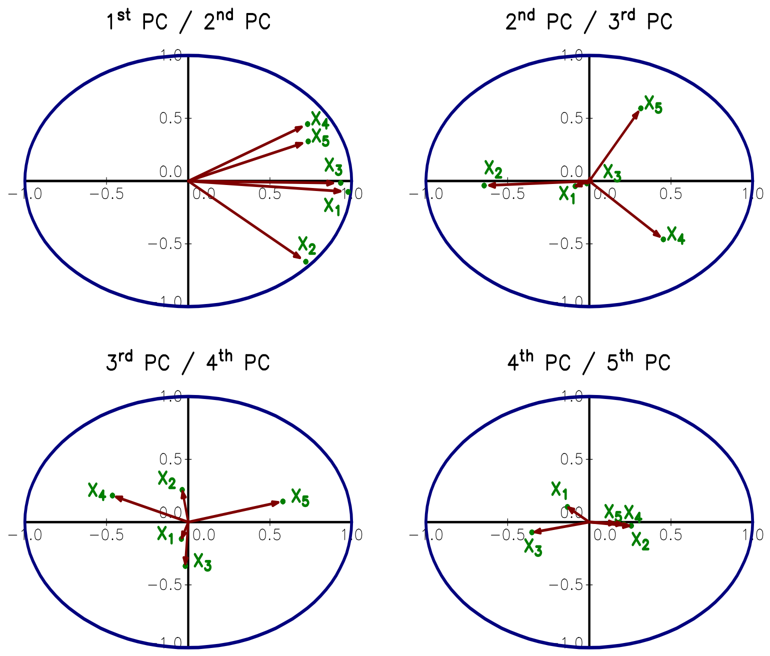

FIGURE 15.11: PCA correlation circle

of A. There are also different ways to obtain this approximation meaning that B and C are not necessarily unique[21]. We also notice that the decomposition $A \approx BC$ is equivalent to $A^\top \approx C^\top B^\top$. It means that the storage of the data is not important. Rows of A may represent either the observations or the variables, but the interpretation of the B and C matrices depend on the choice of the storage. We remark that:

$$A_{i,j} = \sum_{k=1}^{n} B_{i,k} C_{k,j}$$

Suppose that we consider a variable/observation storage. Therefore, $B_{i,k}$ depends on the variable i whereas $C_{k,j}$ depends on the observation j. In this case, we may interpret B as a matrix of weights. In factor analysis, B is called the loading matrix and C is the factor matrix. $B_{i,k}$ is then the weight of factor k for variable i and $C_{k,j}$ is the value taken by factor k for observation j. If we use an observation/variable storage which is the common way to store data in statistics, B and C become the factor matrix and the loading matrix.

Because the dimensions m, n and p may be very large, one of the difficulties with NMF is to derive a numerical algorithm with a reasonable computational time. Lee and Seung (1999) developed a simple algorithm with strong performance and applied it to pattern recognition with success. Since this seminal work, this algorithm has been improved and there are today several ways to obtain a non-negative matrix factorization. In order to find the approximate factorization, we need to define the loss function $\mathcal{L}$ which measures the quality of the factorization. The optimization program is then:

$$\begin{aligned} \{B^\star, C^\star\} &= \arg\min \mathcal{L}(A, BC) \\ &\text{u.c.} \quad \left\{ \begin{array}{l} B \succ \mathbf{0} \\ C \succ \mathbf{0} \end{array} \right. \end{aligned} \tag{15.6}$$

Lee and Seung (2001) considered two loss functions. The first one is the Frobenious norm:

$$\mathcal{L}(A, BC) = \sum_{i=1}^{m} \sum_{j=1}^{p} \left(A_{i,j} - (BC)_{i,j} \right)^2$$

whereas the second one is Kullback-Leibler divergence:

$$\mathcal{L}(A, BC) = \sum_{i=1}^{m} \sum_{j=1}^{p} \left(A_{i,j} \ln \frac{A_{i,j}}{(BC)_{i,j}} - A_{i,j} + (BC)_{i,j} \right)$$

To solve Problem (15.6), Lee and Seung (2001) proposed to use the multiplicative update algorithm. Let $B_{(s)}$ and $C_{(s)}$ be the matrices at iteration s. For the Frobenious norm[22], we have:

$$\left\{ \begin{array}{l} B_{(s+1)} = B_{(s)} \odot \left(A C_{(s)}^\top \right) \oslash \left(B_{(s)} C_{(s)} C_{(s)}^\top \right) \\ C_{(s+1)} = C_{(s)} \odot \left(B_{(s+1)}^\top A \right) \oslash \left(B_{(s+1)}^\top B_{(s+1)} C_{(s)} \right) \end{array} \right.$$

[21]Let D be a nonnegative matrix such that D^{-1} is nonnegative too. For example, D may be a permutation of a diagonal matrix. In this case, we have:

$$A \approx BD^{-1}DC \approx B'C'$$

where $B' = BD^{-1}$ and $C' = DC$ are two nonnegative matrices. This shows that the decomposition is not unique.

[22]A similar algorithm may be derived for the Kullback-Leibler divergence.

where $\odot$ and $\oslash$ are respectively the element-wise multiplication and division operators. Under some assumption, we may show that $B^{\star} = B_{(\infty)}$ and $C^{\star} = C_{(\infty)}$, meaning that the multiplicative update algorithm converges to the optimal solution.

For large datasets, the computational time to find the optimal solution may be large with the previous algorithm. Since the seminal work of Lee and Seung, a lot of methods have also been proposed to improve the multiplicative update algorithm and speed the converge. Among these methods, we may mention the algorithm developed by Lin (2007), which is based on the alternating non-negative least squares:

$$\left\{ \begin{array}{l} B_{(s+1)} = \arg\min \mathcal{L}\left(A, BC_{(s)}\right) \\ C_{(s+1)} = \arg\min \mathcal{L}\left(A, B_{(s+1)}C\right) \end{array} \right. \tag{15.7}$$

with the constraints $B_{(s+1)} \succ \mathbf{0}$ and $C_{(s+1)} \succ \mathbf{0}$. We notice that the two optimization problems (15.7) are symmetric because we may cast the first problem in the form of the second problem: $B_{(s+1)}^{\top} = \arg\min \mathcal{L}\left(A^{\top}, C_{(s)}^{\top} B^{\top}\right)$. So, we may only focus on the following optimization problem:

$$\begin{aligned} C^{\star} &= \arg\min \mathcal{L}\left(A, BC\right) \\ \text{u.c.} \quad & C \succ \mathbf{0} \end{aligned}$$

In the case of the Frobenious norm, we have $\partial_C \mathcal{L}\left(A, BC\right) = 2B^{\top}\left(BC - A\right)$. The projected gradient method consists in the following iterating scheme:

$$C \leftarrow C - \alpha \cdot \frac{\partial \mathcal{L}\left(A, BC\right)}{\partial C}$$

where α is the descent length. Let (β, γ) be two scalars in $]0,1[$. Instead of finding the optimal value of α at each iteration, Lin (2007) proposed to update α in a very simple way depending on the inequality equation:

$$(1-\gamma)\frac{\partial \mathcal{L}\left(A, BC\right)}{\partial C}^{\top}\left(\tilde{C} - C\right) + \frac{1}{2}\left(\tilde{C} - C\right)^{\top}\frac{\partial^2 \mathcal{L}\left(A, BC\right)}{\partial C\,\partial C^{\top}}\left(\tilde{C} - C\right) \leq 0$$

where $\tilde{C}$ is the update of C. If this inequality equation is verified, α is increased ($\alpha \leftarrow \alpha/\beta$), otherwise α is decreased ($\alpha \leftarrow \alpha\beta$).

Remark 184 *The choice of $B_{(0)}$ and $C_{(0)}$ for initializing NMF algorithms is important. The random method consists in generating matrices with positive random numbers*[23]*. Another popular approach is the non-negative double singular value decomposition, which is a modification of the singular value decomposition by considering only the non-negative part of the singular values (Boutsidis and Gallopoulos, 2008).*

In order to understand why NMF is different from other factor methods, we consider a simulation study. We consider a basket of four financial assets. The asset prices are driven by a multidimensional geometric Brownian motion. The drift parameter is equal to 5% whereas the diffusion parameter is 20%. The cross-correlation $\rho_{i,j}$ between assets i and j is equal to 20%, but $\rho_{1,2} = 70\%$ and $\rho_{3,4} = 50\%$. In order to preserve the time homogeneity, the data correspond to $x_{i,t} = \ln S_{i,t}$ where $S_{i,t}$ is the price of the asset i at time t. In Figure 15.12, we report the time series $x_{i,t}$ for the four assets (panel 1) and, the first factor estimated

[23]For example, we can use the probability distributions $\mathcal{U}_{[0,1]}$ or $\left|\mathcal{N}\left(0,1\right)\right|$.

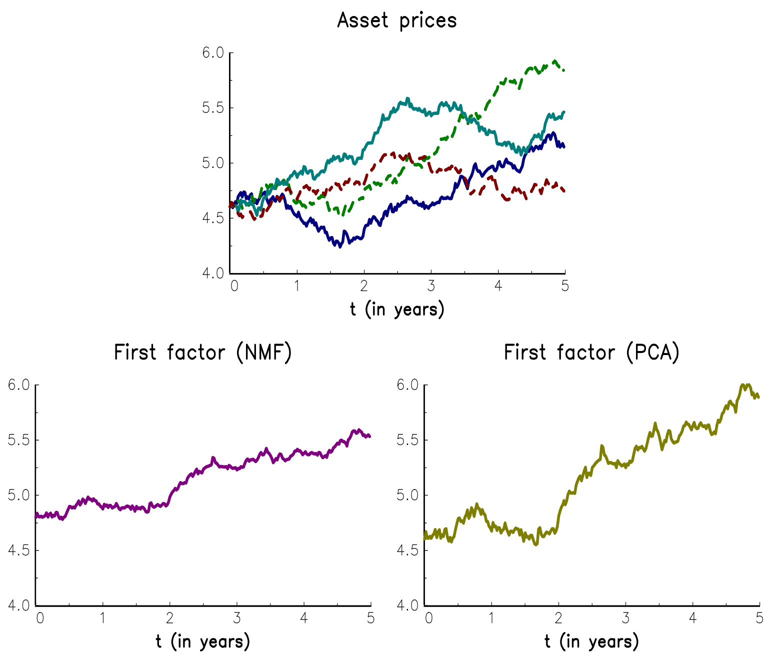

FIGURE 15.12: Estimating the first factor of a basket of financial assets

by NMF[24] (panel 2) and PCA (panel 3). We notice that the NMF factor[25] is not scaled in the same way than the PCA factor. However, the correlation between the first differences is equal to 98.8%. In the first panel in Figure 15.13, we compare the decomposition of the variance according to the factors. We notice that PCA explains more variance than NMF for a given number of factors. We obtain this result because NMF may be viewed as a constrained principal component analysis with nonnegative matrices. However, it does not mean that each PCA factor explains more variance than the corresponding NMF factor. For example, the second NMF factor explains more variance than the second PCA factor in Figure 15.13. In the other panels, we compare the dynamics of the first asset with the dynamics given by the NMF factors[26]. With three risk factors, the reconstructed signal has a correlation of 93.7% with the original signal.

15.2.2 Parametric supervised methods

15.2.2.1 Discriminant analysis

Discriminant analysis was first developed by Fisher (1936). This approach is close to the principal component analysis (PCA) and is used to predict class membership for independent variables. For that, we assume that we have n_C disjoint classes $\mathcal{C}_j$ where $j = 1, \ldots, J$. Discriminant analysis consists then in assigning an observation to one and only one class.

[24]The NMF decomposition corresponds to:

$$\underbrace{\ln X}_{n_X \times n_T} \approx \underbrace{B}_{n_X \times n_F} \cdot \underbrace{C}_{n_F \times n_T}$$

where n_X is the number of time-series, n_T is the number of dates and n_F is the number of NMF factors.

[25]In this example, B is the loading matrix while C is the matrix of time-series factors.

[26]The reconstructed multidimensional signal is just the matrix product BC for different values of n_F.

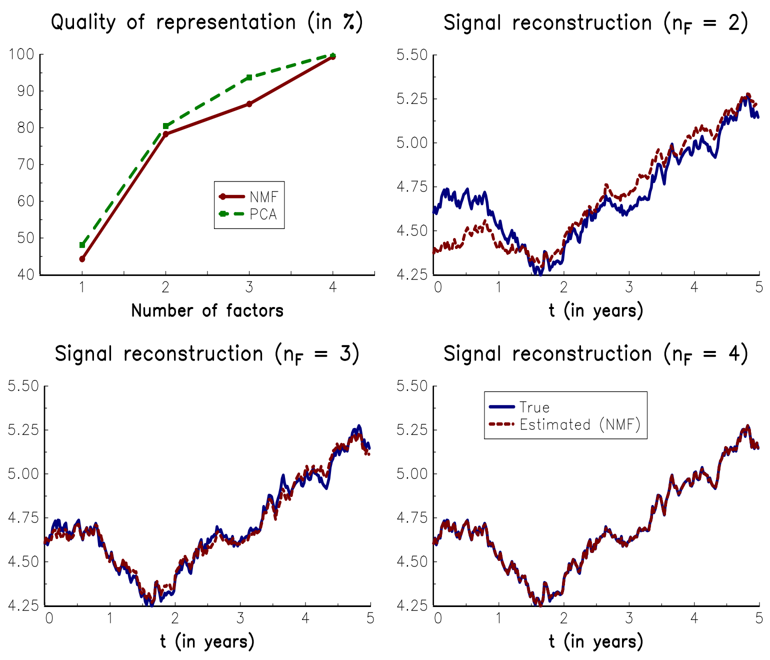

FIGURE 15.13: Variance decomposition and signal reconstruction

We consider an input vector x and we divide the input space into n_C decision regions, whose boundaries are called decision boundaries (Bishop, 2006). Classification methods can then be seen as a supervised clustering methods, where the categorical response variable is directly the class. For example, Figure 15.14 corresponds to a classification problem with seven classes and two explanatory variables X_1 and X_2. The goal is then to predict for each observation its class. For instance, we would like that the model predicts that the first observation belongs to the first class, the second observation belongs to the fifth class, etc.

The two-dimensional case Using the Bayes theorem, we have:

$$\begin{aligned}\Pr\{A \cap B\} &= \Pr\{A \mid B\} \cdot \Pr\{B\} \\ &= \Pr\{B \mid A\} \cdot \Pr\{A\}\end{aligned}$$

It follows that:

$$\Pr\{A \mid B\} = \Pr\{B \mid A\} \cdot \frac{\Pr\{A\}}{\Pr\{B\}}$$

If we apply this result to the conditional probability $\Pr\{i \in \mathcal{C}_1 \mid X = x\}$, we obtain:

$$\Pr\{i \in \mathcal{C}_1 \mid X = x\} = \Pr\{X = x \mid i \in \mathcal{C}_1\} \cdot \frac{\Pr\{i \in \mathcal{C}_1\}}{\Pr\{X = x\}}$$

The log-probability ratio is then equal to:

$$\begin{aligned}\ln \frac{\Pr\{i \in \mathcal{C}_1 \mid X = x\}}{\Pr\{i \in \mathcal{C}_2 \mid X = x\}} &= \ln\left(\frac{\Pr\{X = x \mid i \in \mathcal{C}_1\}}{\Pr\{X = x \mid i \in \mathcal{C}_2\}} \cdot \frac{\Pr\{i \in \mathcal{C}_1\}}{\Pr\{i \in \mathcal{C}_2\}}\right) \\ &= \ln \frac{\Pr\{X = x \mid i \in \mathcal{C}_1\}}{\Pr\{X = x \mid i \in \mathcal{C}_2\}} + \ln \frac{\Pr\{i \in \mathcal{C}_1\}}{\Pr\{i \in \mathcal{C}_2\}} \\ &= \ln \frac{f_1(x)}{f_2(x)} + \ln \frac{\pi_1}{\pi_2}\end{aligned}$$

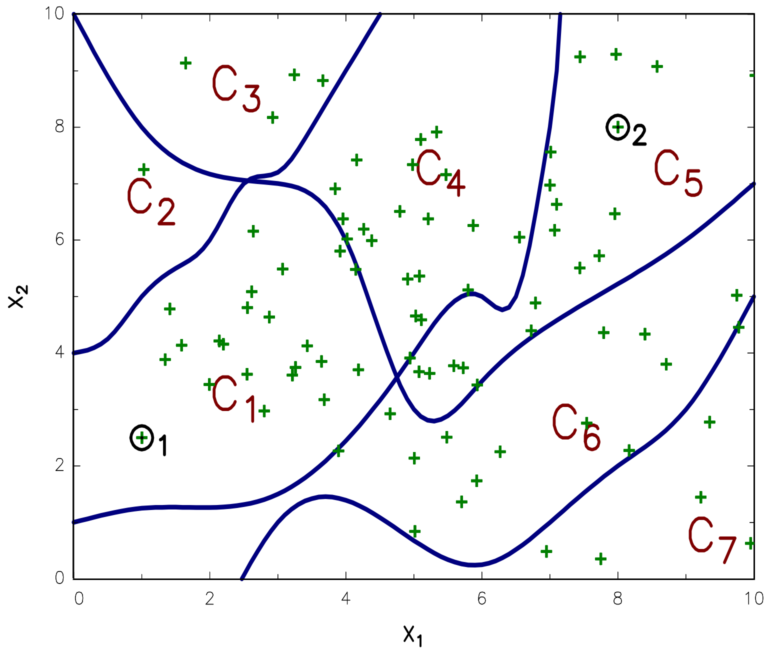

FIGURE 15.14: Classification statistical problem

where $\pi_j = \Pr\{i \in \mathcal{C}_j\}$ is the probability of the j^{th} class and $f_j(x) = \Pr\{X = x \mid i \in \mathcal{C}_j\}$ is the conditional probability density function of X. By construction, the decision boundary is defined such that we are indifferent to an assignment rule ($i \in \mathcal{C}_1$ and $i \in \mathcal{C}_2$), implying that:

$$\Pr\{i \in \mathcal{C}_1 \mid X = x\} = \Pr\{i \in \mathcal{C}_2 \mid X = x\} = \frac{1}{2}$$

Finally, we deduce that the decision boundary satisfies the following equation:

$$\ln \frac{f_1(x)}{f_2(x)} + \ln \frac{\pi_1}{\pi_2} = 0$$

If we model each class density as a multivariate normal distribution:

$$X \mid i \in \mathcal{C}_j \sim \mathcal{N}(\mu_j, \Sigma_j)$$

we have:

$$f_j(x) = \frac{1}{(2\pi)^{K/2} |\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_j)^\top \Sigma_j^{-1}(x-\mu_j)\right)$$

We deduce that:

$$\begin{aligned}\ln \frac{f_1(x)}{f_2(x)} &= \frac{1}{2}\ln\frac{|\Sigma_2|}{|\Sigma_1|} - \frac{1}{2}(x-\mu_1)^\top \Sigma_1^{-1}(x-\mu_1) + \\ &\quad \frac{1}{2}(x-\mu_2)^\top \Sigma_2^{-1}(x-\mu_2)\end{aligned}$$

The decision boundary is then given by:

$$\begin{aligned}\frac{1}{2}\ln\frac{|\Sigma_2|}{|\Sigma_1|} - \frac{1}{2}(x-\mu_1)^\top \Sigma_1^{-1}(x-\mu_1) + \\ \frac{1}{2}(x-\mu_2)^\top \Sigma_2^{-1}(x-\mu_2) + \ln\frac{\pi_1}{\pi_2} &= 0 \qquad (15.8)\end{aligned}$$

Since the decision boundary is quadratic in x, such approach is called quadratic discriminant analysis (QDA).

If we assume that $\Sigma_1 = \Sigma_2 = \Sigma$, Equation (15.8) becomes:

$$\frac{1}{2}(x-\mu_2)^\top \Sigma^{-1}(x-\mu_2) - \frac{1}{2}(x-\mu_1)^\top \Sigma^{-1}(x-\mu_1) + \ln\frac{\pi_1}{\pi_2} = 0$$

or:

$$(\mu_2-\mu_1)^\top \Sigma^{-1} x = \frac{1}{2}\left(\mu_2^\top \Sigma^{-1}\mu_2 - \mu_1^\top \Sigma^{-1}\mu_1\right) + \ln\frac{\pi_2}{\pi_1} \qquad (15.9)$$

It follows that the decision boundary is then linear in x. This is why we called this approach the linear discriminant analysis (LDA).

Example 168 *We consider two classes and two explanatory variables $X = (X_1, X_2)$ where $\pi_1 = 50\%$, $\pi_2 = 1 - \pi_1 = 50\%$, $\mu_1 = (1, 3)$, $\mu_2 = (4, 1)$, $\Sigma_1 = I_2$ and $\Sigma_2 = \gamma I_2$ where $\gamma = 1.5$.*

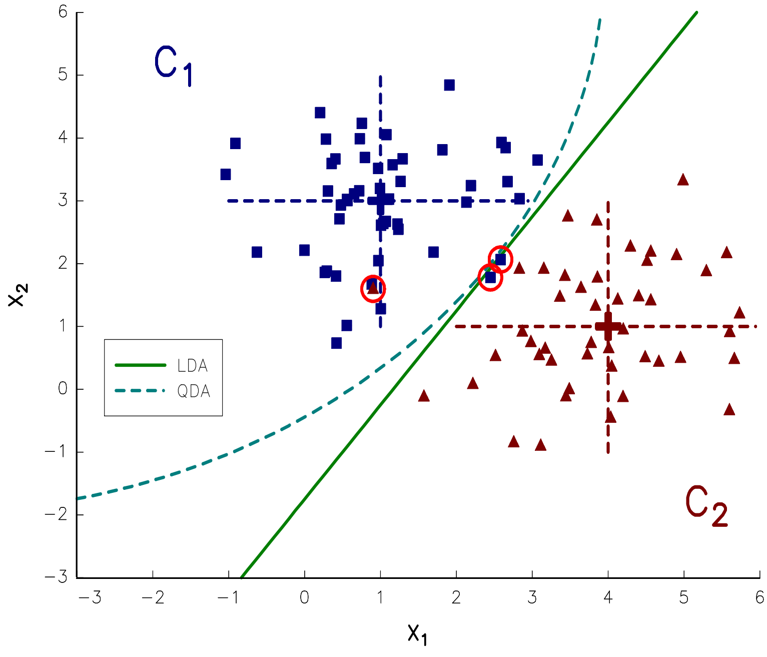

FIGURE 15.15: Boundary decision of discriminant analysis

By solving Equations (15.8) and (15.9), we obtain the QDA and LDA decision boundaries[27] reported in Figure 15.15. We verify that the LDA decision boundary is linear while the QDA decision region is convex. For each class, we have also simulated 50 realizations. We observe that the discriminant analysis performs the right classification most of the times. However, we notice that two observations from class $\mathcal{C}_1$ and one observation from class $\mathcal{C}_2$ are not properly classified. In Figure 15.16, we analyze the impact of the parameters on the decision boundary. The top/left panel corresponds to the previous example, whereas we only change one parameter for each other panel. For instance, we increase the variance of the second variable in the top/right panel. We observe that the impact on the LDA decision

[27]For the linear discriminant analysis, we have used $\Sigma = (\Sigma_1 + \Sigma_2)/2$.

boundary is minor, but this is not the case for the QDA decision boundary. Indeed, the convexity is stronger because X_2 can take more larger values than X_1. This is why for the extreme values, the QDA decision boundary can be approximated by a vertical line when $x_1 \to -\infty$ and an horizontal line when $x_2 \to +\infty$. Let us now introduce a correlation ρ between X_1 and X_2. It follows that the QDA decision boundary becomes more and more linear when we increase ρ (bottom/left panel). Finally, the impact of the probabilities (π_1, π_2) is crucial as shown in the bottom/right panel. It is obvious that the boundary decision moves to the right when π_1 increases, because the decision region concerning $i \in \mathcal{C}_1$ must be larger. For instance, we must always accept $i \in \mathcal{C}_1$ at the limit case $\pi_1 = 100\%$.

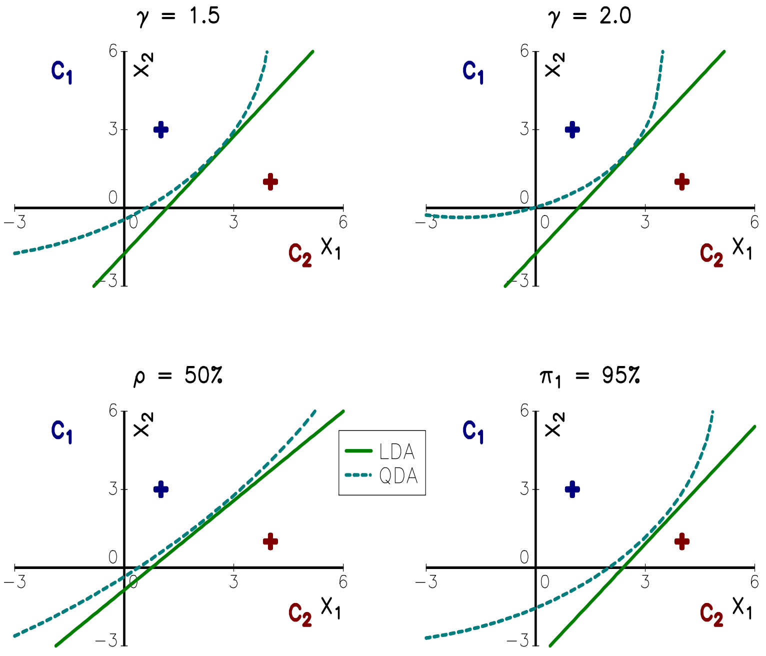

FIGURE 15.16: Impact of the parameters on LDA/QDA boundary decisions

The general case We can generalize the previous analysis to J classes. In this case, the Bayes formula gives:

$$\begin{aligned}\Pr\{i \in \mathcal{C}_j \mid X = x\} &= \Pr\{X = x \mid i \in \mathcal{C}_j\} \cdot \frac{\Pr\{i \in \mathcal{C}_j\}}{\Pr\{X = x\}} \\ &= c \cdot f_j(x) \cdot \pi_j\end{aligned}$$

where $c = 1/\Pr\{X = x\}$ is a normalization constant that does not depend on j. We note $S_j(x) = \ln \Pr\{i \in \mathcal{C}_j \mid X = x\}$ the discriminant score function for the j^{th} class. We have:

$$S_j(x) = \ln c + \ln f_j(x) + \ln \pi_j$$

If we again assume that $X \mid i \in \mathcal{C}_j \sim \mathcal{N}(\mu_j, \Sigma_j)$, we obtain:

$$\begin{aligned}S_j(x) &= \ln c' + \ln \pi_j - \frac{1}{2}\ln|\Sigma_j| - \frac{1}{2}(x-\mu_j)^\top \Sigma_j^{-1}(x-\mu_j) \\ &\propto \ln \pi_j - \frac{1}{2}\ln|\Sigma_j| - \frac{1}{2}(x-\mu_j)^\top \Sigma_j^{-1}(x-\mu_j) \qquad (15.10)\end{aligned}$$

where $\ln c' = \ln c - \dfrac{K}{2}\ln 2\pi$. Given an input x, we calculate the scores $S_j(x)$ for $j = 1, \ldots, J$ and we choose the label $j^\star$ with the highest score value. As in the two-class case, we can assume an homoscedastic model ($\Sigma_j = \Sigma$), implying that the discriminant score function becomes:

$$\begin{aligned} S_j(x) &= \ln c'' + \ln \pi_j - \frac{1}{2}(x-\mu_j)^\top \Sigma_j^{-1}(x-\mu_j) \\ &\propto \ln \pi_j + \mu_j^\top \Sigma^{-1} x - \frac{1}{2}\mu_j^\top \Sigma^{-1}\mu_j \end{aligned} \tag{15.11}$$

where $\ln c'' = \ln c' - \dfrac{1}{2}\ln|\Sigma| - \dfrac{1}{2}x^\top\Sigma^{-1}x$. Equation (15.11) defines the LDA score function, whereas Equation (15.10) defines the QDA score function.

Remark 185 *In practice, the parameters π_j, μ_j and Σ_j are unknown. We replace them by the corresponding estimates $\hat{\pi}_j$, $\hat{\mu}_j$ and $\hat{\Sigma}_j$. For the linear discriminant analysis, $\hat{\Sigma}$ is estimated by pooling all the classes.*

Example 169 *We consider the classification problem of 33 observations with two explanatory variables X_1 and X_2, and three classes $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$. The data are reported in Table 15.12.*

TABLE 15.12: Data of the classification problem

i	$\mathcal{C}_j$	X_1	X_2	i	$\mathcal{C}_j$	X_1	X_2	i	$\mathcal{C}_j$	X_1	X_2
1	1	1.03	2.85	12	2	3.70	5.08	23	3	3.55	0.58
2	1	0.20	3.30	13	2	2.81	1.99	24	3	3.86	1.83
3	1	1.69	3.73	14	2	3.66	2.61	25	3	5.39	0.47
4	1	0.98	3.52	15	2	5.63	4.19	26	3	3.15	−0.18
5	1	0.98	5.15	16	2	3.35	3.64	27	3	4.93	1.91
6	1	3.47	6.56	17	2	2.97	3.55	28	3	3.87	2.61
7	1	3.94	4.68	18	2	3.16	2.92	29	3	4.09	1.43
8	1	1.55	5.99	19	3	3.00	0.98	30	3	3.80	2.11
9	1	1.15	3.60	20	3	3.09	1.99	31	3	2.79	2.10
10	2	1.20	2.27	21	3	5.45	0.60	32	3	4.49	2.71
11	2	3.66	5.49	22	3	3.59	−0.46	33	3	3.51	1.82

The first step is to estimate the parameters π_j, μ_j and Σ_j, whose values[28] are reported in Table 15.13. The second step consists in calculating the score function $S_j(x)$ for each class j using Equations (15.10) and (15.11). Results are given in Table 15.14. Besides the QDA and LDA methods, we have also considered a third approach LDA^2, which corresponds to a linear discriminant analysis by including the squared values of variables. This means that the explanatory variables are X_1, X_2, X_1^2 and X_2^2 in LDA^2. By including polynomials, the LDA^2 method is more convex than the original LDA method, and can be seen as an approximation of the QDA method.

If we consider the first observation, the maximum score is reached for the first class (-2.28 for QDA, 0.21 for LDA and 6.93 for LDA^2). If we consider the 14th observation,

[28]For the LDA method, we have:

$$\hat{\Sigma} = \begin{pmatrix} 1.91355 & -0.71720 \\ -0.71720 & 3.01577 \end{pmatrix}$$

TABLE 15.13: Parameter estimation of the discriminant analysis

Class	$\mathcal{C}_1$		$\mathcal{C}_2$		$\mathcal{C}_3$	
$\hat{\pi}_j$	0.273		0.273		0.455	
$\hat{\mu}_j$	1.666	4.376	3.349	3.527	3.904	1.367
$\hat{\Sigma}_j$	1.525	0.929	1.326	0.752	0.694	−0.031
	0.929	1.663	0.752	1.484	−0.031	0.960

QDA and LDA predict the third class, whereas LDA^2 predicts the second class, which is the true value. In Figure 15.17, we have reported the class assignment performed by the three approaches, and we have indicated the bad class predictions by a circle. In order to understand these results, we have also calculated the decision regions in Figure 15.18. According to QDA, the decision boundary is almost linear between $\mathcal{C}_1$ and $\mathcal{C}_2$, whereas it is quadratic between $\mathcal{C}_2$ and $\mathcal{C}_3$. LDA produces linear decision boundaries, but the decision surface for $\mathcal{C}_1$ has changed. Finally, LDA^2 can produce complex decision surfaces, even more complex than those produced by QDA.

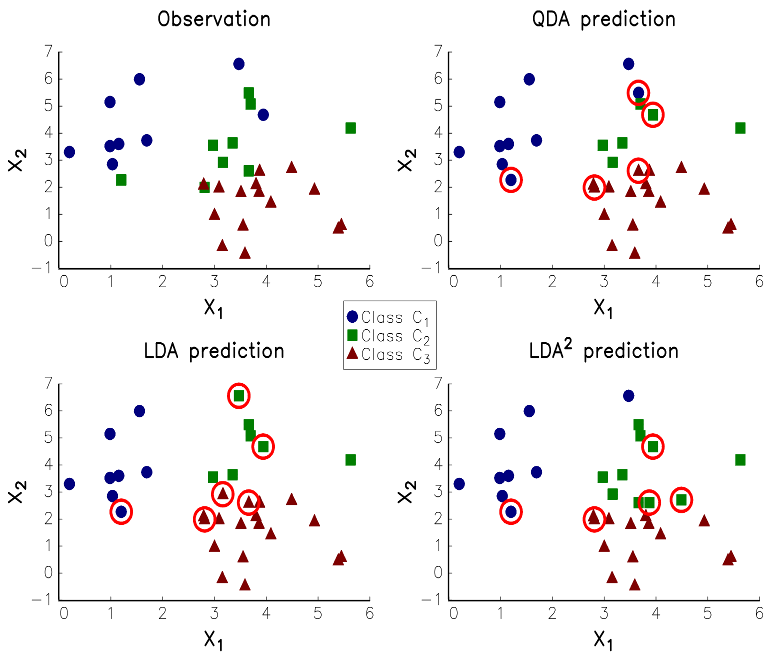

FIGURE 15.17: Comparing QDA, LDA and LDA^2 predictions

Class separation maximization In the following, we show that the linear discriminant analysis is equivalent to maximize class separability and is also related to the principal component analysis. We note $x_i = (x_{i,1}, \ldots, x_{i,K})$ the $K \times 1$ vector of exogenous variables X for the i^{th} observation. The mean vector and the variance (or scatter) matrix of Class $\mathcal{C}_j$ is equal to:

$$\hat{\mu}_j = \frac{1}{n_j} \sum_{i \in \mathcal{C}_j} x_i$$

TABLE 15.14: Computation of the discriminant scores $S_j(x)$

i	QDA			LDA			LDA^2		
	$S_1(x)$	$S_2(x)$	$S_3(x)$	$S_1(x)$	$S_2(x)$	$S_3(x)$	$S_1(x)$	$S_2(x)$	$S_3(x)$
1	−2.28	−3.69	−7.49	0.21	−0.96	−0.79	6.93	5.60	5.76
2	−2.28	−6.36	−12.10	−0.26	−2.17	−2.34	1.38	−2.13	−1.89
3	−1.76	−3.13	−6.79	2.84	2.16	1.71	12.13	12.01	11.38
4	−1.80	−4.43	−8.88	1.35	0.09	−0.22	7.73	6.20	5.93
5	−2.36	−7.75	−13.70	4.32	2.93	1.45	8.12	5.54	4.76
6	−3.16	−5.63	−14.68	10.75	11.36	8.95	14.82	13.99	12.96
7	−3.79	−1.92	−6.32	8.06	9.22	8.15	17.36	19.03	17.89
8	−2.85	−8.43	−15.23	6.73	5.76	3.70	10.47	8.09	7.15
9	−1.74	−4.12	−8.37	1.76	0.64	0.27	8.94	7.77	7.39
10	−3.14	−3.21	−6.17	−0.58	−1.56	−0.98	6.59	5.55	6.15
11	−2.87	−3.01	−9.45	9.10	9.96	8.31	16.89	17.65	16.42
12	−3.04	−2.38	−7.77	8.42	9.34	7.98	17.28	18.50	17.28
13	−6.32	−2.29	−1.62	1.41	1.82	2.64	12.48	13.94	14.46
14	−6.91	−2.07	−1.42	3.86	4.94	5.34	15.15	17.41	17.34
15	−9.79	−3.62	−7.12	9.79	12.43	11.75	12.58	14.01	13.50
16	−3.90	−1.47	−3.44	5.25	5.99	5.65	16.84	18.82	18.03
17	−3.31	−1.55	−3.61	4.50	4.92	4.63	16.25	17.95	17.21
18	−4.84	−1.60	−2.19	3.65	4.28	4.45	15.51	17.48	17.14
19	−10.21	−4.12	−1.27	−0.13	0.52	2.06	8.98	9.99	11.70
20	−7.05	−2.41	−1.24	1.85	2.50	3.32	12.99	14.72	15.22
21	−23.11	−11.16	−2.56	2.98	5.75	7.61	3.79	4.57	7.26
22	−19.22	−9.53	−2.42	−1.84	−0.57	2.01	1.81	1.53	5.51
23	−13.86	−5.92	−1.01	−0.01	1.15	2.98	7.65	8.67	10.95
24	−10.01	−3.43	−0.70	2.75	4.07	5.02	12.84	14.95	15.65
25	−23.48	−11.44	−2.54	2.65	5.38	7.33	3.40	4.09	6.95
26	−15.87	−7.59	−2.30	−2.01	−1.14	1.23	3.19	3.02	6.50
27	−14.09	−5.40	−1.52	4.56	6.78	7.70	11.17	13.24	14.08
28	−7.55	−2.27	−1.39	4.18	5.45	5.85	15.10	17.44	17.40
29	−12.40	−4.67	−0.61	2.38	3.92	5.17	11.21	13.14	14.33
30	−8.85	−2.87	−0.88	3.17	4.41	5.17	13.77	15.97	16.37
31	−5.97	−2.17	−1.72	1.58	1.97	2.70	12.78	14.26	14.67
32	−9.40	−2.97	−1.81	5.33	7.11	7.46	14.55	16.95	16.93
33	−8.84	−3.01	−0.80	2.19	3.21	4.16	12.82	14.77	15.45

and[29]:

$$\mathbf{S}_j = n\hat{\Sigma}_j = \sum_{i \in \mathcal{C}_j} (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^\top$$

where n_j is the number of observations in the j^{th} class. If consider the total population, we also have:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

[29]The variance matrix is equal to the unscaled covariance matrix and is also called the scatter matrix.

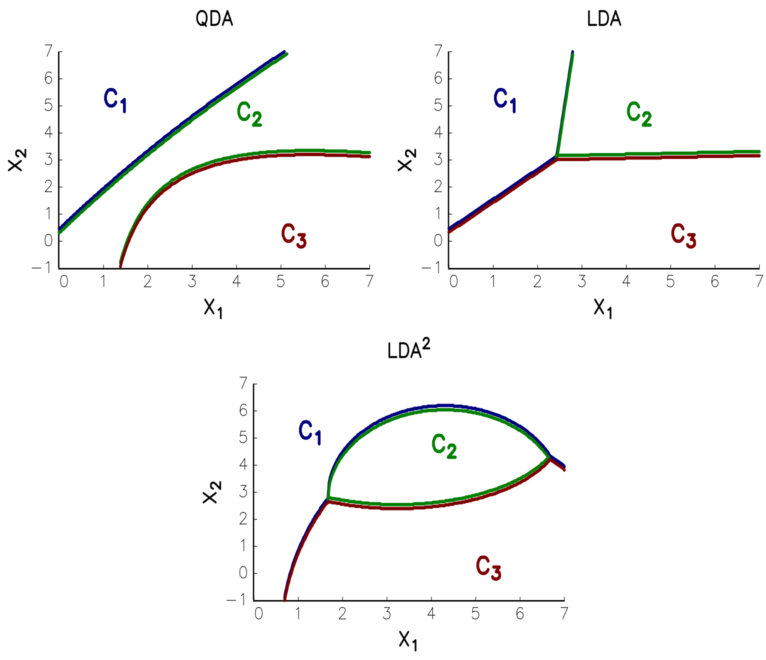

FIGURE 15.18: QDA, LDA and LDA^2 decision regions

and:

$$\mathbf{S} = n\hat{\Sigma} = \sum_{i=1}^{n} (x_i - \hat{\mu})(x_i - \hat{\mu})^\top$$

We notice that:

$$\hat{\mu} = \frac{1}{n}\sum_{j=1}^{J} n_j \hat{\mu}_j$$

We define the between-class variance matrix as:

$$\mathbf{S}_B = \sum_{j=1}^{J} n_j (\hat{\mu}_j - \hat{\mu})(\hat{\mu}_j - \hat{\mu})^\top$$

and the within-class variance matrix as:

$$\mathbf{S}_W = \sum_{j=1}^{J} \mathbf{S}_j$$

We can show that the total variance matrix can be decomposed into the sum of the within-class and between-class variance matrices[30]:

$$\mathbf{S} = \mathbf{S}_W + \mathbf{S}_B$$

The discriminant analysis defined by Fisher (1936) consists in finding the discriminant linear combination $\beta^\top X$ that has the maximum between-class variance relative to the within-class variance: $\beta^\star = \arg\max J(\beta)$ where $J(\beta)$ is the Fisher criterion:

$$J(\beta) = \frac{\beta^\top \mathbf{S}_B \beta}{\beta^\top \mathbf{S}_W \beta}$$

[30]See Exercise 15.4.5 on page 1024.

Since the objective function is invariant if we rescale the vector β – $J(\beta') = J(\beta)$ if $\beta' = c\beta$, we can impose that $\beta^\top \mathbf{S}_W \beta = 1$. It follows that:

$$\begin{aligned} \hat{\beta} &= \arg\max \beta^\top \mathbf{S}_B \beta \\ \text{s.t.} &\quad \beta^\top \mathbf{S}_W \beta = 1 \end{aligned} \tag{15.12}$$

The Lagrange function is:

$$\mathcal{L}(\beta;\lambda) = \beta^\top \mathbf{S}_B \beta - \lambda\left(\beta^\top \mathbf{S}_W \beta - 1\right)$$

We deduce that the first-order condition is equal to:

$$\frac{\partial\,\mathcal{L}(\beta;\lambda)}{\partial\,\beta^\top} = 2\mathbf{S}_B\beta - 2\lambda\mathbf{S}_W\beta = \mathbf{0} \tag{15.13}$$

It is remarkable that we obtain a generalized eigenvalue problem[31] $\mathbf{S}_B\beta = \lambda\mathbf{S}_W\beta$ or equivalently:

$$\mathbf{S}_W^{-1}\mathbf{S}_B\beta = \lambda\beta \tag{15.14}$$

Even if $\mathbf{S}_W$ and $\mathbf{S}_B$ are two symmetric matrices, it is not necessarily the case for the product $\mathbf{S}_W^{-1}\mathbf{S}_B$. Using the eigendecomposition $\mathbf{S}_B = V\Lambda V^\top$, we have $\mathbf{S}_B^{1/2} = V\Lambda^{1/2}V^\top$. With the parametrization $\alpha = \mathbf{S}_B^{1/2}\beta$, Equation (15.14) becomes:

$$\mathbf{S}_B^{1/2}\mathbf{S}_W^{-1}\mathbf{S}_B^{1/2}\alpha = \lambda\alpha \tag{15.15}$$

because $\beta = \mathbf{S}_B^{-1/2}\alpha$. Equation (15.15) defines a right regular eigenvalue problem. Let λ_k and v_k be the k^{th} eigenvalue and eigenvector of the symmetric matrix $\mathbf{S}_B^{1/2}\mathbf{S}_W^{-1}\mathbf{S}_B^{1/2}$. It is obvious that the optimal solution $\alpha^\star$ is the first eigenvector v_1 corresponding to the largest eigenvalue λ_1. We conclude that the estimator is $\hat{\beta} = \mathbf{S}_B^{-1/2}v_1$ and the discriminant linear relationship is $Y^c = v_1^\top \mathbf{S}_B^{-1/2}X$. Moreover, we have[32]:

$$\lambda_1 = J\left(\hat{\beta}\right) = \frac{\hat{\beta}^\top \mathbf{S}_B \hat{\beta}}{\hat{\beta}^\top \mathbf{S}_W \hat{\beta}}$$

In Exercise 15.4.5 on page 1024, we show that the Fisher discriminant analysis is equivalent to the linear discriminant analysis in the case of two classes. This result can be extended to multiple classes and explains why this approach is also called Fisher linear discriminant analysis.

Example 170 *We consider a problem with two classes $\mathcal{C}_1$ and $\mathcal{C}_2$, and two explanatory variables (X_1, X_2). Class $\mathcal{C}_1$ is composed of 7 observations:* $(1,2)$, $(1,4)$, $(3,6)$, $(3,3)$, $(4,2)$, $(5,6)$, $(5,5)$, *whereas class $\mathcal{C}_2$ is composed of 6 observations:* $(1,0)$, $(2,1)$, $(4,1)$, $(3,2)$, $(6,4)$ *and* $(6,5)$.

In Figure 15.19, we have reported these 13 observations in the plane (x_1, x_2). The computation of the first generalized eigenvector gives $\beta = (0.7547, -0.9361)$. We deduce that the slope of the optimal line direction is $\beta_1/\beta_2 = -0.8062$. Computing the Fisher score $s_i = \beta^\top x_i$ for the i^{th} observation is then equivalent to perform the orthogonal projection of

[31] See Appendix A.1.1.2 on page 1034 for the definition of the generalized eigendecomposition.

[32] Thanks to Equation (15.13), we have $\mathbf{S}_B\beta = \lambda\mathbf{S}_W\beta$ and $\beta^\top\mathbf{S}_B\beta = \lambda\beta^\top\mathbf{S}_W\beta$.

the points on this optimal line (Bishop, 2006). Concerning the assignment decision, we can consider the midpoint rule:

$$\begin{cases} s_i < \bar{\mu} \Rightarrow i \in \mathcal{C}_1 \\ s_i > \bar{\mu} \Rightarrow i \in \mathcal{C}_2 \end{cases}$$

where $\bar{\mu} = (\bar{\mu}_1 + \bar{\mu}_2)/2$, $\bar{\mu}_1 = \beta^\top \hat{\mu}_1$ and $\bar{\mu}_2 = \beta^\top \hat{\mu}_2$. However, this rule is not always optimal because it does not depend on the variance $\bar{s}_1^2$ and $\bar{s}_2^2$ of each class. In Figure 15.20, we have reported the Gaussian density of the scores for the two classes. Since we observe that the first class has a larger variance, the previous rule is not adapted. This is why we can use the tools presented in Section 15.3 in order to calibrate the optimal decision rule.

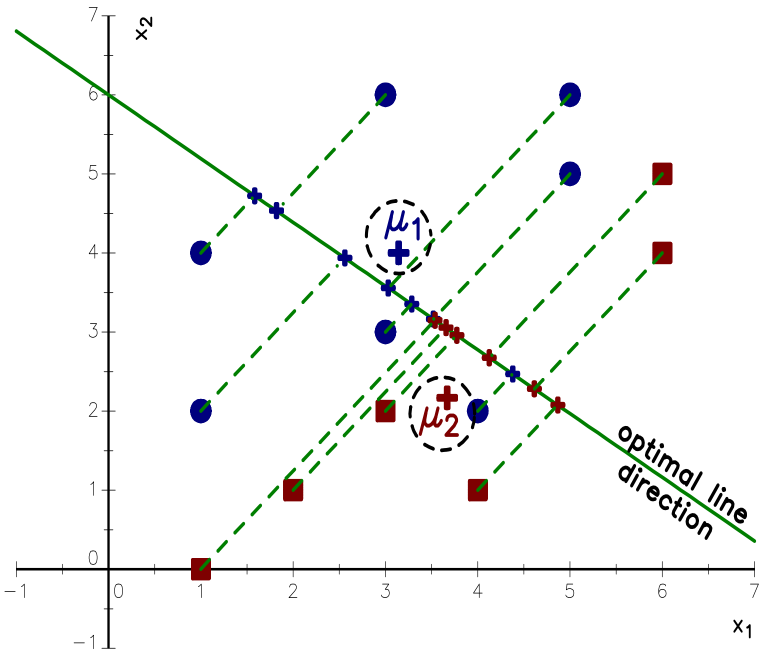

FIGURE 15.19: Linear projection and the Fisher solution

Remark 186 *$v_1^\top \mathbf{S}_B^{-1/2} X$ is called the first canonical or discriminant variable (Hastie et al., 2009) and we denote it by $Y_{(1)}^c$. The previous analysis can be used to find the second canonical variable $Y_{(2)}^c = \beta_{(2)}^\top X$ that is not correlated to $Y_{(1)}^c$ such that $J\left(\beta_{(2)}\right)$ is maximum. The solution is $\hat{\beta}_{(2)} = \mathbf{S}_B^{-1/2} v_2$ where v_2 is the eigenvector associated to the second largest eigenvalue λ_2. This method can be extended to the general problem of finding the k^{th} canonical variable $Y_{(k)}^c = \beta_{(k)}^\top X$ that is not correlated to $\left(Y_{(1)}^c, \ldots, Y_{(k-1)}^c\right)$ such that $J\left(\beta_{(k)}\right)$ is maximum. Again, we can show that the solution is $\hat{\beta}_{(k)} = \mathbf{S}_B^{-1/2} v_k$ where v_k is the eigenvector associated to the k^{th} largest eigenvalue λ_k. The computation of the K linear relationships $Y_{(k)}^c = \beta_{(k)}^\top X$ is called the multiple discriminant analysis (MDA). MDA can be seen as a generalized PCA method by taking into account a categorical response variable. Indeed, PCA performs an eigendecomposition of $\mathbf{S}$ (or $\hat{\Sigma}$) whereas MDA performs an eigendecomposition of $\mathbf{S}_W^{-1} \mathbf{S}_B$.*

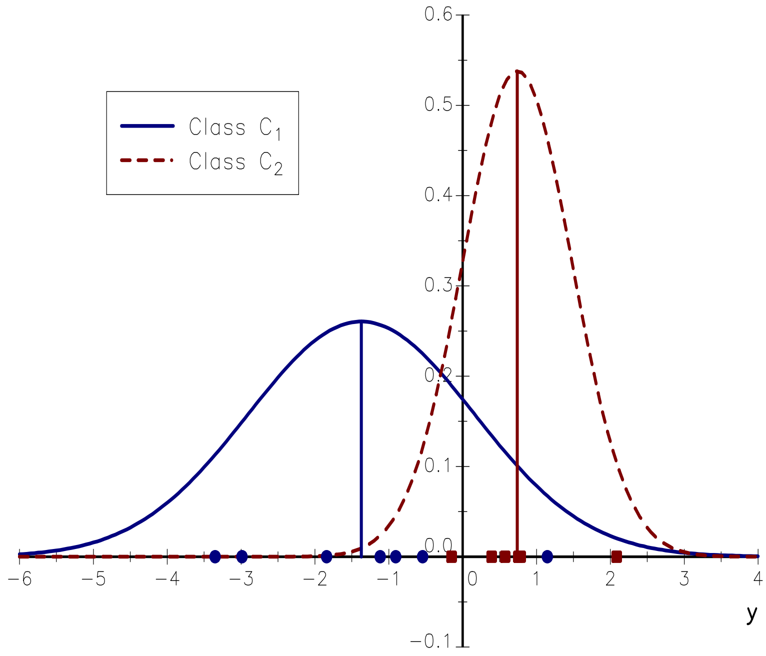

FIGURE 15.20: Class separation and the cut-off criterion

15.2.2.2 Binary choice models

The underlying idea of such models is to estimate the probability of a binary response based on several explanatory variables. They have been developed in several fields of research (biology, epidemiology, economy, etc.). In statistics, the two seminal papers are again written by Fisher (1935) and Cox (1958). Since these publications, these models have been extended and now represent a major field of study in statistics and econometrics[33].

General framework In this section, we assume that Y can take two values 0 and 1. We consider models that link the outcome to a set of factors X:

$$\Pr\{Y = 1 \mid X = x\} = \mathbf{F}\left(x^\top \beta\right)$$

By construction, $\mathbf{F}$ must be a cumulative distribution function in order to ensure that $\mathbf{F}(z) \in [0,1]$. We also assume that the model is symmetric, implying that $\mathbf{F}(z) + \mathbf{F}(-z) = 1$. Given a sample $\{(x_i, y_i), i = 1, \ldots, n\}$, the log-likelihood function is equal to:

$$\ell(\theta) = \sum_{i=1}^{n} \ln \Pr\{Y_i = y_i\}$$

where y_i takes the values 0 or 1. We have:

$$\Pr\{Y_i = y_i\} = p_i^{y_i} \cdot (1 - p_i)^{1-y_i}$$

[33]The materials presented below is based on surveys by Amemiya (1981, 1985) and McFadden (1984).

where $p_i = \Pr\{Y_i = 1 \mid X_i = x_i\}$. We deduce that:

$$\begin{aligned}\boldsymbol{\ell}(\theta) &= \sum_{i=1}^{n} y_i \ln p_i + (1-y_i)\ln(1-p_i) \\ &= \sum_{i=1}^{n} y_i \ln \mathbf{F}\left(x_i^\top \beta\right) + (1-y_i)\ln\left(1-\mathbf{F}\left(x_i^\top \beta\right)\right)\end{aligned}$$

We notice that the vector θ includes only the parameters β. By noting $f(z)$ the probability density function, it follows that the associated score vector and Hessian matrix of the log-likelihood function are:

$$\begin{aligned}\mathcal{S}(\beta) &= \frac{\partial\,\boldsymbol{\ell}(\beta)}{\partial\,\beta} \\ &= \sum_{i=1}^{n}\left(y_i \frac{f\left(x_i^\top\beta\right)}{\mathbf{F}\left(x_i^\top\beta\right)} - (1-y_i)\frac{f\left(x_i^\top\beta\right)}{1-\mathbf{F}\left(x_i^\top\beta\right)}\right)x_i \\ &= \sum_{i=1}^{n}\frac{f\left(x_i^\top\beta\right)}{\mathbf{F}\left(x_i^\top\beta\right)\mathbf{F}\left(-x_i^\top\beta\right)}\left(y_i - \mathbf{F}\left(x_i^\top\beta\right)\right)x_i\end{aligned}$$

and:

$$\begin{aligned}H(\beta) &= \frac{\partial^2\,\boldsymbol{\ell}(\beta)}{\partial\,\beta\,\partial\,\beta^\top} \\ &= -\sum_{i=1}^{n} H_i \cdot \left(x_i x_i^\top\right)\end{aligned}$$

where:

$$\begin{aligned}H_i = {} & \frac{f\left(x_i^\top\beta\right)^2}{\mathbf{F}\left(x_i^\top\beta\right)\mathbf{F}\left(-x_i^\top\beta\right)} - \left(y_i - \mathbf{F}\left(x_i^\top\beta\right)\right)\cdot \\ & \left(\frac{f'\left(x_i^\top\beta\right)}{\mathbf{F}\left(x_i^\top\beta\right)\mathbf{F}\left(-x_i^\top\beta\right)} - \frac{f\left(x_i^\top\beta\right)^2\left(1-2\mathbf{F}\left(x_i^\top\beta\right)\right)}{\mathbf{F}\left(x_i^\top\beta\right)^2\mathbf{F}\left(-x_i^\top\beta\right)^2}\right)\end{aligned}$$

Once $\hat{\beta}$ is estimated by the method of maximum likelihood, we can calculated the predicted probability for the i^{th} observation:

$$\hat{p}_i = \mathbf{F}\left(x_i^\top\hat{\beta}\right)$$

Like a linear regression model, we can define the residual as the difference between the observation y_i and the predicted value $\hat{p}_i$. We can also exploit the property that the conditional distribution of Y_i is a Bernoulli distribution $\mathcal{B}(p_i)$. This is why it is better to use the standardized (or Pearson) residuals:

$$\hat{u}_i = \frac{y_i - \hat{p}_i}{\sqrt{\hat{p}_i(1-\hat{p}_i)}}$$

These residuals are related to the Pearson's chi-squared statistic:

$$\begin{aligned}\chi^2_{\text{Pearson}} &= \sum_{i=1}^{n}\hat{u}_i^2 \\ &= \sum_{i=1}^{n}\frac{(y_i-\hat{p}_i)^2}{\hat{p}_i(1-\hat{p}_i)}\end{aligned}$$

This statistic may used to measure the goodness-of-fit of the model. Under the assumption $\mathcal{H}_0$ that there is no lack-of-fit, we have $\chi^2_{\text{Pearson}} \sim \chi^2_{n-K}$ where K is the number of exogenous variables. Another goodness-of-fit statistic is the likelihood ratio. For the '*saturated*' model, the estimated probability $\hat{p}_i$ is exactly equal to y_i. We deduce that the likelihood ratio is equal to:

$$\begin{aligned} -2\ln\Lambda &= 2\sum_{i=1}^{n} y_i \ln y_i + (1-y_i)\ln(1-y_i) - \\ & \quad 2\sum_{i=1}^{n} y_i \ln \hat{p}_i + (1-y_i)\ln(1-\hat{p}_i) \\ &= 2\sum_{i=1}^{n} y_i \ln\left(\frac{y_i}{\hat{p}_i}\right) + (1-y_i)\ln\left(\frac{1-y_i}{1-\hat{p}_i}\right) \end{aligned}$$

In binomial choice models, $D = -2\ln\Lambda$ is also called the deviance and we have $D \sim \chi^2_{n-K}$. In a perfect fit $\hat{p}_i = y_i$, the likelihood ratio is exactly equal to zero. The forecasting procedure consists of estimating the probability $\hat{p} = \mathbf{F}\left(x^\top \hat{\beta}\right)$ for a given set of variables x and to use the following decision criterion:

$$Y = 1 \Leftrightarrow \hat{p} \geq \frac{1}{2}$$

Remark 187 *It could also be interesting to compute the marginal effects. We have:*

$$\mathbb{E}\left[Y \mid X = x\right] = \mathbf{F}\left(x^\top \hat{\beta}\right)$$

and:

$$\frac{\partial\, \mathbb{E}\left[Y \mid X = x\right]}{\partial\, x} = f\left(x^\top \hat{\beta}\right) \cdot \hat{\beta}$$

The marginal effects depend on the vector x and are then not easy to understand. This is why we generally compute them by using the mean of the regressors or averaging them across all the observations of the sample.

Logit analysis The logit model uses the following cumulative distribution function:

$$\mathbf{F}(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$$

The probability density function is then equal to:

$$f(z) = \frac{e^{-z}}{\left(1+e^{-z}\right)^2}$$

We verify the property $\mathbf{F}(z) + \mathbf{F}(-z) = 1$. The log-likelihood function is equal to:

$$\begin{aligned} \boldsymbol{\ell}(\beta) &= \sum_{i=1}^{n} (1-y_i)\ln\left(1 - \mathbf{F}\left(x_i^\top \beta\right)\right) + y_i \ln \mathbf{F}\left(x_i^\top \beta\right) \\ &= \sum_{i=1}^{n} (1-y_i)\ln\left(\frac{e^{-x_i^\top \beta}}{1+e^{-x_i^\top \beta}}\right) - y_i \ln\left(1+e^{-x_i^\top \beta}\right) \\ &= -\sum_{i=1}^{n} \ln\left(1+e^{-x_i^\top \beta}\right) + (1-y_i)\left(x_i^\top \beta\right) \end{aligned}$$

We also have[34]:

$$\mathcal{S}(\beta) = \sum_{i=1}^{n} \left(y_i - \mathbf{F}\left(x_i^\top \beta\right)\right) x_i$$

and[35]:

$$H(\beta) = -\sum_{i=1}^{n} f\left(x_i^\top \beta\right) \cdot \left(x_i x_i^\top\right)$$

Probit analysis The probit model assumes that $\mathbf{F}(z)$ is the Gaussian distribution. The log-likelihood function is then:

$$\boldsymbol{\ell}(\beta) = \sum_{i=1}^{n} (1 - y_i) \ln\left(1 - \Phi\left(x_i^\top \beta\right)\right) + y_i \ln \Phi\left(x_i^\top \beta\right)$$

The probit model can be seen as a latent variable model. Let us consider the linear model $Y^\star = \beta^\top X + U$ where $U \sim \mathcal{N}\left(0, \sigma^2\right)$. We assume that we do not observe $Y^\star$ but $Y = g(Y^\star)$. For example, if $g(z) = \mathbb{1}\{z > 0\}$, we obtain:

$$\begin{aligned} \Pr\{Y = 1 \mid X = x\} &= \Pr\left\{\beta^\top X + U > 0 \mid X = x\right\} \\ &= \Phi\left(\frac{\beta^\top x}{\sigma}\right) \end{aligned}$$

We notice that only the ratio β/σ is identifiable. Since we can set $\sigma = 1$, we obtain the probit model.

Regularization Let $\boldsymbol{\ell}(\theta)$ be the log-likelihood function. The regularized log-likelihood function is equal to:

$$\boldsymbol{\ell}(\theta; \lambda) = \boldsymbol{\ell}(\theta) - \frac{\lambda}{p} \|\theta\|_p^p$$

The case $p = 1$ is equivalent to consider a lasso penalization, whereas $p = 2$ corresponds to the ridge regularization. The optimal value $\theta^\star$ is obtained by maximizing the regularized log-likelihood function:

$$\theta^\star(\lambda) = \arg\max \boldsymbol{\ell}(\theta; \lambda)$$

In this problem, we consider λ as an hyperparameter, meaning that λ is not directly estimated by maximizing the penalized log-likelihood function with respect to $(\theta; \lambda)$. For instance, in the case of the lasso regularization, λ can be calibrated in order to obtain a sparse model or using cross-validation techniques.

[34]We use the property $f(z) = \mathbf{F}(z)(1 - \mathbf{F}(z))$, implying that:

$$\frac{f(z)}{\mathbf{F}(z)\mathbf{F}(-z)} = \frac{f(z)}{\mathbf{F}(z)(1 - \mathbf{F}(z))} = 1$$

[35]We use the property $f'(z) = -f(z)\mathbf{F}(z)\left(1 - e^{-z}\right)$, implying that:

$$\frac{f'(z)}{f(z)} - (1 - 2\mathbf{F}(z)) = 0$$

Extension to multinomial logistic regression We assume that Y can take J labels $(\mathfrak{L}_1, \ldots, \mathfrak{L}_J)$ or belongs to J disjoint classes $(\mathcal{C}_1, \ldots, \mathcal{C}_J)$. We define the conditional probability as follows:

$$\begin{aligned} p_j(x) &= \Pr\{Y = \mathfrak{L}_j \mid X = x\} \\ &= \Pr\{Y \in \mathcal{C}_j \mid X = x\} \\ &= \frac{e^{\beta_j^\top x}}{1 + \sum_{j=1}^{J-1} e^{\beta_j^\top x}} \end{aligned}$$

for $j = 1, \ldots, J-1$. The probability of the last label is then equal to:

$$\begin{aligned} p_J(x) &= 1 - \sum_{j=1}^{J-1} p_j(x) \\ &= \frac{1}{1 + \sum_{j=1}^{J-1} e^{\beta_j^\top x}} \end{aligned}$$

We verify that $0 \leq p_j(x) \leq 1$ for all $j = 1, \ldots, J$. The log-likelihood function becomes:

$$\boldsymbol{\ell}(\theta) = \sum_{i=1}^{n} \ln \left(\prod_{j=1}^{J} p_j(x_i)^{i \in \mathcal{C}_j} \right)$$

where θ is the vector of parameters $(\beta_1, \ldots, \beta_{J-1})$.

The multinomial logistic model can be formulated as a log-linear model. We note:

$$\ln p_j(x) = \beta_0 + \beta_j^\top x$$

Since we have $\sum_{j=1}^{J} p_j(x) = 1$, we deduce that the constant β_0 is given by:

$$\sum_{j=1}^{J} e^{\beta_0 + \beta_j^\top x} = 1 \Leftrightarrow \beta_0 = \ln \frac{1}{\sum_{j=1}^{J} e^{\beta_j^\top x}}$$

It follows that:

$$p_j(x) = \frac{e^{\beta_j^\top x}}{\sum_{j=1}^{J} e^{\beta_j^\top x}}$$

This function is known as the softmax function and plays an important role in neural networks. We also notice that the model is overidentified because the sum of probabilities is equal to 1. However, if we use the parametrization $\breve{\beta}_j = \beta_j - \beta_J$, we obtain the previous model[36], which is just identified.

[36]Indeed, we have:

$$\begin{aligned} p_j(x) &= \frac{e^{\breve{\beta}_j^\top x} e^{-\beta_J^\top x}}{e^{-\beta_J^\top x} \sum_{j=1}^{J} e^{\breve{\beta}_j^\top x}} \\ &= \frac{e^{\breve{\beta}_j^\top x}}{1 + \sum_{j=1}^{J-1} e^{\breve{\beta}_j^\top x}} \end{aligned}$$

because $e^{\breve{\beta}_J^\top x} = e^{(\beta_J - \beta_J)^\top x} = 1$.

15.2.3 Non-parametric supervised methods

We have named this section '*non-parametric supervised methods*' in order to group some approaches that share some of the same characteristics. First, even if some of them are parametric, these models are highly non-linear, meaning that it is extremely difficult to interpret these models. In this case, '*forecasting*' is the main motivation and is more important than '*modeling*'. Second, it would be illusory to consider or to do statistical inference. Most of the time, it is impossible to calculate the variance of the parameters and the associated t-statistics. Therefore, the term '*model calibration*' is more appropriate than the term '*model estimation*'. Finally, the number of parameters or unknowns can be large.

If we consider the linear regression model, we have $Y = \beta^\top X + u$ where (Y, X) forms a random vector. If we consider an observation i, we have $y_i = f(x_i) + u_i$ where $f(x_i) = \sum_{k=1}^K \beta_k x_{i,k}$. Let us now consider some non-linear features. We can replace the linear function by:

$$f(x_i) = \sum_{k=1}^K \beta_k \phi_k(x_i) = \beta^\top \phi(x_i)$$

For example, we can use quadratic, cubic or piecewise features. We abandon the framework of Gaussian conditional distribution, which is the basis of linear regression, and the reference to the random variables X and Y is not necessary. This means that the calibrated parameters $\left(\hat{\beta}_1, \ldots, \hat{\beta}_K\right)$ are less relevant. Only the calibrated function $\hat{f}(x)$ is important. For instance, if we use radial basis functions:

$$\phi_k(x) = \exp\left(-\frac{1}{2}\left\|x - c_k\right\|^2\right)$$

where c_k is the centering parameter, we obtain:

$$\hat{f}(x_i) = \sum_{k=1}^K \hat{\beta}_k e^{-\frac{1}{2}\|x_i - c_k\|^2}$$

Even if $\hat{f}(x)$ is a parametric function, it can be considered as a non-parametric model. Indeed, the functional form is the relevant quantity, not the parameters.

15.2.3.1 k-nearest neighbor classifier

The k-NN algorithm is one of the simplest non-parametric models. Let $\{(x_i, y_i)\}$ be the training sample of dimension n. We assume that the labels y_i can be assigned to J classes $(\mathcal{C}_1, \ldots, \mathcal{C}_J)$. The goal is to assign a label y for a given unlabeled observation x. For that, we select the k closest labeled observations in the training sample and we find the label $\hat{y}$ that appears most frequently within the k-subset. Said differently, the k-NN classifier uses the majority vote of the k closest neighbors and the classification rule depends on k, which is the hyperparameter. It is obvious that a high value of k helps to smooth the decision regions, but it increases the computational complexity. Moreover, there is a trade-off between bias and variance. If $k = 1$, we assign to x the label of the input x_i that is the closest. If $k = n$, we assign to x the most frequent label of the training sample. In the first case, we see that $\hat{y}$ is an unbiased estimator of y, but its variance is large. In the second case, the estimator is biased but it has a small variance.

The implementation of the k-NN algorithm requires defining the distance between the points x_i and x_j. Generally, we use the Euclidean distance, but we can consider the Minkowski distance. To find the k closest labeled observations, the simplest way is the brute-force approach. When the number of observations n is large, we can use more efficient methods based on tree-based partition[37].

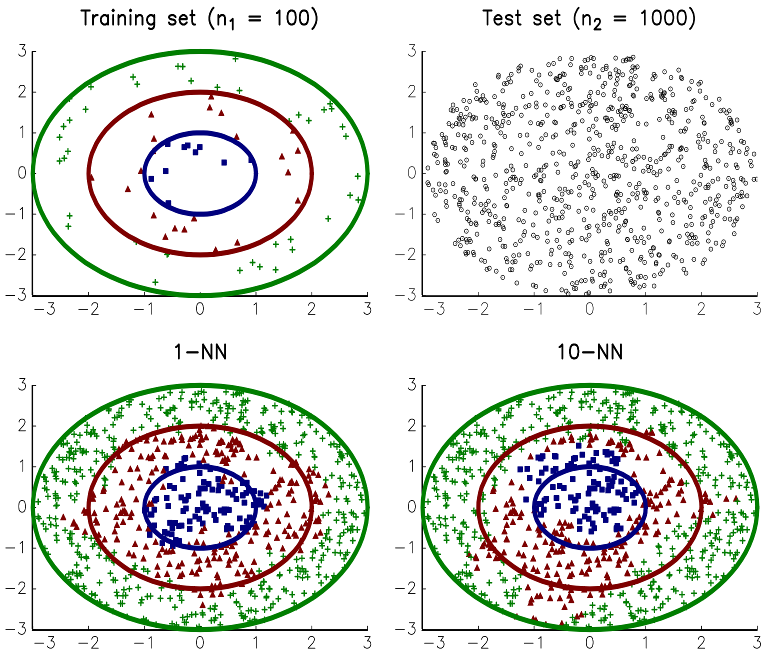

FIGURE 15.21: Illustration of the k-NN classifier

We consider the non-linearly separable classification problem, where the classes are distributed in rings around the point (0,0):

$$\mathcal{C}_j = \left\{ (x_{i,1}, x_{i,2}) \in \mathbb{R}^2 : r_{j-1}^2 < x_{i,1}^2 + x_{i,2}^2 \leq r_j^2 \right\}$$

where $j = \{1, 2, 3\}$ and $r_j = j$ is the radius of the ring. In the first panel in Figure 15.21, we have represented the three rings, and we have reported 100 simulated observations $(x_{i,1}, x_{i,2})$ that form the training set. In the second panel, we consider 1 000 observations. Solutions provided by 1-NN and 10-NN classifiers are given in the third and fourth panels. We notice that the 10-NN classifier is less efficient than the 1-NN classifier, because the number of closest neighbors is large compared to the number of observations in the training set.

Remark 188 *We can apply the k-NN algorithm to the regression. In this case, the predicted value $\hat{y}$ is the (weighted) average of the values y_i of the k closest neighbors*[38].

15.2.3.2 Neural networks

Neural networks as non-linear models We have seen that we can extend the linear model as follows[39]:

$$y_i = \beta_0 + \beta^\top \phi(x_i) + \varepsilon_i$$

In this case, we transform the input data $(x_{i,1}, \ldots, x_{i,K})$ into the auxiliary data $(z_{i,1}, \ldots, z_{i,K})$ where $z_{i,k} = \phi_k(x_{i,k})$. Here, the non-linearity property is introduced thanks

[37]The two most famous methods are the K-D tree and ball tree algorithms.
[38]The weight is generally inversely proportional to the distance.
[39]Here, we include a constant in the model.

to the non-linear function ϕ_k. However, there are many other ways to build a non-linear model. For instance, we can assume that:

$$y_i = \phi\left(\beta_0 + \beta^\top x_i\right) + \varepsilon_i$$

We first create the auxiliary data $z_i = \beta_0 + \beta^\top x_i$ from the inputs and then apply the non-linear function $\phi(x)$. If we use several non-linear functions, we obtain:

$$y_i = \sum_{j=1}^{J} \gamma_j \phi_j\left(\beta_0 + \beta^\top x_i\right) + \varepsilon_i$$

or:

$$\begin{aligned} y_i &= \varphi\left(\gamma_0 + \sum_{j=1}^{J} \gamma_j \phi_j\left(\beta_0 + \beta^\top x_i\right)\right) + \varepsilon_i \\ &= f(x_i) + \varepsilon_i \end{aligned}$$

The underlying idea of neural networks is to define a non-linear function $f(x)$, which is sufficiently flexible to fit complex relationships.

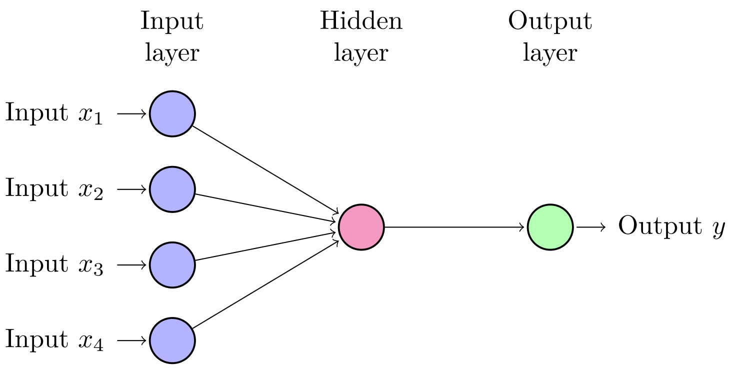

FIGURE 15.22: The perceptron

Neural networks as a mathematical representation of biological systems The term '*neural network*' makes reference to biological systems, in particular the brain. For instance, Rosenblatt (1958) proposed a self-organizing and adaptive model called the perceptron. It is no coincidence that the title of this publication is "*The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*". We have represented the perceptron in Figure 15.22. The input data are combined in order to produce an hidden variable $z = \sum_{k=1}^{K} \beta_k x_k$. Then, we apply the function $f(z)$ in order to obtain the output y:

$$y = f(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

In the context of neural networks, the function $f(z)$ is called the activation function and z is the hidden unit. If we generalize the perceptron by considering different hidden units, we obtain the artificial neural network described in Figure 15.23. In this example, we have four input units, five hidden units and one output unit. This model is also called a feed-forward neural network with one hidden layer. It can be extended in two directions.

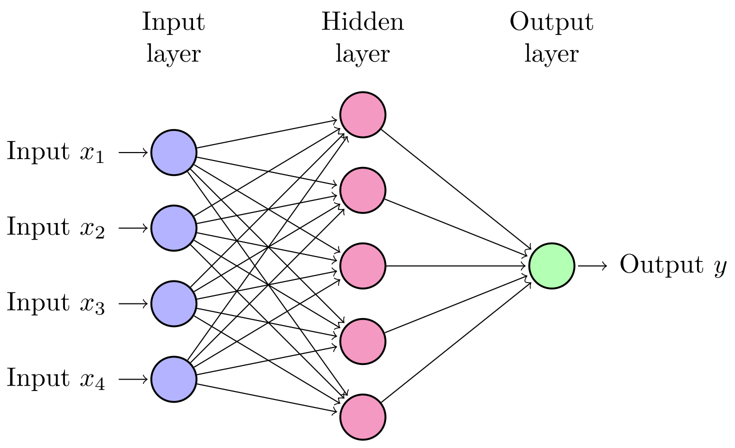

FIGURE 15.23: Feed-forward neural network with a single hidden layer

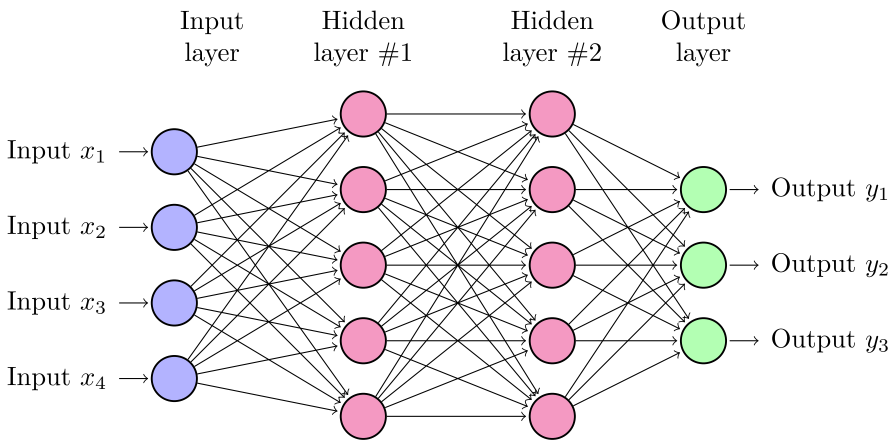

FIGURE 15.24: Feed-forward neural network with two hidden layers and three output units

First, we can consider several output units. Second, we can use several hidden layers. In this case, we speak about multi-layer neural networks (Figure 15.24). For example, deep learning refers to a neural network with a large number of hidden layers.

The term neural network does not only refer to the structure input layer – hidden layer – output layer. Activation functions generally map the resulting values into the range $[0,1]$ or $[-1,1]$ and correspond to sigmoidal functions. For example, the perceptron uses the Heaviside step function $f(z) = \mathbb{1}\{z > 0\}$, because it indicates if the neuron is activated or not. We can also use the sign function $f(z) = \operatorname{sign}(z)$ in order to indicate a '*positive*' or '*negative*' potential. However, the most popular activation functions are continuous:

1. the logistic function is equal to:
$$f(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z} \tag{15.16}$$
we have $f(z) \in [0,1]$, meaning that we can interpret $f(z)$ as a probability function; moreover, it is symmetric about 0.5;

2. the hyperbolic tangent function is defined by:
$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^{2z}-1}{e^{2z}+1} \tag{15.17}$$
and we have $f(z) \in [-1,1]$;

3. the rectified linear unit (ReLU) function corresponds to:
$$f(z) = \max(0,z) \tag{15.18}$$
and we have $f(z) \in [0,\infty)$.

Furthermore, neural networks are also characterized by the concept of learning algorithms. Neural networks can be seen as non-linear functions with some unknown parameters. The first idea is then to estimate the parameters by minimizing the residual sum of squares, meaning that neural networks are just a particulate case of non-linear least squares. However, neural networks generally use other techniques for identifying the parameters. Statistical learning implicitly refers to human brains or natural neural networks. The concept of learning is then central and shall contrast with the concept of optimization. The latter implies that there is one solution. In an artificial neural network, each node represents a neuron and each connection can be seen as a synapse. Since these connections transmit a signal from one neuron to another, the underlying idea is that they learn like in a human brain. This is why the parameters that control these connections are updated until the artificial neural network has learnt. In fact, the difference between optimization and learning is somewhat forced. Indeed, optimization also uses iterative algorithms that can be interpreted as learning algorithms. However, the learning algorithms that are used in artificial neural networks try to imitate the learning process of human brains[40]. They are also called adaptive learning rules in order to say that they are adaptive and they try to learn.

Remark 189 *According to Bishop (2006), the term neural network "has been used very broadly to cover a wide range of different models, many of which have been the subject of exaggerated claims regarding their biological plausibility". In fact, we use neural networks as non-linear regression models in the sequel.*

[40]It is particularly true for the first generation of algorithms that were discovered before 1990s.

Structure of the canonical neural network The notations used in neural networks and machine learning are generally different than those used in statistics. In this book, we have tried to use similar homogenous notations in order to make the reading easier:

- the observation (also called the example) is denoted by i;
- the input variable uses the index k and $x_{i,k}$ is the k^{th} input variable of the i^{th} observation;
- $z_{i,h}$ is the value taken by the h^{th} hidden variable and the i^{th} observation;
- for the output variables (also called the patterns), we introduce the notation $y_j(x_i)$ to name the model output taken by the j^{th} output variable and the i^{th} observation; sometimes, we use the alternative notation $\hat{y}_{i,j}$, which is more traditional in statistical inference theory.

The number of input, hidden and output variables are respectively equal n_x, n_z and n_y. The activation functions $f_{x,z}$ and $f_{z,y}$ links respectively the x's to the z's, and the z's to the y's. In order to distinguish them, $f_{z,y}$ is also called the output scaling function. We have[41]:

$$z_{i,h} = f_{x,z}(u_{i,h}) = f_{x,z}\left(\sum_{k=1}^{n_x} \beta_{h,k} x_{i,k}\right)$$

and:

$$y_j(x_i) = f_{z,y}(v_{i,j}) = f_{z,y}\left(\sum_{h=1}^{n_z} \gamma_{j,h} z_{i,h}\right)$$

where $u_{i,h}$ and $v_{i,j}$ are the intermediary variables before the activation of the functions $f_{x,z}$ and $f_{z,y}$. Finally, we have:

$$y_j(x_i) = f_{z,y}\left(\sum_{h=1}^{n_z} \gamma_{j,h} f_{x,z}\left(\sum_{k=1}^{n_x} \beta_{h,k} x_{i,k}\right)\right) \tag{15.19}$$

Figure 15.25 summarizes the structure and the notations of this neural network.

Remark 190 *Including a constant is equivalent to consider that $x_{i,1} = 1$. A variant model is to define $y_j(x_i)$ as follows:*

$$y_j(x_i) = f_{z,y}\left(\gamma_{j,0} + \sum_{h=1}^{n_z} \gamma_{j,h} f_{x,z}\left(\beta_{h,0} + \sum_{k=1}^{n_x} \beta_{h,k} x_{i,k}\right)\right) \tag{15.20}$$

In this case, we add a constant as an input variable ($\beta_{h,0}$) and a constant as a hidden variable ($\gamma_{j,0}$). Bishop (2006) shows that this model can be written as:

$$y_j(x_i) = f_{z,y}\left(\sum_{h=0}^{n_z} \gamma_{j,h} f_{x,z}\left(\sum_{k=0}^{n_x} \beta_{h,k} x_{i,k}\right)\right)$$

where $x_{i,0} = 1$. The other possibility is to have a direct link between the x's to the y's or skip-layer connections:

$$y_j(x_i) = f_{z,y}\left(\gamma_{j,0} + \sum_{h=1}^{n_z} \gamma_{j,h} f_{x,z}\left(\beta_{h,0} + \sum_{k=1}^{n_x} \beta_{h,k} x_{i,k}\right) + \sum_{k=1}^{n_x} \gamma_{j,n_z+k} x_{i,k}\right) \tag{15.21}$$

[41]Most of the time, we use the same activation function $f_{x,z}(u) = f_{z,y}(u) = f(u)$.

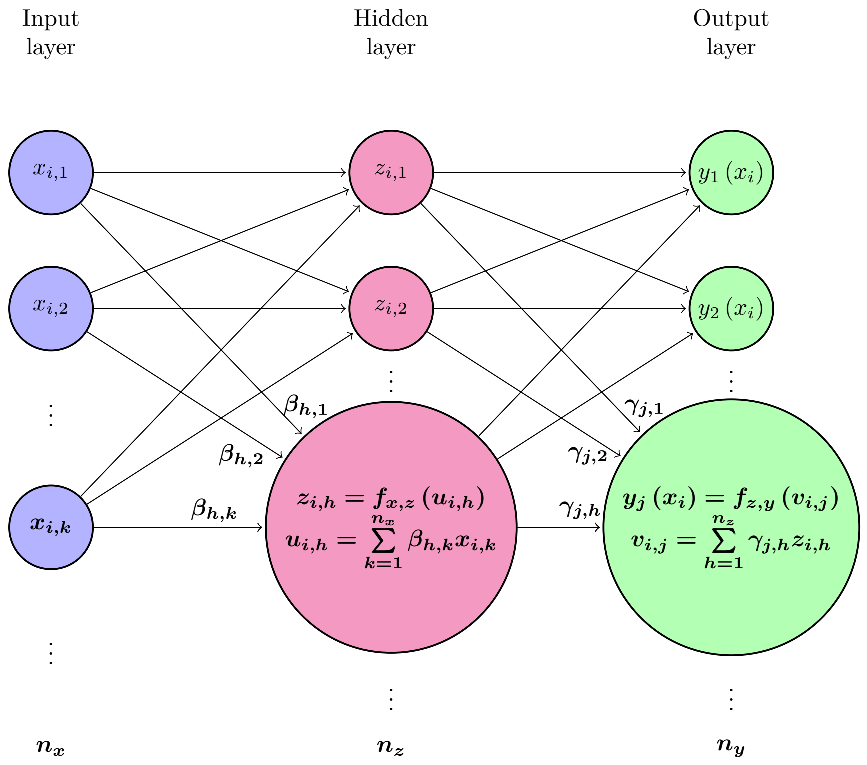

FIGURE 15.25: Canonical neural network

Loss function If we note $y_{i,j}$ the value of the output variable that is observed[42], we would like to verify:

$$y_j(x_i) = y_{i,j}$$

It follows that a natural loss function is the sum of squared errors:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{n_y} \frac{1}{2} \left(y_j(x_i) - y_{i,j}\right)^2 \tag{15.22}$$

where θ is the vector of parameters and n is the number of observations. Minimizing this loss function is also equivalent to maximize the log-likelihood function associated to the non-linear regression model:

$$y_{i,j} = y_j(x_i) + \varepsilon_{i,j}$$

where $\varepsilon_{i,j} \sim \mathcal{N}\left(0, \sigma^2\right)$ and $\varepsilon_{i,j} \perp \varepsilon_{i',j'}$ if $i \neq i'$ or $j \neq j'$.

The previous loss function is natural when considering a non-linear regression. In the case of binary classification ($y_i = 0$ or $y_i = 1$) and if the output $y(x_i)$ represents a probability, it is better to use the cross-entropy error loss[43]:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{n} \left(y_i \ln y(x_i) + (1 - y_i) \ln(1 - y(x_i))\right) \tag{15.23}$$

[42] It is called the target value or the pattern.
[43] We skip the subscript j because we assume that $j = 1$. We have then $y_i = y_{i,1}$ and $y(x_i) = y_1(x_i)$.

The choice of the loss function depends then on the output variable, but also on the activation function. For example, the cross-entropy error loss is adapted if $f_{z,y}$ corresponds to the logistic function, but not to the hyperbolic tangent function. In the case of the multi-class classification problem, Bishop (2006) proposes to consider the following loss function:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{n} \sum_{j=1}^{n_y} y_{i,j} \ln y_j(x_i) \tag{15.24}$$

where n_y is equal to the number of classes n_C and $f_{z,y}$ corresponds to the softmax function that was previously defined in the case of the multi-logistic model:

$$\begin{aligned} y_j(x_i) &= f_{z,y}(v_{i,j}) \\ &= \frac{e^{v_{i,j}}}{\sum_{j'=1}^{n_y} e^{v_{i,j'}}} \end{aligned}$$

The loss function is then the opposite of the log-likelihood function.

Learning rules In order to minimize the loss function, we can use classical optimization algorithm[44] (Newton-Raphson, conjugate gradient, BFGS, DFP, Levenberg-Marquardt, etc.). As we have already said previously, this is not the philosophy of neural networks, and we generally prefer to use a statistical learning rule, which corresponds to an iterative algorithm:

$$\theta^{(t+1)} = \theta^{(t)} + \Delta\theta^{(t)}$$

where $\theta^{(t)}$ is the value of θ at the iteration (or epoch) t, and $\Delta\theta^{(t)}$ is the adjustment vector. The learning rule consists in defining how $\theta^{(t)}$ is updated, and is mostly based on the gradient of the loss function:

$$G(\theta) = \frac{\partial \mathcal{L}(\theta)}{\partial \theta}$$

Here are the main methods (Smith, 1993):

- The steepest descent method is defined by:

$$\Delta\theta^{(t)} = -\eta \cdot G\left(\theta^{(t)}\right)$$

  where $\eta > 0$ is the learning rate parameter. For minimizing the loss function, $\Delta\theta^{(t)}$ should go in the opposite direction of the gradient.

- For the momentum method, we have:

$$\begin{aligned} \Delta\theta^{(t)} &= -(1-\alpha_m)\,\eta \cdot G\left(\theta^{(t)}\right) + \alpha_m \cdot \Delta\theta^{(t-1)} \\ &= -\eta_m \cdot G\left(\theta^{(t)}\right) + \alpha_m \cdot \Delta\theta^{(t-1)} \end{aligned}$$

  where $\alpha_m \in [0,1]$ is the momentum weight and $\eta_m > 0$ is the momentum learning rate parameter. Therefore, the adjustment at iteration t is the weighted average of the adjustment at iteration $t-1$ and the steepest descent adjustment. The underlying idea of the term $\alpha_m \Delta\theta^{(t-1)}$ is to keep going in the previous direction. This method can speed up the algorithm because it may avoid oscillations[45].

[44] See Appendix A.1.3 on Page 1046.

[45] A better method is to consider the Nesterov approach:

$$\Delta\theta^{(t)} = -\eta_m \cdot G\left(\theta^{(t)} + \alpha_m \Delta\theta^{(t-1)}\right) + \alpha_m \cdot \Delta\theta^{(t-1)}$$

- The adaptive learning method is given by:

$$\Delta\theta^{(t)} = -\eta^{(t)} \cdot G\left(\theta^{(t)}\right)$$

where:

$$\eta^{(t)} = \begin{cases} \eta^{(t-1)} + \kappa & \text{if } G\left(\theta^{(t)}\right) \cdot K\left(\theta^{(t)}\right) \geq 0 \\ \phi \cdot \eta^{(t-1)} & \text{otherwise} \end{cases}$$

$\kappa > 0$, $0 < \phi < 1$ and $K\left(\theta^{(t)}\right) = G\left(\theta^{(t-1)}\right)$. Instead of using a fixed step η, we consider a variable step $\eta^{(t)}$ that depends on the previous value $\eta^{(t-1)}$. The variable step increases when the gradient does not change between iterations $t-1$ and t, and decreases otherwise. Another rule is to consider a moving average of the gradient:

$$\begin{aligned} K\left(\theta^{(t)}\right) &= (1-\varrho) \cdot G\left(\theta^{(t)}\right) + \varrho \cdot K\left(\theta^{(t-1)}\right) \\ &\approx (1-\varrho) \sum_{\tau=1} \varrho^{\tau-1} G\left(\theta^{(t-\tau)}\right) \end{aligned}$$

where $\varrho \in [0,1]$. In the case $\varrho = 0$, we retrieve the previous rule $K\left(\theta^{(t)}\right) = G\left(\theta^{(t-1)}\right)$.

- The adaptive learning with momentum method combines the two previous approaches:

$$\begin{aligned} \Delta\theta^{(t)} &= -(1-\alpha)\,\eta^{(t)} \cdot G\left(\theta^{(t)}\right) + \alpha \cdot \Delta\theta^{(t-1)} \\ &= -\eta_m^{(t)} \cdot G\left(\theta^{(t)}\right) + \alpha_m \cdot \Delta\theta^{(t-1)} \end{aligned}$$

There are numerous other algorithms[46] (adagrad, adam, nadam, rprop, rmsprop, etc.), and a lot of tricks for accelerating the convergence. First, we distinguish three approaches for evaluating the gradient of the objective function:

1. the batch gradient descent (BGD) computes the gradient with respect to the entire training dataset:

$$G\left(\theta^{(t)}\right) = \frac{\partial\,\mathcal{L}\left(\theta^{(t)}\right)}{\partial\,\theta}$$

2. the stochastic gradient descent (SGD) considers only one different training example at each iteration:

$$G\left(\theta^{(t)}\right) = \frac{\partial\,\mathcal{L}_i\left(\theta^{(t)}\right)}{\partial\,\theta}$$

where $\mathcal{L}_i(\theta)$ is the loss function for the i^{th} observation;

3. the mini-batch gradient descent (MGD) updates the parameters by using a subset of the training dataset:

$$G\left(\theta^{(t)}\right) = \sum_{i \in \mathcal{S}^{(t)}} \frac{\partial\,\mathcal{L}_i\left(\theta^{(t)}\right)}{\partial\,\theta}$$

where the subset $\mathcal{S}^{(t)}$ changes at each iteration.

The underlying idea is to evaluate the gradient not with respect to the current value $\theta^{(t)}$, but with respect to the prediction of the future value $\theta^{(t+1)}$. This prediction is equal to $\hat{\theta}^{(t+1)} = \theta^{(t)} + \alpha_m \Delta\theta^{(t-1)}$ in the momentum method.

[46] See Ruder (2016) for a review of recent approaches.

It is obvious that the choice of one approach depends on the size of the training data. Moreover, we better understand why the momentum approach is important when defining the learning rule. Indeed, in SGD and MGD approaches, the estimation of the gradient is more noisy than in the BGD approach. The momentum method helps to smooth the gradient and to obtain a more consistent direction.

We give here some default values that are used for the learning rules: $\eta = 1$, $\alpha_m = 0.75$, $\kappa = 0.1$, $\phi = 0.9$, $\alpha_m = 0.6$ and $\varrho = 0.5$. However, these parameters can change during the learning process. For instance, the learning rate parameter η can be greater at the beginning of the learning process, because of the large gradients. In a similar way, we can use a small momentum parameter α_m and then we can increase it progressively. We can also assume that the appropriate learning rules can vary between the parameters.

Error backpropagation In order to calculate the gradient $G(\theta)$, we consider a method called error backpropagation (or backward propagation). In the case of the loss function (15.22), we have $\mathcal{L}(\theta) = \sum_{i=1}^{n} \mathcal{L}_i(\theta)$ and:

$$\mathcal{L}_i(\theta) = \sum_{j=1}^{n_y} \frac{1}{2} \left(y_j(x_i) - y_{i,j} \right)^2$$

It follows that $G(\theta) = \sum_{i=1}^{n} G_i(\theta)$ where $G_i(\theta)$ is the gradient of $\mathcal{L}_i(\theta)$. In the case of the previous loss, we can use the decomposition $\mathcal{L}_i(\theta) = \sum_{j=1}^{n_y} \mathcal{L}_{i,j}(\theta)$ where:

$$\mathcal{L}_{i,j}(\theta) = \frac{1}{2} \left(y_j(x_i) - y_{i,j} \right)^2$$

Using chain rule, we obtain[47]:

$$\begin{aligned} \frac{\partial \mathcal{L}_{i,j}(\theta)}{\partial \gamma_{j,h}} &= \frac{\partial \mathcal{L}_{i,j}(\theta)}{\partial y_j(x_i)} \cdot \frac{\partial y_j(x_i)}{\partial v_{i,j}} \cdot \frac{\partial v_{i,j}}{\partial \gamma_{j,h}} \\ &= \left(y_j(x_i) - y_{i,j} \right) f'_{z,y}(v_{i,j}) z_{i,h} \end{aligned}$$

and $\partial_{\gamma_{j,h}} \mathcal{L}_{i,j'}(\theta) = 0$ when $j \neq j'$. We also deduce that[48]:

$$\begin{aligned} \frac{\partial \mathcal{L}_{i,j}(\theta)}{\partial \beta_{h,k}} &= \frac{\partial \mathcal{L}_{i,j}(\theta)}{\partial z_{i,h}} \cdot \frac{\partial z_{i,h}}{\partial u_{i,h}} \cdot \frac{\partial u_{i,h}}{\partial \beta_{h,k}} \\ &= \frac{\partial \mathcal{L}_{i,j}(\theta)}{\partial z_{i,h}} f'_{x,z}(u_{i,h}) x_{i,k} \\ &= \left(y_j(x_i) - y_{i,j} \right) f'_{z,y}(v_{i,j}) \gamma_{j,h} f'_{x,z}(u_{i,h}) x_{i,k} \end{aligned}$$

In the case of Model (15.20), there is a constant and we have:

$$\frac{\partial \mathcal{L}_{i,j}(\theta)}{\partial \gamma_{j,0}} = \left(y_j(x_i) - y_{i,j} \right) f'_{z,y}(v_{i,j})$$

[47] The distinction between j and j' is important when we consider the softmax function (see Exercise 15.4.7 on page 1025).

[48] Because we have:

$$\begin{aligned} \frac{\partial \mathcal{L}_{i,j}(\theta)}{\partial z_{i,h}} &= \frac{\partial \mathcal{L}_{i,j}(\theta)}{\partial y_j(x_i)} \cdot \frac{\partial y_j(x_i)}{\partial v_{i,j}} \cdot \frac{\partial v_{i,j}}{\partial z_{i,h}} \\ &= \left(y_j(x_i) - y_{i,j} \right) f'_{z,y}(v_{i,j}) \gamma_{j,h} \end{aligned}$$

and:

$$\frac{\partial \mathcal{L}_{i,j}(\theta)}{\partial \beta_{h,0}} = (y_j(x_i) - y_{i,j}) f'_{z,y}(v_{i,j}) \gamma_{j,h} f'_{x,z}(u_{i,h})$$

In the case of Model (15.21), we have for the direct links:

$$\frac{\partial \mathcal{L}_{i,j}(\theta)}{\partial \gamma_{j,n_z+k}} = (y_j(x_i) - y_{i,j}) f'_{z,y}(v_{i,j}) x_{i,k}$$

It follows that the neural network consists in two steps. The forward propagation computes $u_{i,h}$, $z_{i,h}$, $v_{i,j}$ and $y_j(x_i)$, meaning that the information comes from left to right. The backward propagation computes all the derivatives using the chain rule, implying that the information goes from right to left.

Remark 191 *All the previous quantities can be calculated in a matrix form in order to avoid loop implementation (see Exercise 15.4.7 on page 1025).*

Since $f'_{z,y}$ and $f'_{x,z}$ are easy to calculate, all the derivatives are calculated in a closed-form expression. For instance, the derivative of the logistic activation function is equal to:

$$\begin{aligned} f'(z) &= \frac{e^{-z}}{(1+e^{-z})^2} \\ &= \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) \\ &= f(z)(1-f(z)) \end{aligned}$$

It follows that $f'_{z,y}(v_{i,j}) = f_{z,y}(v_{i,j})(1 - f_{z,y}(v_{i,j})) = y_j(x_i)(1 - y_j(x_i))$ and $f'_{x,z}(u_{i,h}) = z_{i,h}(1 - z_{i,h})$. In Exercise 15.4.7 on page 1025, we consider other activation functions and loss functions.

Examples Neural networks are sufficiently flexible that they can approximate any continuous function. Therefore, they are said to be '*universal approximators*' (Bishop, 2006). Figure 15.26 illustrates this property when the function is $f(x) = 2\cos(x)$ or $f(x) = |x| - 2$. For that, we use the network structure (15.20) with two constants and direct links. The activation function $f_{x,z}$ is the hyperbolic tangent function, while the output scaling function $f_{z,y}$ is the identity function. The training step is done with 201 uniform points between -4 and $+4$. We notice that the accuracy depends on the number n_z of hidden nodes. In particular, the approximation is very good when we consider three hidden nodes. The universal approximation property is certainly the main strength of neural networks. It suffices to increase the number of hidden nodes in order to achieve a given accuracy. However, this property is also the main weakness of neural networks. Indeed, the distinction between training and validation steps is not obvious, and overfitting risk is large.

The trade-off between n_z and $\mathcal{L}(\theta)$ is not the only issue with neural networks. Another problem is the scaling of data. By applying activation functions, the output domain is not necessarily the set $\mathbb{R}^{n_y}$. In Figure 15.27, we have reported the approximation of $f(x) = |x| - 2$ by considering two hidden nodes and different configurations. The first panel corresponds to the network structure (15.19) without constant and direct link ($\beta_0 = 0$, $\gamma_0 = 0$ and $\gamma_x = 0$). In the second panel, we include the two constants β_0 and γ_0, but not the direct links ($\gamma_x = 0$). We notice that this second structure is better to approximate the function than the structure of the first panel. The reason is the range of $\operatorname{dom} f(x)$, which is better managed by including a constant γ_0. This is confirmed by the third panel. Finally, the fourth panel assumes that the output scaling function $f_{z,y}$ is the logistic sigmoid function. In this

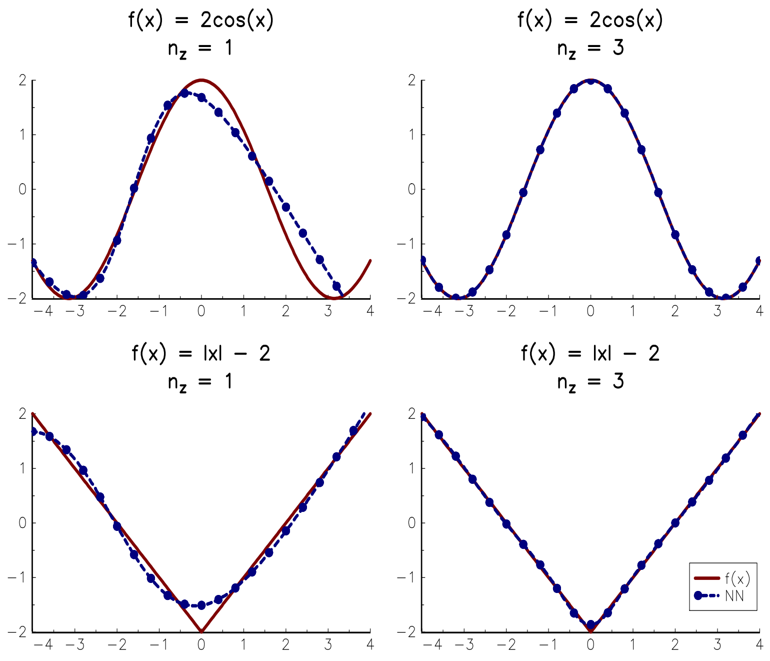

FIGURE 15.26: Neural networks as universal approximators

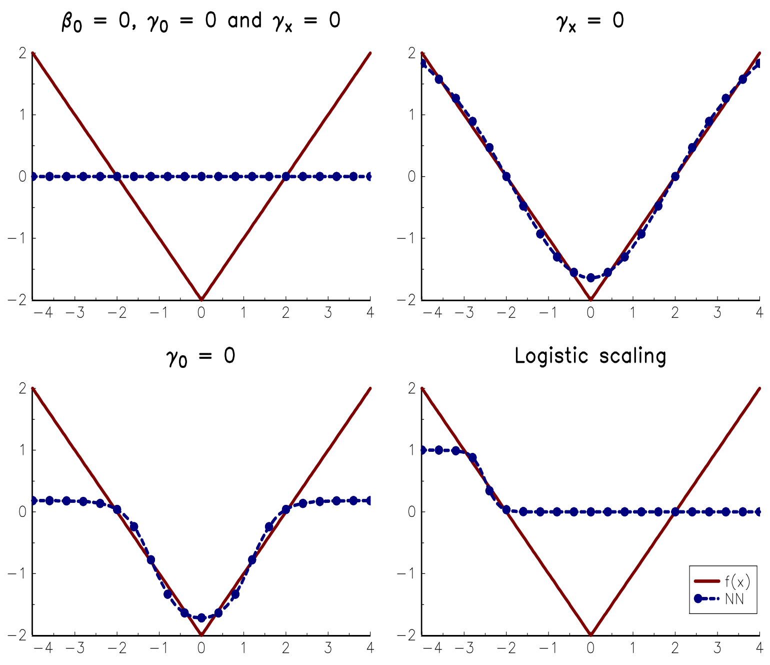

FIGURE 15.27: The scaling issue of neural networks ($f(x) = |x| - 2$)

case, it is obvious that the output $y(x_j) \in [0,1]$ cannot reach $\operatorname{dom} f(x) = [-2,2]$. This case is trivial while the three previous cases are not. This means that the network structure is crucial, not only the number of hidden units, but also the choice of activation and scaling functions, adding or not constants and direct links, and the scaling of both input and output data.

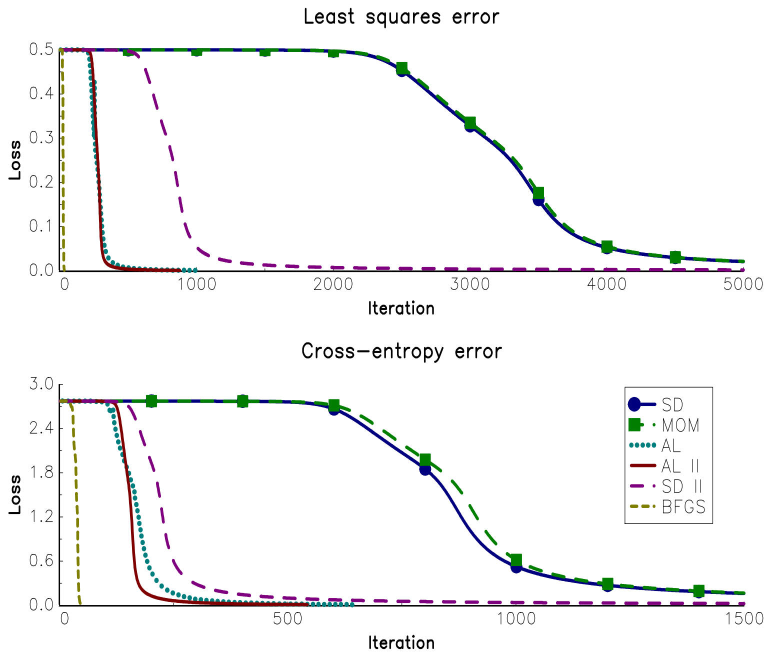

FIGURE 15.28: Convergence of the XOR problem

The XOR (or exclusive or) problem is a classic problem in neural networks. We note $y = x_1 \oplus x_2$ where x_1 and x_2 are two binary outputs:

$$\begin{cases} 0 \oplus 0 = 0 \\ 0 \oplus 1 = 1 \\ 1 \oplus 0 = 1 \\ 1 \oplus 1 = 0 \end{cases}$$

The XOR problem can be viewed as a supervised classification problem. In order to solve this problem, we use a neural network with three hidden nodes with no constant and no direct link. The activation and output scaling functions are set to the logistic function. In Figure 15.28, we have represented the evolution of the loss function $\mathcal{L}(\theta)$ with respect to the iterations of the learning rules, which are steepest descent (SD), momentum (MOM), adaptive learning (AL) and adaptive learning with momentum (AL II) methods. We also consider a steepest descent with optimal stepsize (SD II) and the BFGS algorithm. Moreover, we have used the two loss criteria: least squares and cross-entropy errors. The results show the following major lessons. First, we notice that the convergence highly depends on the learning rule, but also on the loss criterion. Second, a comparison of the optimal parameters $\hat{\theta}$ shows that they are all different. They differ from one learning rule to another, but they also differ from one loss criterion to another even if we use the same learning rule. This result is not surprising, because we observe that the solution $\hat{\theta}$ changes each time we consider new starting values. This means that neural networks produce models that are overidentified. In this context, it is perfectly illusory to analyze and understand the estimated model. As we have already said, only the predictions $\hat{y}_{i,j}$ are relevant.

TABLE 15.15: Data of program effectiveness

OBS	GPA	TUCE	PSI	GRD	OBS	GPA	TUCE	PSI	GRD
1	2.66	20	0	0	17	2.75	25	0	0
2	2.89	22	0	0	18	2.83	19	0	0
3	3.28	24	0	0	19	3.12	23	1	0
4	2.92	12	0	0	20	3.16	25	1	1
5	4.00	21	0	1	21	2.06	22	1	0
6	2.86	17	0	0	22	3.62	28	1	1
7	2.76	17	0	0	23	2.89	14	1	0
8	2.87	21	0	0	24	3.51	26	1	0
9	3.03	25	0	0	25	3.54	24	1	1
10	3.92	29	0	1	26	2.83	27	1	1
11	2.63	20	0	0	27	3.39	17	1	1
12	3.32	23	0	0	28	2.67	24	1	0
13	3.57	23	0	0	29	3.65	21	1	1
14	3.26	25	0	1	30	4.00	23	1	1
15	3.53	26	0	0	31	3.10	21	1	0
16	2.74	19	0	0	32	2.39	19	1	1

Source: Greene (2017), Table F14.1 and Spector and Mazzeo (1980).

We consider the classification problem described in Greene (2017) based on the study of Spector and Mazzeo (1980), who examined whether a new method of teaching economics, the personalized system of instruction (PSI), significantly influenced performance in later economics courses. The corresponding data are reproduced in Table 15.15. OBS is the observation, that is the student. The output variable is GRD, which corresponds to the grade increase (1) or decrease (0) indicator for the student. The explanatory variables are the constant C, the grade point average GPA, the test score on economics test TUCE, and the binary variable PSI that indicates the participation to the new teaching method. Following Greene (2017), we estimate the following logit model:

$$\begin{aligned} \Pr\{\mathrm{GRD}_i = 1\} &= \mathbf{F}\left(\beta_0 + \beta_1 \,\mathrm{GPA}_i + \beta_2 \,\mathrm{TUCE}_i + \beta_3 \,\mathrm{PSI}_i\right) \\ &= \mathbf{F}\left(x_i^\top \beta\right) \end{aligned}$$

where $\mathbf{F}$ is the cumulative distribution function of the logistic distribution. The results are reported in Table 15.16, and the value of the optimized log-likelihood function is $\boldsymbol{\ell}\left(\hat{\beta}\right) = -12.8896$. In order to challenge the logistic regression, we consider a neural network with three hidden nodes. The logistic function is used for both the activation and output scaling functions and we consider a direct link between the input variables (C, GPA, TUCE and PSI) and the output variable GRD. In Table 15.17, we have calculated the estimated probability $\hat{p}_i = \Pr\{\mathrm{GRD}_i = 1\}$ in the cases of the logit model:

$$\hat{p}_i = \mathbf{F}\left(x_i^\top \hat{\beta}^{(\mathrm{logit})}\right)$$

and the neural network:

$$\hat{p}_i = \mathbf{F}\left(\hat{\gamma}_z^{(\mathrm{nn})} \mathbf{F}\left(\hat{\beta}_x^{(\mathrm{nn})} x_i\right) + \hat{\gamma}_x^{(\mathrm{nn})} x_i\right)$$

TABLE 15.16: Results of the logistic regression

Parameter	Estimate	Standard error	t-statistic	p-value
β_0	−13.0214	4.9313	−2.6405	0.0134
β_1	2.8261	1.2629	2.2377	0.0334
β_2	0.0952	0.1415	0.6722	0.5069
β_3	2.3787	1.0646	2.2344	0.0336

TABLE 15.17: Estimated probability $\hat{p}_i = \Pr\{\text{GRD}_i = 1\}$

OBS	Logit	NN	OBS	Logit	NN
1	2.658	2.658	17	5.363	5.363
2	5.950	5.950	18	3.859	3.859
3	18.726	18.726	19	58.987	58.987
4	2.590	2.590	20	66.079	66.079
5	56.989	56.989	21	6.138	6.138
6	3.486	3.486	22	90.485	90.485
7	2.650	2.650	23	24.177	24.177
8	5.156	5.156	24	85.209	85.209
9	11.113	11.113	25	83.829	83.829
10	69.351	69.351	26	48.113	48.113
11	2.447	2.447	27	63.542	63.542
12	19.000	19.000	28	30.722	30.722
13	32.224	32.224	29	84.170	84.170
14	19.321	19.321	30	94.534	94.534
15	36.099	36.099	31	52.912	52.912
16	3.018	3.018	32	11.103	11.103

The results are surprising. The estimated probability calculated with the neural network is exactly equal to the estimated probability calculated with the logit model. If we inspect the estimated coefficient, we obtain:

$$\hat{\beta}_x^{(\text{nn})} = \begin{pmatrix} 1.0343 & 0.8482 & 1.0678 & 0.5770 \\ 0.3856 & 0.1976 & 1.4420 & 0.8744 \\ 0.1925 & 0.8791 & 2.0427 & 0.5439 \end{pmatrix}$$

$\hat{\gamma}_z^{(\text{nn})} = (-2.9240, -2.9538, -3.6783)$ and $\hat{\gamma}_x^{(\text{nn})} = (-3.4652, 2.8261, 0.0952, 2.3787)$. Moreover, the loss error is equal to $\mathcal{L}\left(\hat{\theta}\right) = 12.8896$, which is exactly the opposite of the optimized log-likelihood function. This result is not surprising because the neural network encompasses the logit model:

$$\Pr\{\text{GRD}_i = 1\} = \mathbf{F}\left(\underbrace{\gamma_z^{(\text{nn})}\mathbf{F}\left(\beta_x^{(\text{nn})} x_i\right)}_{\text{specific nn effect}} + \underbrace{\gamma_x^{(\text{nn})} x_i}_{\text{logit effect}}\right)$$

We also notice that the logit coefficients are the same than the neural network coefficients for the direct link units ($\hat{\beta}^{(\text{logit})} = \hat{\gamma}_x^{(\text{nn})}$) with the exception of the constant[49]. Let us estimate

[49]The constant is equal to −13.0214 for the logit model and −3.4652 for the neural network.

the neural network by using other starting values for the optimization step. We obtain the same probability than previously, but the estimated coefficients are not the same. We have:

$$\hat{\beta} = \begin{pmatrix} 0.4230 & 0.9108 & 0.5875 & 0.0882 \\ 0.9586 & 0.2078 & 0.7862 & 0.4852 \\ 0.7835 & 2.9180 & 8.7259 & 0.4899 \end{pmatrix}$$

$\hat{\gamma}_z^{(\mathrm{nn})} = (-4.5296, -4.3299, -4.2120)$ and $\hat{\gamma}_x^{(\mathrm{nn})} = (0.0501, 2.8261, 0.0952, 2.3787)$. Again, the neural network coefficients for the direct link units are equal to the logit coefficients with the exception of the constant. We deduce that the neural network does not differ from the logit model, because we have:

$$\hat{\beta}_0^{(\mathrm{logit})} = \gamma_z^{(\mathrm{nn})} \mathbf{F}\left(\beta_x^{(\mathrm{nn})} x_i\right) + \gamma_1^{(\mathrm{nn})}$$

This result is interesting, because it shows that the neural network did not better than the logit model, although it presents more flexibility.

Remark 192 *The previous results are explained because we optimize the cross-entropy error loss for estimating the parameters of the neural network. This implies that the logit framework is perfectly compatible with the neural network framework.*

15.2.3.3 Support vector machines

The overidentification of neural networks is an important issue and the optimization step involves an objective function, which is generally not convex with respect to the parameters. This implies that there are many local minima. Moreover, the foundation of neural networks suffers from little theoretical basis of these learning models. Like neural networks, support vector machines (SVM) can be seen as an extension of the perceptron. However, it presents nice theoretical properties and a strong geometrical framework. Once SVMs have been first developed for linear classification, they have been extended for non-linear classification and regression.

TABLE 15.18: An example of linearly separable observations

i	1	2	3	4	5	6	7
$x_{i,1}$	0.5	2.7	2.7	1.7	1.5	2.3	4.0
$x_{i,2}$	2.5	4.2	2.0	4.2	0.7	5.3	6.9
y_i	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$

i	8	9	10	11	12	13	14	15
$x_{i,1}$	6.4	7.7	8.8	7.4	6.5	8.3	6.0	5.0
$x_{i,2}$	4.5	2.2	6.0	6.5	1.7	1.3	1.3	0.5
y_i	-1	-1	-1	-1	-1	-1	-1	-1

Separating hyperplanes We consider a training set $\{(x_i, y_i), i = 1, \ldots, n\}$, where the response variable y_i can take the values -1 and $+1$. This training set is said linearly separable if there is a hyperplane $\mathcal{H} = \left\{x \in \mathbb{R}^K : f(x) = \beta_0 + x^\top \beta = 0\right\}$ such that:

$$y_i = \operatorname{sign} f(x_i)$$

This means that the hyperplane divides the affine space in two half-spaces[50] such that $\{i : y_i = +1\} \in \mathbf{H}^+$ and $\{i : y_i = -1\} \in \mathbf{H}^-$. Let us consider the example with two explanatory variables given in Table 15.18. We have represented the data $(x_{i,1}, x_{i,2})$ and the

[50]The upper half-space $\mathbf{H}^+$ is defined by $f(x) > 0$ while the lower half-space $\mathbf{H}^-$ corresponds to $f(x) < 0$.

corresponding label y_i in Figure 15.29. It is obvious that this training set is linearly separable. For example, we have reported three hyperplanes $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$ that perform a perfect classification.

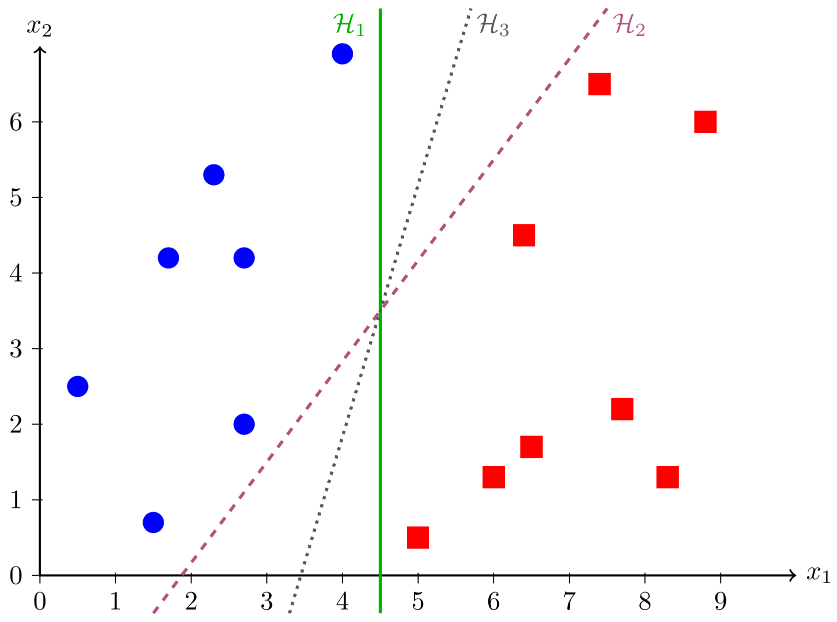

FIGURE 15.29: Separating hyperplane picking

Since there are many solutions, we may wonder if there exists one solution that dominates the others. The answer has been proposed by Vladimir Vapnik and Alexey Chervonenkis in the sixties, who have formulated the concept of support vector machines. Following Cortes and Vapnik (1995), the optimal hyperplane is the one that maximizes the margin. In Figure 15.30, we have represented an hyperplane and the two margins $M+$ and $M-$, which corresponds to the Euclidean distance between the hyperplane and the closest positive and negative points. The underlying idea of Vapnik and Chervonenkis is then to find the hyperplane $\mathcal{H}$ with the largest values of $M+$ and $M-$.

We notice that finding a hyperplane with two different margins $M_+ \neq M_-$ is equivalent to define a hyperplane with the same positive and negative margins: $M_+ = M_- = M$. This implies that the two separating hyperplanes $\mathcal{H}_+$ and $\mathcal{H}_-$ are equidistant to the hyperplane $\mathcal{H}$. The estimation of $\mathcal{H}_+$ and $\mathcal{H}_-$ requires identifying the training points that belongs to $\mathcal{H}_+$ and $\mathcal{H}_-$. These points are called the support vectors. In the case of Figure 15.30, two support vectors are necessary to define $\mathcal{H}_+$ and $\mathcal{H}_-$, or equivalently $\mathcal{H}$ and the margin M. By construction, the number of support points is at least equal to the number of explanatory variables. Except in degenerate cases, there are much less number of support points than the number of observations. This implies that not all observations are relevant for defining the decision boundary of an optimal linear classifier. Only the support vectors are important.

Hard margin classification The maximization problem is:

$$\begin{aligned} \left\{\hat{\beta}_0, \hat{\beta}\right\} &= \arg\max M \\ &\text{s.t.} \quad \left\{ \begin{array}{lll} f(x_i) \geq M & \text{if} & y_i = +1 \\ f(x_i) \leq -M & \text{if} & y_i = -1 \end{array} \right. \end{aligned}$$

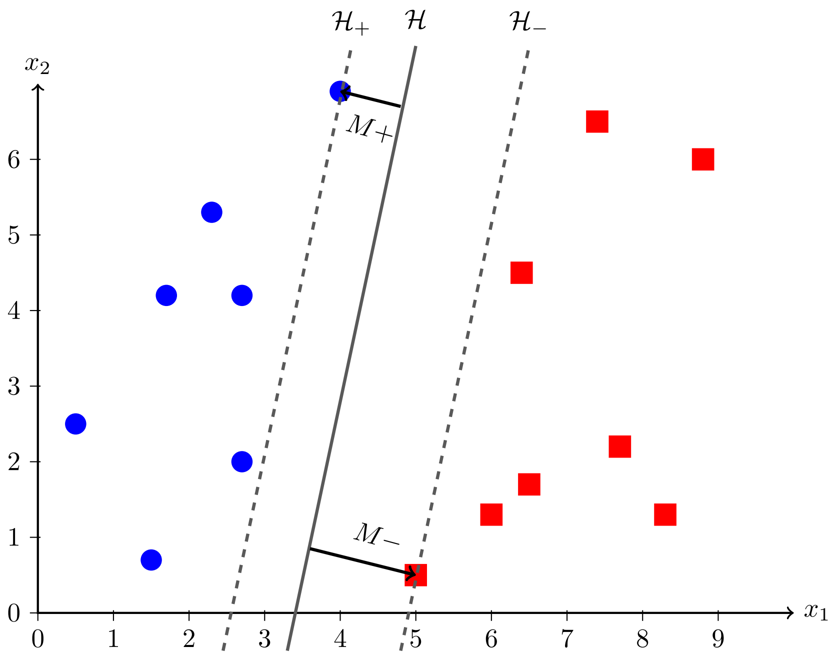

FIGURE 15.30: Margins of separation

However, this optimization problem is not well defined, since M depends on β. More precisely, it is inversely proportional to $\|\beta\|_2$. This is why we need to add another constraint, e.g. $\beta_1 = 1$ or $\|\beta\|_2 = 1$. Another approach is to standardize the problem by setting $M = 1$.

Let x_- and x_+ be two (negative and positive) support vectors, we deduce that the distance between x_- and x_+ is equal to[51]:

$$d(x_-, x_+) = \beta^\top (x_+ - x_-) = 2M$$

If we replace β by the corresponding unit vector $\hat{\beta} = \beta / \|\beta\|_2$, we obtain $\beta^\top (x_+ - x_-) = 2\hat{M} \|\beta\|_2$. By setting $M = 1$, we obtain $\hat{M} = 1 / \|\beta\|_2$. Maximizing the margin is then equivalent to maximize $1 / \|\beta\|_2$ or minimize $\|\beta\|_2$ (or $\|\beta\|_2^2$). Moreover, we notice that the inequality constraints[52] can be compacted as $y_i f(x_i) \geq 1$. Finally, we obtain the following optimization problem:

$$\begin{aligned} \left\{\hat{\beta}_0, \hat{\beta}\right\} &= \arg\min \frac{1}{2} \|\beta\|_2^2 \\ \text{s.t.} \quad & y_i \left(\beta_0 + x_i^\top \beta\right) \geq 1 \qquad \text{for } i = 1, \ldots, n \end{aligned} \tag{15.25}$$

We recognize a standard quadratic programming (QP) problem that can be easily solved from a numerical point of view.

Using the training set given in Table 15.18 on page 989, and solving the QP problem (15.25), we obtain $\hat{\beta}_0 = 2.416$, $\hat{\beta}_1 = -0.708$ and $\hat{\beta}_2 = 0.248$. It follows that the margin M is equal to 1.333. Since the equation $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = c$ is equivalent to:

$$x_2 = \frac{c - \beta_0}{\beta_2} - \frac{\beta_1}{\beta_2} x_1$$

[51] We have $\beta_0 + x_-^\top \beta = -M$ and $\beta_0 + x_+^\top \beta = M$.

[52] Because we have set $M = 1$.

we deduce that the equations of the three hyperplanes $\mathcal{H}_-$, $\mathcal{H}$ and $\mathcal{H}_+$ are:

$$\begin{array}{lll} \mathcal{H}_- : & x_2 = -13.786 + 2.857 \cdot x_1 & (c = -1) \\ \mathcal{H} : & x_2 = -9.750 + 2.857 \cdot x_1 & (c = 0) \\ \mathcal{H}_+ : & x_2 = -5.714 + 2.857 \cdot x_1 & (c = +1) \end{array}$$

We have reported the estimated hyperplanes in Figure 15.31, and have also indicated the support vectors, which are only three.

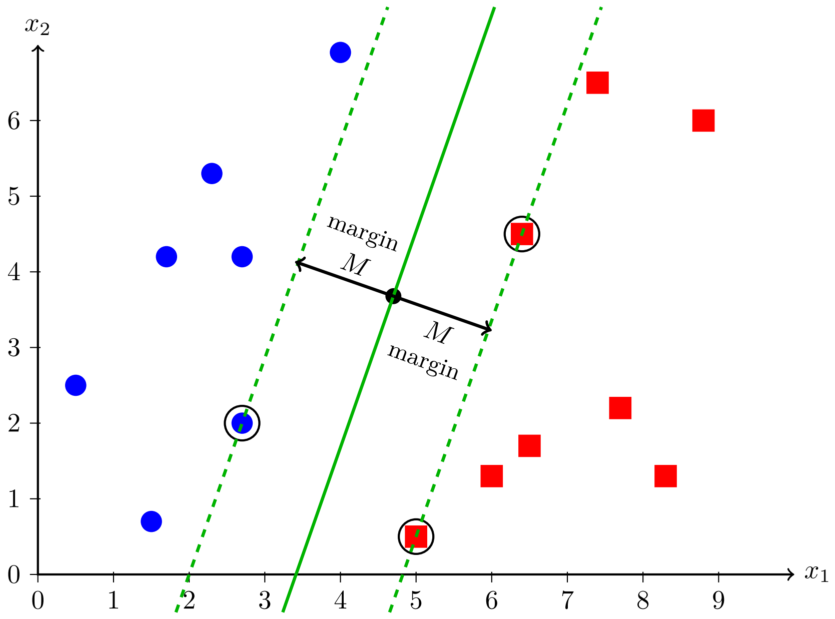

FIGURE 15.31: Optimal hyperplane

The historical approach to estimate a support vector machine is to map the primal QP problem to the dual QP problem. Using the results provided in Appendix A.1.3.1 on page 1046, we can show that[53]:

$$\begin{aligned} \hat{\alpha} &= \arg\min \frac{1}{2}\alpha^\top \Gamma \alpha - \alpha^\top \mathbf{1}_n \\ &\text{s.t. } \left\{ \begin{array}{l} y^\top \alpha = 0 \\ \alpha \geq \mathbf{0}_n \end{array} \right. \end{aligned} \tag{15.26}$$

where α is the vector of Lagrange multipliers associated to the n inequality constraints and $\Gamma_{i,j} = y_i y_j x_i^\top x_j$. Moreover, we have:

$$\hat{\beta} = \sum_{i=1}^{n} \hat{\alpha}_i y_i x_i$$

The optimal value of $\hat{\beta}_0$ can be deduced from any support vectors. In the case of a positive support vector x_+, we have $\hat{\beta}_0 = 1 - x_+^\top \hat{\beta}$, while we have $\hat{\beta}_0 = -1 - x_-^\top \hat{\beta}$ for any negative support vector x_-. Moreover, we can classify new observations by considering the following rule:

$$\hat{y} = \operatorname{sign}\left(\hat{\beta}_0 + x^\top \hat{\beta}\right)$$

[53]See Exercise 15.4.8 on page 1027.

If we consider our example, we observe that $\hat{\alpha}_i$ is different from zero for three observations: $i \in \{3, 8, 15\}$. They correspond to the three support vectors that we have found graphically. We obtain $\hat{\alpha}_3 = 0.2813$, $\hat{\alpha}_8 = 0.0435$ and $\hat{\alpha}_{15} = 0.2378$. With these values, we deduce that $\hat{\beta}_1 = -0.708$ and $\hat{\beta}_2 = 0.248$. In order to compute $\hat{\beta}_0$, we consider one of the support vectors and calculate $\hat{\beta}_0 = y_i - x_i^\top \hat{\beta}$. For example, in the case of the first support vector (or the third observation), we have: $\hat{\beta}_0 = 1 + 2.7 \times 0.708 - 2 \times 0.248 = 2.416$.

Remark 193 *We may wonder what the rational of using the dual problem is. The primal problem is a QP problem with $K + 1$ unknowns and n inequality constraints. The dual problem is a QP problem with n unknowns, one equality constraint and n box constraints. Since the last constraints are straightforward to manage, the second problem is easier to solve than the first problem. However, the dimension of the second problem is larger than this of the first problem, since we have to calculate the Γ matrix of dimension $n \times n$. Therefore, it is difficult to justify that the dual problem presents less computational issues than the primal problem. The reason is to be found elsewhere. In fact, the calculation of Γ involves the calculation of the inner product $\langle x_i, x_j \rangle = x_i^\top x_j$. We will see later that it corresponds to a covariance kernel, and the dual problem can be used in a more efficient way than the primal problem with other covariance kernels when we consider non-linear SVM problems.*

Soft margin classification The inequality constraints $y_i \left(\beta_0 + x_i^\top \beta\right) \geq 1$ ensure that all the training points are well-classified and belongs to the half-spaces $\mathbf{H}_+$ and $\mathbf{H}_-$. However, training data are generally not fully linearly separable. Therefore, we can relax these constraints by introducing slack variables $\xi_i > 0$:

$$y_i \left(\beta_0 + x_i^\top \beta\right) \geq 1 - \xi_i$$

We then face three situations:

1. if $\xi_i = 0$, the observation i is well-classified since we have $y_i \left(\beta_0 + x_i^\top \beta\right) \geq 1$;
2. if $0 < \xi_i \leq 1$, the observation i is located in the '*street*', that is in the area between the two separating planes $\mathcal{H}_-$ and $\mathcal{H}_+$; in this case, ξ_i can be interpreted as the margin error ($\xi_i \leq M$);
3. if $\xi_i > 1$, the observation i is fully misclassified.

The quality of the classification can be measured by the misclassification error sum, that we can bound:

$$\sum_{i=1}^{n} \xi_i \leq \xi^+$$

The parameter ξ^+ indicates the tolerance we have with respect to the hard margin classification. Instead of adding the inequality constraint $\sum_{i=1}^{n} \xi_i \leq \xi^+$ in Problem (15.25), we can penalize the objective function:

$$\begin{aligned} \left\{\hat{\beta}_0, \hat{\beta}, \hat{\xi}\right\} &= \arg\min \frac{1}{2} \|\beta\|_2^2 + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & y_i \left(\beta_0 + x_i^\top \beta\right) \geq 1 - \xi_i \qquad \text{for } i = 1, \ldots, n \end{aligned} \tag{15.27}$$

where the parameter C controls the level of errors. If C is large, the norm $\|\beta\|_2$ can be large. On the contrary, if C is small, the sum $\sum_{i=1}^{n} \xi_i$ can be large, but not the norm $\|\beta\|_2$.

As the margin M is equal to $1/\left\|\beta\right\|_2$, C controls then the trade-off between the size of the margin and the misclassification error rate. The dual problem is[54]:

$$\begin{aligned} \hat{\alpha} &= \arg\min \frac{1}{2}\alpha^\top \Gamma \alpha - \alpha^\top \mathbf{1}_n \qquad (15.28) \\ \text{s.t.} & \quad \left\{ \begin{array}{l} y^\top \alpha = 0 \\ \mathbf{0}_n \leq \alpha \leq C \cdot \mathbf{1}_n \end{array} \right. \end{aligned}$$

Again, we have $\hat{\beta} = \sum_{i=1}^n \hat{\alpha}_i y_i x_i$. Support vectors corresponds then to training points such that $0 < \alpha_i < C$. For computing $\hat{\beta}_0$, we average over all the support vectors:

$$\hat{\beta}_0 = \frac{\sum_{i=1}^n \mathbb{1}\left\{0 < \hat{\alpha}_i < C\right\} \cdot \left(y_i - x_i^\top \hat{\beta}\right)}{\sum_{i=1}^n \mathbb{1}\left\{0 < \hat{\alpha}_i < C\right\}}$$

Since we have $y_i\left(\beta_0 + x_i^\top \beta\right) \geq 1 - \xi_i$ and $\xi_i \geq 0$, the Kuhn-Tucker conditions implies that:

$$\hat{\xi}_i = \max\left(0, 1 - y_i\left(\hat{\beta}_0 + x_i^\top \hat{\beta}\right)\right) \qquad (15.29)$$

The classification rule does not change, and we have $\hat{y} = \operatorname{sign}\left(\hat{\beta}_0 + x^\top \hat{\beta}\right)$.

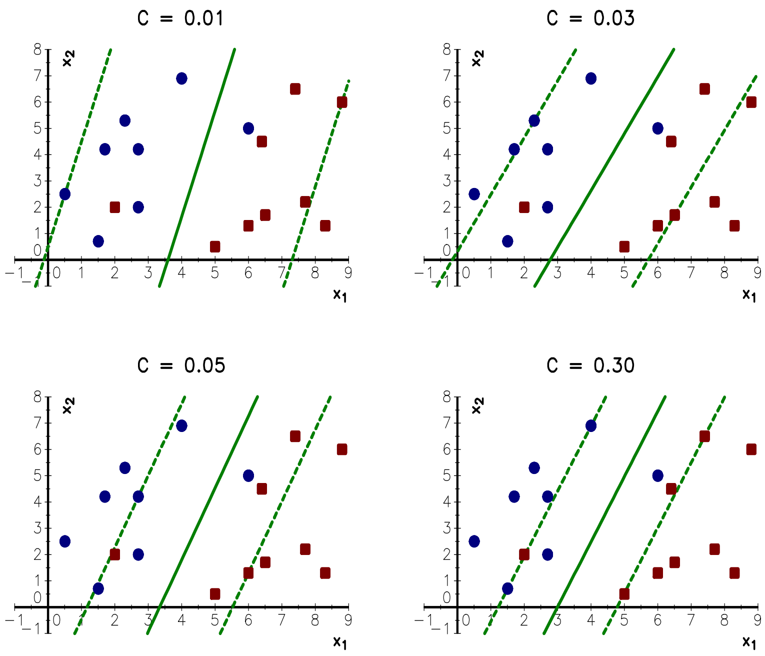

FIGURE 15.32: Soft margin SVM classifiers

We consider the previous training set given in Table 15.18 and we introduce two points $(6.0, 5.0, +1)$ $(i = 16)$ and $(2.0, 2.0, -1)$ $(i = 17)$. In this case, the training set is not linearly separable. Considering different values of C, we have represented the optimal hyperplanes in Figure 15.32. We verify that the margin decreases when C increases. In the case where C is equal to 0.05, we obtain $\hat{\beta}_0 = 1.533$, $\hat{\beta}_1 = -0.458$, $\hat{\beta}_2 = 0.168$, and the optimal value of α_i and ξ_i are reported in Table 15.19.

[54]See Exercise 15.4.8 on page 1027

TABLE 15.19: Soft margin classification with $C = 0.05$

i	y_i	$x_{i,1}$	$x_{i,2}$	$\hat{\alpha}_i$	$\hat{\xi}_i$
1	+1	0.5	2.5	0.000	0.000
2	+1	2.7	4.2	0.039	0.000
3	+1	2.7	2.0	0.050	0.369
4	+1	1.7	4.2	0.000	0.000
5	+1	1.5	0.7	0.050	0.038
6	+1	2.3	5.3	0.000	0.000
7	+1	4.0	6.9	0.050	0.143
8	−1	6.4	4.5	0.050	0.354
9	−1	7.7	2.2	0.000	0.000
10	−1	8.8	6.0	0.000	0.000
11	−1	7.4	6.5	0.050	0.231
12	−1	6.5	1.7	0.000	0.000
13	−1	8.3	1.3	0.000	0.000
14	−1	6.0	1.3	0.039	0.000
15	−1	5.0	0.5	0.050	0.324
16	+1	6.0	5.0	0.050	1.379
17	−1	2.0	2.0	0.050	1.952

If we combine Equations (15.27) and (15.29), we obtain:

$$\begin{aligned} f(\beta_0, \beta) &= \frac{1}{2}\left\|\beta\right\|_2^2 + C\sum_{i=1}^{n}\max\left(0, 1 - y_i\left(\beta_0 + x_i^\top \beta\right)\right) \\ &= C \cdot \left(\sum_{i=1}^{n}\max\left(0, 1 - y_i\left(\beta_0 + x_i^\top \beta\right)\right) + \frac{1}{2C}\left\|\beta\right\|_2^2\right) \end{aligned}$$

We deduce that the optimization program is:

$$\arg\min \mathcal{R}(x, y) + \frac{1}{2C}\left\|\beta\right\|_2^2 \tag{15.30}$$

where $\mathcal{R}(x, y) = \sum_{i=1}^{n}\mathcal{L}(x_i, y_i)$ and $\mathcal{L}(x_i, y_i)$ is the binary hinge loss:

$$\mathcal{L}(x_i, y_i) = \max\left(0, 1 - y_i\left(\beta_0 + x_i^\top \beta\right)\right)$$

It follows that the soft margin classification corresponds to a risk minimization problem with a ridge penalization. The problem is convex but non-smooth because $\mathcal{L}(x_i, y_i)$ is non-differentiable. More generally, we can use other loss functions, for instance the $0 - 1$ loss:

$$\mathcal{L}^{0-1}(x_i, y_i) = \begin{cases} 0 & \text{if } y_i\left(\beta_0 + x_i^\top \beta\right) \geq 1 \\ 1 & \text{otherwise} \end{cases}$$

However, the associated risk measure is non-convex, and the minimization problem is computationally hard. A better approach is to consider the squared hinge loss:

$$\mathcal{L}^{\text{squared}}(x_i, y_i) = \mathcal{L}^{\text{hinge}}(x_i, y_i)^2$$

In this case, the problem is convex and smooth. Another popular loss function is the ramp loss:

$$\mathcal{L}^{\text{ramp}}(x_i, y_i) = \min\left(1, \mathcal{L}^{\text{hinge}}(x_i, y_i)\right)$$

The derivation of the dual problems and the comparison of these different loss functions are discussed in Exercise 15.4.8 on page 1028.

SVM regression Support vector machines can be extended to output variables that are continuous. In this case, we have to define an appropriate loss function. For instance, if we consider the least squares loss function, we have:

$$\mathcal{L}^{\text{ls}}(x_i, y_i) = (y_i - f(x_i))^2$$

where $f(x) = \beta_0 + x^\top \beta$. The corresponding SVM regression is then:

$$\begin{aligned}\left\{\hat{\beta}_0, \hat{\beta}, \hat{\xi}\right\} &= \arg\min \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^n \xi_i^2 \\ &\text{s.t.} \quad y_i = \beta_0 + x_i^\top \beta + \xi_i \qquad \text{for } i = 1, \ldots, n\end{aligned} \tag{15.31}$$

It is obvious that ξ_i plays the role of the residual. This regression problem looks very similar to the SVM problem for the soft margin classification and the squared hinge loss function. In particular, we can show that the dual problem is[55]:

$$\begin{aligned}\hat{\alpha} &= \arg\min \frac{1}{2}\alpha^\top \left(XX^\top + \frac{1}{2C}I_n\right)\alpha - \alpha^\top Y \\ &\text{s.t.} \quad \mathbf{1}_n^\top \alpha = 0\end{aligned} \tag{15.32}$$

Once we have solved this QP problem, we can calculate the prediction for x: $\hat{y} = \hat{\beta}_0 + x^\top \hat{\beta}$.

Vapnik (1998) proposed another loss function in order to keep the formalism of the original soft margin problem:

$$\mathcal{L}^{\text{ls}}(x_i, y_i) = \mathbb{1}\{|y_i - f(x_i)| \geq \varepsilon\} \cdot (|y_i - f(x_i)| - \varepsilon)$$

where $\varepsilon > 0$. It follows that:

$$\mathcal{L}^{\text{ls}}(x_i, y_i) = \begin{cases} |y_i - f(x_i)| - \varepsilon & \text{if} \quad |y_i - f(x_i)| \geq \varepsilon \\ 0 & \text{if} \quad |y_i - f(x_i)| \leq \varepsilon \end{cases}$$

Therefore, we would like to find a hyperplane such that we don't care about the errors that are smaller than ε. We have:

$$\begin{aligned}\mathcal{L}^{\text{ls}}(x_i, y_i) &= \mathbb{1}\{y_i - f(x_i) \leq -\varepsilon\} \cdot (f(x_i) - y_i - \varepsilon) + \\ &\quad\ \mathbb{1}\{y_i - f(x_i) \geq \varepsilon\} \cdot (y_i - f(x_i) - \varepsilon) \\ &= \mathbb{1}\left\{\xi_i^- \geq 0\right\} \cdot \xi_i^- + \mathbb{1}\left\{\xi_i^+ \geq 0\right\} \cdot \xi_i^+\end{aligned}$$

where $\xi_i^- = f(x_i) - y_i - \varepsilon$ and $\xi_i^+ = y_i - f(x_i) - \varepsilon$. We deduce that the ε-SVM regression problem is:

$$\begin{aligned}\left\{\hat{\beta}_0, \hat{\beta}, \hat{\xi}^-, \hat{\xi}^+\right\} &= \arg\min \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^n \left(\xi_i^- + \xi_i^+\right) \\ &\text{s.t.} \quad \begin{cases} f(x_i) - y_i \leq \varepsilon + \xi_i^- \\ y_i - f(x_i) \leq \varepsilon + \xi_i^+ \\ \xi_i^- \geq 0 \\ \xi_i^+ \geq 0 \end{cases} \qquad \text{for } i = 1, \ldots, n\end{aligned} \tag{15.33}$$

[55]See Exercise 15.4.8 on page 1028.

We can show that the dual problem is[56]:

$$\begin{aligned}\left\{\hat{\alpha}^-,\hat{\alpha}^+\right\} &= \arg\min \frac{1}{2}\left(\alpha^- - \alpha^+\right)^\top XX^\top\left(\alpha^- - \alpha^+\right) + \\ &\quad \varepsilon\left(\alpha^- + \alpha^+\right)^\top \mathbf{1}_n + \left(\alpha^- - \alpha^+\right)^\top Y \\ \text{s.t.} &\quad \left\{\begin{array}{l}\mathbf{1}_n^\top\left(\alpha^- - \alpha^+\right) = 0 \\ \mathbf{0}_n \leq \alpha^- \leq C\cdot\mathbf{1}_n \\ \mathbf{0}_n \leq \alpha^+ \leq C\cdot\mathbf{1}_n\end{array}\right.\end{aligned} \tag{15.34}$$

where α^- and α^+ are the Lagrange multipliers of the inequality constraints. We have $\hat{\beta} = \sum_{i=1}^n\left(\hat{\alpha}_i^+ - \hat{\alpha}_i^-\right)x_i$ and:

$$\hat{\beta}_0 = \frac{1}{n_{\mathcal{SV}}}\left(\sum_{i\in\mathcal{SV}^-}\left(y_i + \varepsilon - x_i^\top\hat{\beta}\right) + \sum_{i\in\mathcal{SV}^+}\left(y_i - \varepsilon - x_i^\top\hat{\beta}\right)\right)$$

where $\mathcal{SV}^- = \left\{i : 0 < \hat{\alpha}_i^- < C\right\}$ and $\mathcal{SV}^+ = \left\{i : 0 < \hat{\alpha}_i^+ < C\right\}$ are the set of negative and positive support vectors, and $n_{\mathcal{SV}}$ is the number of support vectors.

TABLE 15.20: Comparison of OLS, LAD and SVM estimates

$\hat{\beta}_k$	OLS	LAD	LS-SVM	ε-SVM	LS-SVM	ε-SVM
			$(C=1,\varepsilon=1)$		$(C=\infty,\varepsilon=0)$	
$\hat{\beta}_0$	3.446	2.331	3.389	3.262	3.446	2.331
$\hat{\beta}_1$	1.544	1.893	1.542	1.631	1.544	1.893
$\hat{\beta}_2$	−1.645	−1.735	−1.616	−1.526	−1.645	−1.735
$\hat{\beta}_3$	2.895	2.908	2.885	2.726	2.895	2.908

We consider Example 100 on page 606, which has been used to illustrate the linear regression. In Table 15.20, we report OLS, LAD and SVM estimates for $C = 1$ and $\varepsilon = 1$. In the last two columns, we consider the limit cases, when the constant C tends to $+\infty$ and ε is equal to zero. We notice that the LS-SVM estimator converges to the OLS estimator. This is quite intuitive since we use a least squares loss function. In some sense, the LS-SVM regression can be seen as a ridge regression. When C tends to $+\infty$, the ridge penalization disappears. More curiously, the ε-SVM estimator converges to the LAD estimator. In fact, the ε-SVM regression is close to a ridge quantile regression. When ε is equal to zero, we obtain a median regression with a L_2 penalization. This is why the ε-SVM estimator converges to the LAD (or median regression) estimator.

Non-linear support vector machines As we have previously seen, we can introduce non-linearity by replacing the input data x by $\phi(x)$, where ϕ is a map from K-dimension to m-dimension non-linear feature space. In the case of SVM, we notice that the dual formulation generally requires the computation of the inner product $\langle x, x'\rangle$. This implies that we can use the same framework by replacing $\langle x, x'\rangle$ by $\langle\phi(x), \phi(x')\rangle$. Manipulating $\phi(x)$ can be tricky and not always obvious[57], because of the high dimension of the non-linear space. Sometimes, it is better to manipulate the inner product, which is called a kernel function $\mathcal{K}(x, x')$. For example, let us consider $x = (x_1, x_2)$ and $\phi(x) = \left(x_1^2, x_1x_2, x_2x_1, x_2^2\right)$. The corresponding kernel function is $\mathcal{K}(x, x') = \langle x, x'\rangle^2$. We also notice that two mapping

[56]See Exercise 15.4.8 on page 1028.
[57]The dimension m is generally much larger than the original dimension.

functions can give the same kernel. For instance, $\mathcal{K}(x,x') = \langle x,x' \rangle^2$ can be generated by $\phi(x) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$.

Since we have $\mathcal{K}(x,x') = \phi(x)^\top \phi(x')$, we see that the kernel function is symmetric (Bishop, 2006). This is the main property to define kernel functions. Another way to characterize a kernel is to verify that the Kernel (or Gram) matrix $K = (K_{i,j})$, whose elements are $K_{i,j} = \mathcal{K}(x_i,x_j)$, is positive definite. Therefore, we can directly construct kernels without specifying ϕ. For instance, $e^{c\mathcal{K}}$, $\mathcal{K}+c$, $c\mathcal{K}$ and $\mathcal{K}^d$ are also kernel functions when $c>0$ and $d \in \mathbb{N}$. If $\mathcal{K}_1$ and $\mathcal{K}_2$ are two kernels, the sum $\mathcal{K}_1+\mathcal{K}_2$ and the product $\mathcal{K}_1 \cdot \mathcal{K}_2$ are also kernel functions. The simplest kernel function is obtained by considering the identify function $\phi(x) = x$. It follows that $\langle x,x' \rangle + c$ and $\left(\langle x,x' \rangle + c\right)^d$ are also kernel functions. This last one is called the polynomial kernel and is very popular in SVM non-linear classification. Another popular kernel functions are the Gaussian (or radial basis function) kernel[58]:

$$\mathcal{K}(x,x') = \exp\left(-\frac{1}{2\sigma^2}\left\|x-x'\right\|_2^2\right)$$

and the neural network (or sigmoid) kernel:

$$\mathcal{K}(x,x') = \tanh\left(c_1 \langle x,x' \rangle + c_2\right)$$

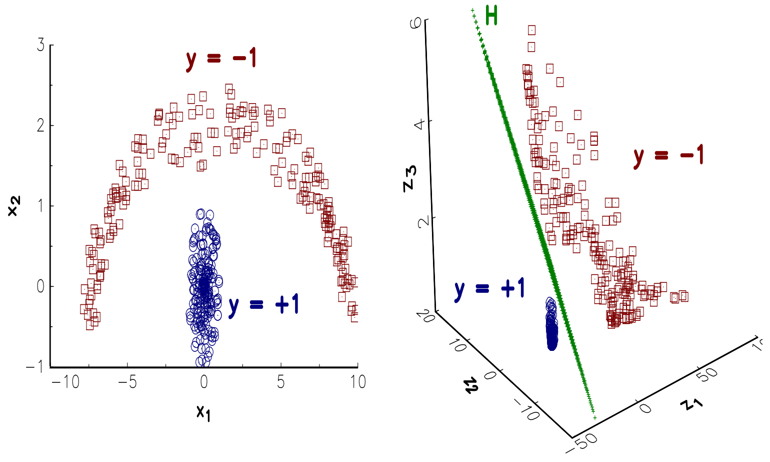

FIGURE 15.33: Transforming a non-linearly separable training set into a linearly separable training set

In order to understand the interest of kernel, we consider a training set[59], which is not linearly separable. In the left panel in Figure 15.33, we have represented the two input variables x_1 and x_2, and the response variable[60] y. Let us apply the polynomial mapping

[58]We can show that dimension of the feature space is infinite: $\phi(x) = (\phi_0(x), \ldots, \phi_s(x), \ldots, \phi_\infty(x))$ where:

$$\phi_s(x) = \left(\frac{1}{\sqrt{s!\sigma^{2s}}} e^{-\frac{x^2}{2\sigma^2}} x^s\right)$$

[59]The data are generated as follows: $x_{1,i} = c_{1,i} + r_{1,i}\cos\theta_i$ and $x_{2,i} = f_i(c_{2,i} + r_{2,i}\sin\theta_i)$ where $\theta_i \sim \mathcal{U}_{[0,2\pi]}$. In the case $y=-1$, we have $c_{1,i} = c_{2,i} = 0$, $r_{1,i} = r_{2,i} \sim \mathcal{U}_{[0,1]}$ and $f_i(x) = x$, otherwise we have $c_{1,i} = 1$, $c_{2,i} = 0$, $r_{1,i} \sim \mathcal{U}_{[8,9]}$, $r_{2,i} \sim \mathcal{U}_{[0,1]}$ and $f_i(x) = |x| - 0.5$.

[60]$y = +1$ corresponds to a circle while $y = -1$ corresponds to a square.

$z = \phi(x) = \left(x_1^2 - 10, \sqrt{2}x_1x_2, x_2^2\right)$. We have reported this transformation in the right panel in Figure 15.33. We observe that the training sets (x, y) and (z, y) are very different, since (z, y) is linearly separable.

All the previous SVM algorithms are valid in the non-linear case and we obtain the following generic framework:

1. the first step consists of defining the mapping function ϕ. Let $z_i = \phi(x_i)$ be the transformed data;

2. in the second step, we calculate the estimated parameters $\hat{\beta}_0$ and $\hat{\beta}$ in the feature space $\mathcal{Z}$;

3. finally, a new observation x is classified by computing $\hat{y} = \operatorname{sign}\left(\hat{\beta}_0 + \phi(x)^\top \hat{\beta}\right)$; in the case of the SVM regression, we have $\hat{y} = \hat{\beta}_0 + \phi(x)^\top \hat{\beta}$.

The previous framework can be simplified by considering the kernel function $\mathcal{K}$ instead of the mapping function ϕ. Indeed, in the dual problems, the input variables are evaluated through the inner product $\langle \phi(x_i), \phi(x_j) \rangle$, that can be replaced[61] by the kernel value $K_{i,j} = \mathcal{K}(x_i, x_j)$. The elements of the Γ matrix used in hard and soft margin QP problem becomes:

$$\begin{aligned} \Gamma_{i,j} &= y_i y_j \phi(x_i)^\top \phi(x_j) \\ &= y_i y_j K_{i,j} \end{aligned}$$

and we have $\Gamma = y \odot y^\top \odot K$ where $K = (k_{i,j})$ is the Gram matrix. Since we have $\hat{\beta} = \sum_{i=1}^n \hat{\alpha}_i y_i \phi(x_i)$ and $\hat{\beta}_0 = \sum_{j \in \mathcal{SV}} \left(y_j - \phi(x_j)^\top \hat{\beta}\right)$, we deduce that $\hat{y} = \operatorname{sign} \hat{f}(x)$ where:

$$\begin{aligned} \hat{f}(x) &= \hat{\beta}_0 + \phi(x)^\top \hat{\beta} \\ &= \sum_{j \in \mathcal{SV}} \left(y_j - \phi(x_j)^\top \sum_{i=1}^n \hat{\alpha}_i y_i \phi(x_i)\right) + \sum_{i=1}^n \hat{\alpha}_i y_i \phi(x)^\top \phi(x_i) \\ &= \sum_{j \in \mathcal{SV}} \left(y_j - \sum_{i=1}^n \hat{\alpha}_i y_i \mathcal{K}(x_j, x_i)\right) + \sum_{i=1}^n \hat{\alpha}_i y_i \mathcal{K}(x, x_i) \end{aligned}$$

for a new feature x. The estimation of $\hat{y}$ involves the computation of $\mathcal{K}(x_j, x_i)$ and $\mathcal{K}(x, x_i)$. However, this expression can be reduced because most of the estimates $\hat{\alpha}_i$ are equal to zero.

Remark 194 *The derivation of the SVM non-linear regression is similar to the framework above, because the dual problem involves the computation of $\phi(X)\phi(X)^\top$, which is exactly equal to the Gram matrix K.*

In Figure 15.33, we have shown that it was possible to transform the data in order to obtain separable training sets. For instance, the hyperplane $\mathcal{H}$, which is estimated using the hard margin classifier, is defined by:

$$0.884 - 0.142 \cdot z_1 + 0.268 \cdot z_2 - 1.422 \cdot z_3 = 0$$

or equivalently:

$$0.884 - 0.142 \cdot \left(x_1^2 - 10\right) + 0.268\sqrt{2} \cdot x_1 x_2 - 1.422 \cdot x_2^2 = 0$$

[61] The fact that we can easily substitute inner products by the Gram matrix in SVM classification and regression is called the kernel trick.

Let us now consider a Monte Carlo simulation. We assume that $X \sim \mathcal{N}(\mathbf{0}_4, I_4)$ and $Y = \operatorname{sign}(\mathcal{N}(0,1))$, meaning that there is no relationship between X and Y. We simulate 300 observations for the training set, and we compute the hard margin classifier for several kernels: linear, quadratic and cubic polynomial with $c = 0$, and RBF ($\sigma = 50$ and $\sigma = 20$). Then, we estimate the predicted value $\hat{y}_i$ for all the observations and calculate the error rate. Since Y is independent from X, the true error rate is equal to 50%, because the score is purely random. Using 500 replications, we have estimated the density function of the error rate in Figure 15.34. We notice that the linear kernel classifier is the worst method, while the RBF kernel with $\sigma = 20$ is the best method. On average, the error rate is respectively equal to 45.0%, 41.5%, 37.7%, 31.8% and 22.3%. Therefore, we have overfitted the model, and this is particularly true with the kernel approach. Indeed, if we consider a validation set, we obtain an average error rate of 50% whatever the kernel function we have used. We conclude that kernel functions are very powerful, but they can lead to large overfitting problems.

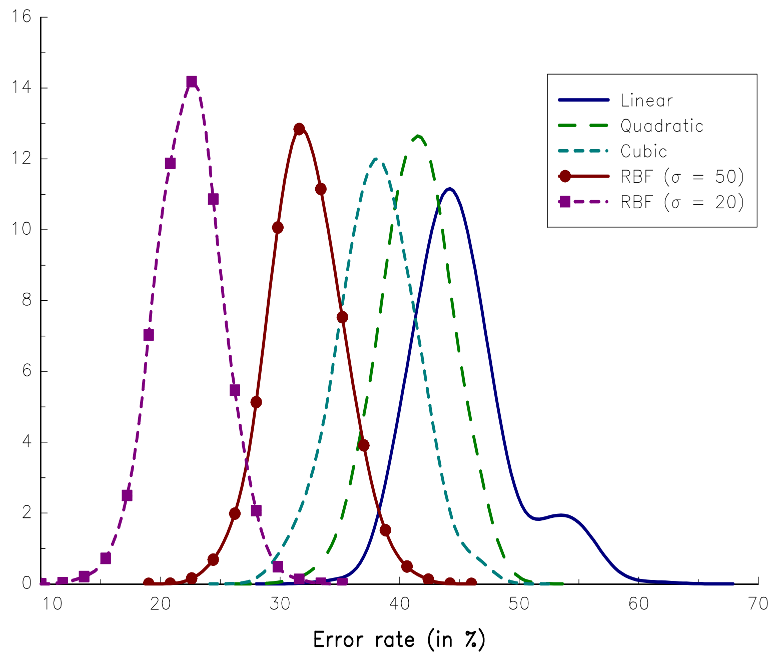

FIGURE 15.34: Probability density function of in-sample error rates

Extension to the multi-class problem We assume that we have n_C disjoint classes $\mathcal{C}_j$ where $j = 1, \ldots, J$. SVMs are inherently two-class classifiers, and the extension to the multi-class problem is not straightforward. However, we distinguish two main approaches. The first approach uses binary classification. In the case of the '*one-against-all*' strategy, we construct J single SVM classifiers in order to separate the training data from every class to the other classes. For the j^{th} classifier, the response variable is then $z_i^{(j)} = +1$ if $y_i \in \mathcal{C}_j$ and $z_i^{(j)} = -1$ if $y_i \notin \mathcal{C}_j$. Using this modified training set, we can estimate the discriminant function $\hat{f}^{(j)}(x) = \hat{\beta}_0^{(j)} + x^\top \hat{\beta}^{(j)}$. In the two-class case, we have $\hat{y} = \operatorname{sign} \hat{f}(x)$. In the multi-class problem, the prediction corresponds to the binary classifier that gives the largest value $\hat{f}^{(j)}(x)$:

$$\hat{y} \in \mathcal{C}_{j^\star} \quad \text{where} \quad j^\star = \arg\max_j \hat{f}^{(j)}(x)$$

Another approach based on the binary classification is called the '*one-against-one*' strategy. In this case, we construct $J(J-1)/2$ single SVM classifiers in order to separate the training data from the class $\mathcal{C}_j$ to the class $\mathcal{C}_{j'}$. Using the estimated discriminant function $\hat{f}^{(j|j')}(x) = \hat{\beta}_0^{(j|j')} + x^\top \hat{\beta}^{(j|j')}$, we can calculate the prediction $\hat{y}^{(j|j')} = \operatorname{sign} \hat{f}^{(j|j')}(x)$. The empirical probability that the observation belongs to the class $\mathcal{C}_j$ is then equal to[62]:

$$\hat{p}_j(x) = \frac{2\sum_{j'=1}^{J} \mathbb{1}\left\{\hat{y}^{(j|j')} = +1\right\}}{J(J-1)}$$

We deduce that the classification rule is defined as follows:

$$\hat{y} \in \mathcal{C}_{j^\star} \quad \text{where} \quad j^\star = \arg\max_j \hat{p}_j(x)$$

The second approach of multi-classification extends the mathematical framework of SVM that has been developed for the binary classification. The idea is then to consider a function $y = f(x) : \mathbb{R}^K \to \{1, \ldots, J\}$ where:

$$f(x) = \arg\max_j \beta_0^{(j)} + x^\top \beta^{(j)}$$

We have now to estimate the $J \times 1$ vector β_0 and the $K \times J$ matrix β. Crammer and Singer (2001) developed both hard and soft margin primal and dual problems in an elegant way. For a review and a comparison of these different methods, the reader can refer to Hsu and Lin (2002).

15.2.3.4 Model averaging

Model averaging (or ensemble averaging) combines multiple learning algorithms to obtain better predictive performance than could be obtained from the individual models. Two types of approaches are generally used. The first one constructs a family of '*random*' models (bagging/random forests), whereas the second one generates a family of '*adaptive*' models (boosting).

The motivation of model averaging is to replace a single expert by a committee of experts. Sometimes, it is difficult to find a skilled expert, or his search has a large cost. In this case, we can imagine that the work produced by this high skilled expert can be done by a committee of less skilled experts. For establishing the committee, we can choose (randomly) experts with similar skills or we can choose experts that are complementary. The parallel with model averaging is obvious when we distinguish random and adaptive models.

Bagging (bootstrap aggregation) Breiman (1996) proposed to use the bootstrap method to improve the performance of weak learners, in particular to reduce their variance and the overfitting bias. Given a training set $\mathcal{Z} = \{(x_i, y_i), i = 1, \ldots, n\}$, the bagging method generates n_S bootstrapped training sets $\mathcal{Z}_{(s)}$ and estimates the output function

[62]We have $\hat{y}^{(j|j')} = +1 \Leftrightarrow \hat{y}^{(j'|j)} = -1$.

$\hat{f}_{(s)}(x)$ for each training set. The s^{th} model is then defined by the pair $\left(\mathcal{Z}_{(s)}, \hat{f}_{(s)}\right)$. In the case of regression, the predicted value is the mean of the predicted values of the different models:

$$\hat{y} = \frac{1}{n_S} \sum_{s=1}^{n_S} \hat{f}_{(s)}(x)$$

In the case of classification, we generally implement the majority vote rule:

$$\hat{y} = \text{MaxVote}\left(\hat{f}_{(1)}(x), \ldots, \hat{f}_{(n_S)}(x)\right)$$

In this approach, the predictions of each model are considered as a '*vote*'. The final prediction corresponds to the class that has the maximum number of votes for multi-classification, or the majority vote for binary classification. As shown by Breiman (1996), the bagging method makes only sense when we consider non-linear models.

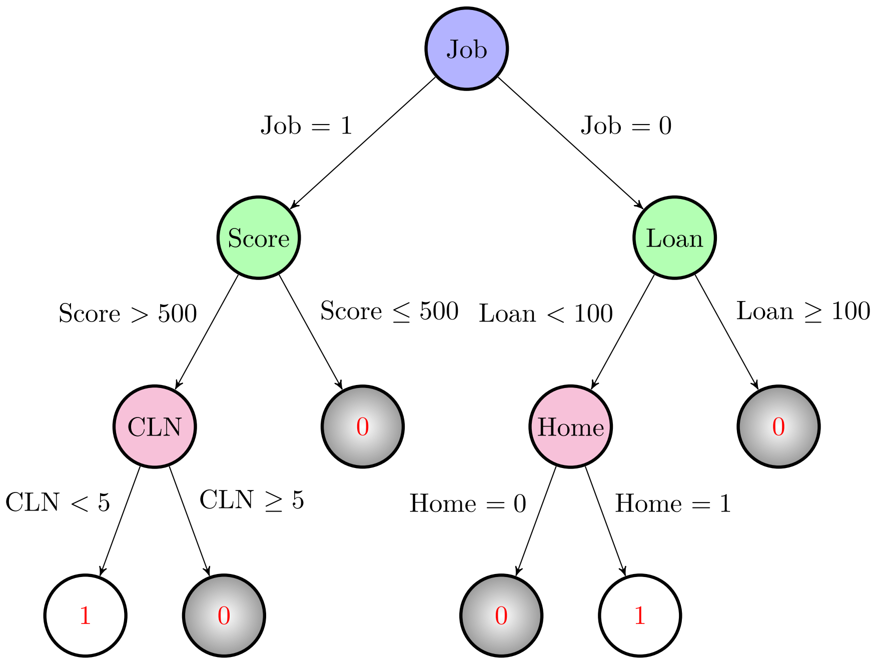

FIGURE 15.35: An example of decision tree

The bagging method is extensively used when considering decision trees. A tree is represented by a series of binary splits. Each node represents a query, except the terminal nodes that correspond to the decision nodes. In the case of a classification tree, the output variable takes a discrete set of class labels, whereas the output variable takes continuous values when considering regression trees. In Figure 15.35, we report an example of a classification tree. We consider an applicant that would like a new credit. If the applicant has not a job, the credit will be automatically refused if the amount of the loan is too high. If the amount of the loan is less than 100, the final decision will depend upon whether the applicant owns his house. In this case, the client can obtain the credit if he applies for a mortgage or a home equity line of credit. If the applicant has a job, the bank computes his credit score. If the score is less than 500, the credit is rejected. Otherwise, the final decision will depend on the number of credits. If the applicant has less than 5 credits, the new credit is accepted, otherwise it is refused. Decision trees are very popular in credit scoring for three main reasons. First, they can handle different types of variables (numeric, continuous, discrete, qualitative, etc.). Second, the rules and the decision process are very easy to understand. Third, they can be estimated with statistical models, and adjusted by experts. In practice, we use greedy approaches based on recursive binary splitting algorithms. One drawback of classification trees is that their prediction power is generally lower than the ones observed with logistic models, neural networks or support vector machines. We generally say that they produce weak classifiers (or learners). However, by combining classification trees and bagging, we can obtain the same performance than strong classifiers (Hastie *et al.*, 2009).

Remark 195 *Bagging is also extensively used when we have a large set of predictors. Instead of running one logistic regression with all the input variables, we can estimate many logit models with a limited number of explanatory variables (e.g. less than 10). In this approach, the bootstrap procedure concerns the variables, not the observations. By construction, the bagging model will produce better and more stable predictions than the single logit model*[63].

Random forests Let $\hat{Y}_{(s)} = \hat{f}_{(s)}(X)$ be the output random variable produced by the s^{th} bootstrapped model. If we assume that $\hat{Y}_{(1)}, \ldots, \hat{Y}_{(n_S)}$ are *iid* random variables with mean μ and variance σ^2, we have:

$$\operatorname{var}\left(\hat{Y}\right) = \operatorname{var}\left(\frac{1}{n_S}\sum_{s=1}^{n_S}\hat{Y}_{(s)}\right) = \frac{\sigma^2}{n_s}$$

where $\hat{Y}$ is the bagging estimator. We deduce that $\operatorname{var}\left(\hat{Y}\right) \to 0$ when $n_S \to \infty$. Theoretically, the bagging method can highly reduce the variance of the prediction. However, the hypothesis that $\hat{Y}_{(1)}, \ldots, \hat{Y}_{(n_S)}$ are not correlated is too strong. If we assume that the average correlation between bootstrapped models is equal to ρ, we obtain:

$$\begin{aligned}
\operatorname{var}\left(\hat{Y}\right) &= \mathbb{E}\left[\left(\frac{1}{n_S}\sum_{s=1}^{n_S}\left(\hat{Y}_{(s)} - \mu\right)\right)^2\right] \\
&= \mathbb{E}\left[\frac{1}{n_S^2}\sum_{s=1}^{n_S}\left(\hat{Y}_{(s)} - \mu\right)^2 + \frac{1}{n_S^2}\sum_{r \neq s}\left(\hat{Y}_{(r)} - \mu\right)\left(\hat{Y}_{(s)} - \mu\right)\right] \\
&= \frac{n_S\sigma^2}{n_S^2} + \frac{n_S\left(n_S - 1\right)\rho\sigma^2}{n_S^2} \\
&= \rho\sigma^2 + \frac{1-\rho}{n_s}\sigma^2
\end{aligned}$$

It follows that $\operatorname{var}\left(\hat{Y}\right) > \rho\sigma^2$. For example, if $\rho = 90\%$, the maximum reduction of the variance is only 10%. It follows that the improvement due to the bagging method can be highly limited when the correlation is high. Breiman (2001) proposed a modification of bagging by building de-correlated trees. At each iteration s, we select randomly a subset of predictors $\mathcal{X}_{(s)}$, implying that the model is then defined by the 3-tuple $\left(\mathcal{Z}_{(s)}, \mathcal{X}_{(s)}, \hat{f}_{(s)}\right)$. Generally, the randomization step is done with a fixed number $K^\star$ of bootstrapped predictors[64].

[63] For instance, if we consider the degenerate case when the number of observations is lower than the number of predictors ($n < K$), the single logit model is highly noisy, which is not the case of the bagging model.

[64] The recommended default value is $K^\star = \sqrt{K}$ for classification and $K^\star = K/3$ for regression (Hastie *et al.*, 2009).

Remark 196 *The method of random forests can be viewed as a double bagging method. Indeed, it mixes observation-based and feature-based bagging methods.*

Boosting In this approach, the training set $(\mathcal{Z}, \mathcal{W})$ is defined by including the weight of each observation:

$$(\mathcal{Z}, \mathcal{W}) = \{(x_i, y_i, w_i), i = 1, \ldots, n\}$$

At each iteration s, boosting computes adaptive weights $\mathcal{W}_{(s)}$ and fits the learning algorithm $\hat{f}_{(s)}$ with the training set $\left(\mathcal{Z}, \mathcal{W}_{(s)}\right)$. Then, it combines the different learning models through a weighting rule:

$$\hat{y} = \hat{f}(x) = \operatorname{Avg}\left(\omega_s \cdot \hat{f}_{(s)}(x)\right)_{s=1}^{n_S}$$

where ω_s is the weight of the s^{th} learning model and Avg is the averaging function. In the case of a binary classification, we have:

$$\hat{y} = \hat{f}(x) = \operatorname{sign}\left(\sum_{s=1}^{n_S} \omega_s \hat{f}_{(s)}(x)\right)$$

The concept of boosting has been introduced by Schapire (1990), who proved that a '*weak*' learning algorithm can be '*boosted*' into a '*strong*' learning algorithm[65]. In the 1990s, many boosting algorithms have been developed, but the high recognition comes with the adaptive boosting method proposed by Freund and Schapire (1997), and described in Algorithm 2.

The algorithm concerns the classification problem $y \in \{-1, +1\}$. We begin by initializing the observation weights w_i to $1/n$. Then, we fit the classifier $\hat{f}_{(1)}$ using the training set $\mathcal{Z}$, because the initial weights have no impact. The first step is the usual manner to fit a classification model. To improve the accuracy, boosting constructs at iteration s another training set by calculating new observation weights:

$$w_{i,s+1} = \begin{cases} w_{i,s} & \text{if } i \text{ is well-classified} \\ w_{i,s} e^{\omega_s} & \text{otherwise} \end{cases}$$

If the observation i is well-classified, the weight remain the same, otherwise it increases: $w_{i,s+1} > w_{i,s}$. Indeed, the update makes only sense if the error rate $\mathcal{L}_{(s)}$ is smaller than 50%, implying that ω_s is strictly positive[66]. At iteration $s+1$, the classifier will be fitted with the training set $\left(\mathcal{Z}, \mathcal{W}_{(s+1)}\right)$, where the misclassified observations at iteration s are more weighted than the well-classified observations. Therefore, the weighting scheme $\mathcal{W}_{(s+1)}$ forces the new classifier $\hat{f}_{(s+1)}$ to be more focus on the training observations that are difficult to classify. Finally, we use the majority vote to predict y:

$$\hat{y} = \operatorname{sign}\left(\sum_{s=1}^{n_S} \omega_s \cdot \hat{f}_{(s)}(x)\right)$$

We represent the classifier weight ω_s with respect to the loss function (or the error rate) in Figure 15.36. If the error rate is equal to 50%, the weight ω_s of the s^{th} classifier is equal to zero. This classifier does not participate to the final model, because it corresponds to a random guessing model. On the contrary, if the error rate of one classifier is equal to zero, its allocation is infinite in the final model. In Figure 15.36, we also show the impact of the

[65] A training set is said to be strongly learnable if "*there exists a polynomial-time algorithm that achieves low error with high confidence*" for all the observations (Schapire, 1990). A weak learning algorithm performs just slightly better than a random learning algorithm.

[66] If the classifier has an error rate greater than 50%, it performs worse than random guessing.

Algorithm 2 AdaBoost.M1 binary classifier

```
Estimate the AdaBoost.M1 classifier $\hat{y} = \hat{f}(x)$
Initialize the observation weights $w_{i,1} = 1/n$ for $i = 1, \ldots, n$
for $s = 1 : n_S$ do
    $\mathcal{W}_{(s)} \leftarrow (w_{1,s}, \ldots, w_{n,s})$
    Fit the classifier $\hat{f}_{(s)}$ using the training set $\left(\mathcal{Z}, \mathcal{W}_{(s)}\right)$
    Compute the loss function:
        $$\mathcal{L}_{(s)} = \frac{\sum_{i=1}^{n} w_{i,s} \cdot \mathbb{1}\left\{y_i \neq \hat{f}_{(s)}(x_i)\right\}}{\sum_{i=1}^{n} w_{i,s}}$$
    Calculate the classifier weight $\omega_s$:
        $$\omega_s \leftarrow \ln\left(\frac{1 - \mathcal{L}_{(s)}}{\mathcal{L}_{(s)}}\right)$$
    Update the observation weights:
        $$w_{i,s+1} \leftarrow w_{i,s} e^{\omega_s \cdot \mathbb{1}\left\{y_i \neq \hat{f}_{(s)}(x_i)\right\}}$$
    Normalize the observation weights:
        $$w_{i,s+1} \leftarrow \frac{w_{i,s+1}}{\sum_{i'=1}^{n} w_{i',s+1}}$$
end for
return $\hat{f}(x) = \operatorname{sign}\left(\sum_{s=1}^{n_S} \omega_s \hat{f}_{(s)}(x)\right)$
```

error rate on the weights $w_{i,s+1}$ when we consider a sample of two observations. We assume that the first observation is misclassified while the second observation is well-classified at the step s. This implies that the first observation will have more weight at the step $s+1$. The re-weighting of observations also depends on the error rate. If the error rate of the s^{th} model is low, the re-weighting is strong, in order to separate well-classified and misclassified observations. It is not obvious that the error rate is a monotonous function of the iteration s. At the beginning, the error rate can increase or decrease depending whether the initial classifier is good or bad. But, at the end, the error rate must reach the upper bound 50%.

In order to illustrate the boosting method, we consider the data given in Table 15.21 and the logit model:

$$\Pr\{y_i = 1\} = \mathbf{F}(\beta_0 + \beta_1 x_i)$$

where $\mathbf{F}(x)$ is the logit function. We have $\Pr\{y_i = -1\} = 1 - \mathbf{F}(\beta_0 + \beta_1 x_i)$. Given the pattern x, the classification rule is then:

$$\hat{y} = 2 \cdot \mathbb{1}\left\{\mathbf{F}\left(\hat{\beta}_0 + \hat{\beta}_1 x\right) > \frac{1}{2}\right\} - 1$$

where $\hat{\beta}_0$ and $\hat{\beta}_1$ are the parameters, which have been estimated by the method of maximum likelihood. Using our data, we obtain $\hat{\beta}_0 = 0.4133$ and $\hat{\beta}_1 = 0.2976$, and the error rate is equal to 45%. In the case of the boosting algorithm, the first iteration is exactly the same as the previous logit estimation. For the second iteration, we have to calculate the weights $w_{i,2}$ of each observation. We have $\omega_1 = 0.2007$ because $\mathcal{L}_{(1)} = 45\%$. Therefore, we update

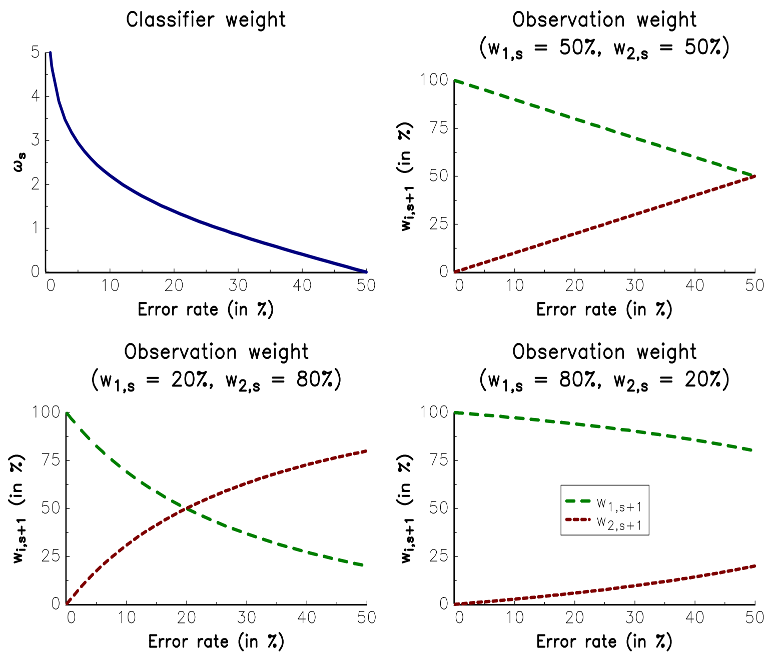

FIGURE 15.36: Weighting schemes of the boosting approach

the weights. In Table 15.21, we have reported the predicted value $\hat{y}_{i,s}$ at the iteration step s, and also the variable $\vartheta_{i,s}$ which indicates the misclassified observations. For instance, observations 3, 4, 5, 7, 9, 10, 12, 16 and 17 are not well-classified at the first iteration by the logit model. This is why the weight of these observations increase by e^{ω_1}. While the weights $w_{i,1}$ take the uniform value of 5%, the weights $w_{i,2}$ are different with respect to observations. After normalizing, $w_{i,2}$ is equal to 5.56% for observations that are misclassified at the first iteration, otherwise it is equal to 4.55%. Using these weights, we estimate the logit model, and found $\hat{\beta}_{0,2} = 0.2334$ and $\hat{\beta}_{1,2} = 0.2558$ (Table 15.22). The loss function is then equal to $\mathcal{L}_{(2)} = 38.89\%$. We see that the second logit model has improved the classification for two observations ($i = 3$ and $i = 10$). We can continue the algorithm. In our example, the boosting method stops after 5 iterations, because $\mathcal{L}_{(5)} = 50.00\%$. The fifth estimated classifier is then a pure random guessing model. While the number of well-classified observations is equal to 11 for the logit model, it is equal to 13 for the boosting model[67]. From a general point of view, the boosting is interesting only if we use a large dataset of observations and variables. When considering small datasets, we face an obvious overfitting issue.

Remark 197 *The boosting method is based on weighted estimation methods. In Chapter 10, we have already defined the weighted least squares estimator*[68]*. In Exercise 15.4.10 on page 1029, we extend the method of maximum likelihood, neural networks and support vector machines when observations are weighted.*

Hastie *et al.* (2009) showed that boosting is related to additive models:

$$g(x) = \sum_{s=1}^{n_S} \beta_{(s)} \mathcal{B}\left(x; \gamma_{(s)}\right)$$

[67]The boosting classifier corresponds to the column $\hat{y}_i$ in Table 15.21.

[68]See Section 10.1.1.5 on page 612.

TABLE 15.21: Illustration of the boosting algorithm ($n_S = 2$)

i	y_i	x_i	$w_{i,1}$ (in %)	$\hat{y}_{i,1}$	$\vartheta_{i,1}$	$w_{i,2}$ (in %)	$\hat{y}_{i,2}$	$\vartheta_{i,2}$	$\hat{y}_i$	ϑ_i
1	1	0.597	5.00	1		4.55	1		1	
2	1	1.496	5.00	1		4.55	1		1	
3	−1	−0.914	5.00	1	✓	5.56	−1		−1	
4	−1	−0.497	5.00	1	✓	5.56	1	✓	1	✓
5	−1	0.493	5.00	1	✓	5.56	1	✓	1	✓
6	1	0.841	5.00	1		4.55	1		1	
7	−1	−0.885	5.00	1	✓	5.56	1	✓	1	✓
8	1	1.418	5.00	1		4.55	1		1	
9	−1	−0.183	5.00	1	✓	5.56	1	✓	1	✓
10	−1	−1.298	5.00	1	✓	5.56	−1		−1	
11	1	−0.324	5.00	1		4.55	1		1	
12	1	−1.454	5.00	−1	✓	5.56	−1	✓	−1	✓
13	1	−0.270	5.00	1		4.55	1		1	
14	1	−0.770	5.00	1		4.55	1		1	
15	1	0.232	5.00	1		4.55	1		1	
16	−1	0.970	5.00	1	✓	5.56	1	✓	1	✓
17	−1	1.196	5.00	1	✓	5.56	1	✓	1	✓
18	1	0.578	5.00	1		4.55	1		1	
19	1	−0.686	5.00	1		4.55	1		1	
20	1	−0.590	5.00	1		4.55	1		1	

TABLE 15.22: Estimated model at each boosting iteration ($n_S = 5$)

s	1	2	3	4	5
$\hat{\beta}_{0,s}$	0.4133	0.2334	−0.0771	0.0009	0.0103
$\hat{\beta}_{1,s}$	0.2976	0.2558	0.0278	0.0277	−0.0751
$\mathcal{L}_{(s)}$	0.4500	0.3889	0.4805	0.4741	0.5000
ω_s	0.2007	0.4520	0.0780	0.1038	0.0000

where $\beta_{(s)}$ is the expansion coefficient and $\mathcal{B}\left(x;\gamma_{(s)}\right)$ is the basis function at iteration s. Forward stagewise regression consists in finding the optimal values of $\hat{\beta}_{(s)}$ and $\hat{\gamma}_{(s)}$:

$$\left(\hat{\beta}_{(s)},\hat{\gamma}_{(s)}\right)=\arg\min\sum_{i=1}^{n}\mathcal{L}\left(y_i,\hat{g}_{(s-1)}\left(x_i\right)+\beta_{(s)}\mathcal{B}\left(x;\gamma_{(s)}\right)\right)$$

where $\hat{g}_{(s)}\left(x_i\right)=\sum_{s'=1}^{s}\hat{\beta}_{(s')}\mathcal{B}\left(x;\hat{\gamma}_{(s')}\right)$ and $\mathcal{L}$ is the loss function. In the case of boosting, we can show that $\mathcal{B}\left(x;\gamma_{(s)}\right)=\hat{f}_{(s)}\left(x\right)$, $\hat{\beta}_{(s)}=\omega_s$ and $\mathcal{L}\left(y,f\left(x\right)\right)=e^{-yf(x)}$. We recognize an additive logit model with the softmax loss function. Using this framework, Friedman (2002) proposed gradient boosting models. The idea is to minimize the loss function $\sum_{i=1}^{n}\mathcal{L}\left(y_i,f\left(x_i\right)\right)$ with respect to the learning algorithm $f\left(x\right)$. The steepest descend algorithm consists in the following iterations:

$$\hat{f}_{(s)}\left(x_i\right)=\hat{f}_{(s-1)}\left(x_i\right)-\eta_{(s)}\frac{\partial\,\mathcal{L}\left(y_i,\hat{f}_{(s-1)}\left(x_i\right)\right)}{\partial\,f\left(x_i\right)}$$

Instead of finding the optimal classifier $\hat{f}_{(s)}$, gradient boosting estimates the optimal step $\eta_{(s)}$ and iterates the previous formula. Finally, the optimal model $\hat{f}\left(x\right)$ is given by the estimate $\hat{f}_{(n_S)}\left(x\right)$ at the last iteration.

Remark 198 *The table below summarizes the differences between bagging, random forests and boosting:*

Method	*Model definition*	$\mathcal{Z}_{(s)}$	$\mathcal{X}_{(s)}$	$\mathcal{W}_{(s)}$	*Weighted average*
Bagging	$\left(\mathcal{Z}_{(s)},\hat{f}_{(s)}\right)$	✓			
Random forests	$\left(\mathcal{Z}_{(s)},\mathcal{X}_{(s)},\hat{f}_{(s)}\right)$	✓	✓		
Boosting	$\left(\mathcal{Z},\mathcal{W}_{(s)},\hat{f}_{(s)}\right)$			✓	✓

In the bagging method, the randomization step concerns observations. In the case of random forests, the models are generated by randomizing both observations and variables. Boosting is a very different approach, since all the observations and variables are used to construct the weak learning models. In this method, the perturbations are introduced by using a weighting scheme for the observations that changes at each iteration. The randomization step is then replaced by an adaptive step, where the $(s+1)^{\text{th}}$ *model depends on the accuracy of the* s^{th} *model. Finally, boosting uses a weighted average of the different weak learning algorithms.*

15.3 Performance evaluation criteria and score consistency

This section is dedicated to the performance assessment of a score. Using information theory, we would like to know if the scoring system is informative or not. The second paragraph presents the graphical tools in order to measure the classification accuracy of the score. Finally, we define the different statistical measures to estimate the performance of the score. We also notice that the tools presented here can be used with both the training set or the validation set.

15.3.1 Shannon entropy

15.3.1.1 Definition and properties

The entropy is a measure of unpredictability or uncertainty of a random variable. Let (X,Y) be a random vector where $p_{i,j} = \Pr\{X = x_i, Y = y_j\}$, $p_i = \Pr\{X = x_i\}$ and $p_j = \Pr\{Y = y_j\}$. The Shannon entropy of the discrete random variable X is given by[69]:

$$H(X) = -\sum\nolimits_{i=1}^{n} p_i \ln p_i$$

We have the property $0 \leq H(X) \leq \ln n$. H is equal to zero if there is a state i such that $p_i = 1$ and is equal to $\ln n$ in the case of the uniform distribution ($p_i = 1/n$). The Shannon entropy is a measure of the average information of the system. The lower the Shannon entropy, the more informative the system. For a random vector (X,Y), we have:

$$H(X,Y) = -\sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{n} p_{i,j} \ln p_{i,j}$$

We deduce that the conditional information of Y given X is equal to:

$$\begin{aligned} H(Y \mid X) &= \mathbb{E}_X[H(Y \mid X = x)] \\ &= -\sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{n} p_{i,j} \ln \frac{p_{i,j}}{p_i} \\ &= H(X,Y) - H(X) \end{aligned}$$

We have the following properties:

- if X and Y are independent, we have $H(Y \mid X) = H(Y)$ and $H(X,Y) = H(Y) + H(X)$;
- if X and Y are perfectly dependent, we have $H(Y \mid X) = 0$ and $H(X,Y) = H(X)$.

The amount of information obtained about one random variable, through the other random variable is measured by the mutual information:

$$\begin{aligned} I(X,Y) &= H(Y) + H(X) - H(X,Y) \\ &= \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{n} p_{i,j} \ln \frac{p_{i,j}}{p_i p_j} \end{aligned}$$

Figure 15.37 shows some examples of Shannon entropy calculation. For each example, we indicate the probabilities $p_{i,j}$ and the values taken by $H(X)$, $H(Y)$, $H(X,Y)$ and $I(X,Y)$. The top/left panel corresponds to a diffuse system. The value of $H(X,Y)$ is maximum, meaning that the system is extremely disordered. The top/right panel represents a highly ordered system in the bivariate case and a diffuse system in the univariate case. We have $H(X \mid Y) = H(Y \mid X) = 0$, implying that the knowledge of X is sufficient to find the state of Y. Generally, the system is not perfectly ordered or perfectly disordered. For instance, in the case of the system described in the bottom/left panel, the knowledge of X informs us about the state of Y. Indeed, if X is in the third state, then we know that Y cannot be in the first or sixth state. Another example is provided in the bottom/right panel.

Remark 199 *If we apply the Shannon entropy to the transition matrix of a Markov chain, we set $X = \mathfrak{R}(s)$ and $Y = \mathfrak{R}(t)$ where $\mathfrak{R}(t)$ is the state variable at the date t. We obtain:*

$$H(\mathfrak{R}(t) \mid \mathfrak{R}(s)) = -\sum_{i=1}^{K} \pi_i^{\star} \sum_{j=1}^{K} p_{i,j}^{(t-s)} \ln p_{i,j}^{(t-s)}$$

where $p_{i,j} = \Pr\{\mathfrak{R}(t+1) = j \mid \mathfrak{R}(t) = i\}$, $\mathcal{S} = \{1, 2, \ldots, K\}$ is the state space of the Markov chain and $\pi^{\star}$ is the associated stationary distribution.

[69] We use the convention $p_i \ln p_i = 0$ when p_i is equal to zero.

1/36	1/36	1/36	1/36	1/36	1/36
1/36	1/36	1/36	1/36	1/36	1/36
1/36	1/36	1/36	1/36	1/36	1/36
1/36	1/36	1/36	1/36	1/36	1/36
1/36	1/36	1/36	1/36	1/36	1/36
1/36	1/36	1/36	1/36	1/36	1/36

$$H(X) = H(Y) = 1.792$$
$$H(X, Y) = 3.584$$
$$I(X, Y) = 0$$

1/6					
	1/6				
		1/6			
			1/6		
				1/6	
					1/6

$$H(X) = H(Y) = 1.792$$
$$H(X, Y) = 1.792$$
$$I(X, Y) = 1.792$$

1/24	1/24				
1/24	1/24	1/24	1/48		
	1/24	1/6	1/24	1/48	
	1/48	1/24	1/6	1/24	
		1/48	1/24	1/24	1/24
				1/24	1/24

$$H(X) = H(Y) = 1.683$$
$$H(X, Y) = 2.774$$
$$I(X, Y) = 0.593$$

					1/12
1/8			1/8		
	1/24				
5/24		1/24			
3/24				1/24	
3/24	1/24	1/24			

$$H(X) = 1.658$$
$$H(Y) = 1.328$$
$$I(X, Y) = 0.750$$

FIGURE 15.37: Examples of Shannon entropy calculation

15.3.1.2 Application to scoring

Let S and Y be the score and the control variable. For instance, Y is a binary random variable that may indicate a bad credit ($Y = 0$) or a good credit ($Y = 1$). Y may also correspond to classes defined by some quantiles. With Shannon entropy, we can measure the information of the system (S, Y). We can also compare two scores S_1 and S_2 by using the statistical measures $I(S_1, Y)$ and $I(S_2, Y)$. Let S_3 be the aggregated score obtained from the two individual scores S_1 and S_2. We can calculate the information contribution of each score with respect to the global score. Therefore, we can verify that a score really adds an information.

We consider the following decision rule:

$$\left\{ \begin{array}{l} S \leq 0 \Rightarrow S^{\star} = 0 \\ S > 0 \Rightarrow S^{\star} = 1 \end{array} \right.$$

We note $n_{i,j}$ the number of observations such that $S^\star = i$ and $Y = j$. We obtain the following system $(S^\star, Y)$:

	$Y = 0$	$Y = 1$
$S^\star = 0$	$n_{0,0}$	$n_{0,1}$
$S^\star = 1$	$n_{1,0}$	$n_{1,1}$

where $n = n_{0,0} + n_{0,1} + n_{1,0} + n_{1,1}$ is the total number of observations. The hit rate is the ratio of good bets:

$$H = \frac{n_{0,0} + n_{1,1}}{n}$$

This statistic can be viewed as an information measure of the system (S, Y). When there are more states, we can consider the Shannon entropy. In Figure 15.38, we report the contingency table of two scores S_1 and S_2 for 100 observations[70]. We have $I(S_1, Y) = 0.763$ and $I(S_2, Y) = 0.636$. We deduce that S_1 is more informative than S_2.

	y_1	y_2	y_3	y_4	y_5
s_1	10	9			
s_2	7	9			
s_3	3		7	2	
s_4		2	10	4	5
s_5				10	2
s_6			3	4	13

$H(S_1) = 1.767$
$H(Y) = 1.609$
$H(S_1, Y) = 2.614$
$I(S_1, Y) = 0.763$

	y_1	y_2	y_3	y_4	y_5
s_1	7	10			
s_2	10	8			
s_3			5	4	3
s_4	3		10	6	4
s_5	2			5	8
s_6			5	5	5

$H(S_1) = 1.771$
$H(Y) = 1.609$
$H(S_1, Y) = 2.745$
$I(S_1, Y) = 0.636$

FIGURE 15.38: Scorecards S_1 and S_2

15.3.2 Graphical methods

We assume that the control variable Y can takes two values: $Y = 0$ corresponds to a bad risk (or bad signal) while $Y = 1$ corresponds to a good risk (or good signal). Gouriéroux (1992) introduced 3 graphical tools for assessing the quality of a score: the performance curve, the selection curve and the discrimination curve[71]. In the following, we assume that the probability $\Pr\{Y = 1 \mid S \geq s\}$ is increasing with respect to the level $s \in [0, 1]$, which corresponds to the rate of acceptance. We deduce that the decision rule is the following:

- if the score of the observation is above the threshold s, the observation is selected;
- if the score of the observation is below the threshold s, the observation is not selected.

[70] Each score is divided into 6 intervals $(s_1, \ldots, s_6)$ while the dependent variable is divided into 5 intervals $(y_1, \ldots, y_5)$.

[71] See also Gouriéroux and Jasiak (2007).

If s is equal to one, we select no observation. If s is equal to zero, we select all the observations. In a scoring system, the threshold s is given. Below, we assume that s is varying and we analyze the relevance of the score with respect to this parameter.

15.3.2.1 Performance curve, selection curve and discriminant curve

The performance curve is the parametric function $y = \mathcal{P}(x)$ defined by:

$$\left\{ \begin{array}{l} x(s) = \Pr\{S \geq s\} \\ y(s) = \dfrac{\Pr\{Y = 0 \mid S \geq s\}}{\Pr\{Y = 0\}} \end{array} \right.$$

where $x(s)$ corresponds to the proportion of selected observations and $y(s)$ corresponds to the ratio between the proportion of selected bad risks and the proportion of bad risks in the population. The score is efficient if the ratio is below one. If $y(s) > 1$, the score selects more bad risks than those we can find in the population[72]. If $y(s) = 1$, the score is random and the performance is equal to zero. In this case, the selected population is representative of the total population.

The selection curve is the parametric curve $y = \mathcal{S}(x)$ defined by:

$$\left\{ \begin{array}{l} x(s) = \Pr\{S \geq s\} \\ y(s) = \Pr\{S \geq s \mid Y = 0\} \end{array} \right.$$

where $y(s)$ corresponds to the ratio of observations that are wrongly selected. By construction, we would like that the curve $y = \mathcal{S}(x)$ is located below the bisecting line $y = x$ in order to verify that $\Pr\{S \geq s \mid Y = 0\} < \Pr\{S \geq s\}$.

Remark 200 *The performance and selection curves are related as follows*[73]*:*

$$\mathcal{S}(x) = x\mathcal{P}(x)$$

The discriminant curve is the parametric curve $y = \mathcal{D}(x)$ defined by:

$$\mathcal{D}(x) = g_1\left(g_0^{-1}(x)\right)$$

where:

$$g_y(s) = \Pr\{S \geq s \mid Y = y\}$$

It represents the proportion of good risks in the selected population with respect to the proportion of bad risks in the selected population. The score is said to be discriminant if the curve $y = \mathcal{D}(x)$ is located above the bisecting line $y = x$.

[72] In this case, we have $\Pr\{Y = 0 \mid S \geq s\} > \Pr\{Y = 0\}$.

[73] We have:

$$\begin{aligned} \Pr\{S \geq s \mid Y = 0\} &= \frac{\Pr\{S \geq s, Y = 0\}}{\Pr\{Y = 0\}} \\ &= \Pr\{S \geq s\} \cdot \frac{\Pr\{S \geq s, Y = 0\}}{\Pr\{S \geq s\}\Pr\{Y = 0\}} \\ &= \Pr\{S \geq s\} \cdot \frac{\Pr\{Y = 0 \mid S \geq s\}}{\Pr\{Y = 0\}} \end{aligned}$$

15.3.2.2 Some properties

We first notice that the previous parametric curves do not depend on the probability distribution of the score S, but only on the ranking of the observations. They are then invariant if we apply an increasing function to the score. Gouriéroux (1992) also established the following properties:

1. the performance curve (respectively, the selection curve) is located below the line $y = 1$ (respectively, the bisecting line $y = x$) if and only if $\operatorname{cov}(f(Y), g(S)) \geq 0$ for any increasing functions f and g;

2. the performance curve is increasing if and only if:
$$\operatorname{cov}(f(Y), g(S) \mid S \geq s) \geq 0$$
for any increasing functions f and g, and any threshold level s;

3. the selection curve is convex if and only if $\mathbb{E}[f(Y) \mid S = s]$ is increasing with respect to the threshold level s for any increasing function f.

Remark 201 *The first property is the least restrictive. It allows us to verify that the score S is better than a random score. We can show that* (3) $\Rightarrow$ (2) $\Rightarrow$ (1). *The last property is then the most restrictive.*

A score is perfect or optimal if there is a threshold level $s^\star$ such that $\Pr\{Y = 1 \mid S \geq s^\star\} = 1$ and $\Pr\{Y = 0 \mid S < s^\star\} = 1$. It separates the population between good and bad risks. Graphically, the selection curve of a perfect score is equal to:

$$y = \mathbb{1}\{x > \Pr\{Y = 1\}\} \cdot \left(1 + \frac{x - 1}{\Pr\{Y = 0\}}\right)$$

Using the relationship $\mathcal{S}(x) = x\mathcal{P}(x)$, we deduce that the performance curve of a perfect score is given by:

$$y = \mathbb{1}\{x > \Pr\{Y = 1\}\} \cdot \left(\frac{x - \Pr\{Y = 1\}}{x \cdot \Pr\{Y = 0\}}\right)$$

For the discriminant curve, a perfect score satisfies $\mathcal{D}(x) = 1$. When the score is random, we have $\mathcal{S}(x) = \mathcal{D}(x) = x$ and $\mathcal{P}(x) = 1$. In Figure 15.39, we have reported the performance, selection and discriminant curves of a given score S. We also show the curves obtained with an optimal (or perfect) score and a random score. A score must be located in the area between the curve computed with a random score and the curve computed with a perfect score, except if the score ranks the observations in a worst way than a random score.

Gouriéroux (1992) also established two properties for comparing two scores S_1 and S_2:

- the score S_1 is more performing on the population P_1 than the score S_2 on the population P_2 if and only if the performance (or selection) curve of (S_1, P_1) is below the performance (or selection) curve of (S_2, P_2);

- the score S_1 is more discriminatory on the population P_1 than the score S_2 on the population P_2 if and only if the discriminant curve of (S_1, P_1) is above the discriminant curve of (S_2, P_2).

Figure 15.40 illustrates the case where the score S_1 is better than the score S_2. However, the order is only partial. Most of the time, the two scores cannot be globally compared. An example is provided in Figure 15.41. The second score is not very good to distinguish good and bad risks when it takes small values, but it is close to a perfect score when it takes high values.

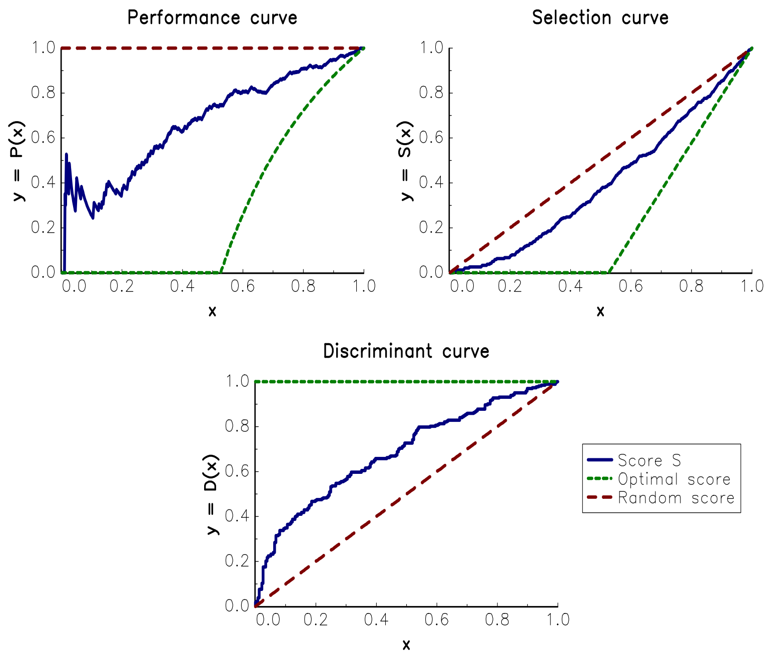

FIGURE 15.39: Performance, selection and discriminant curves

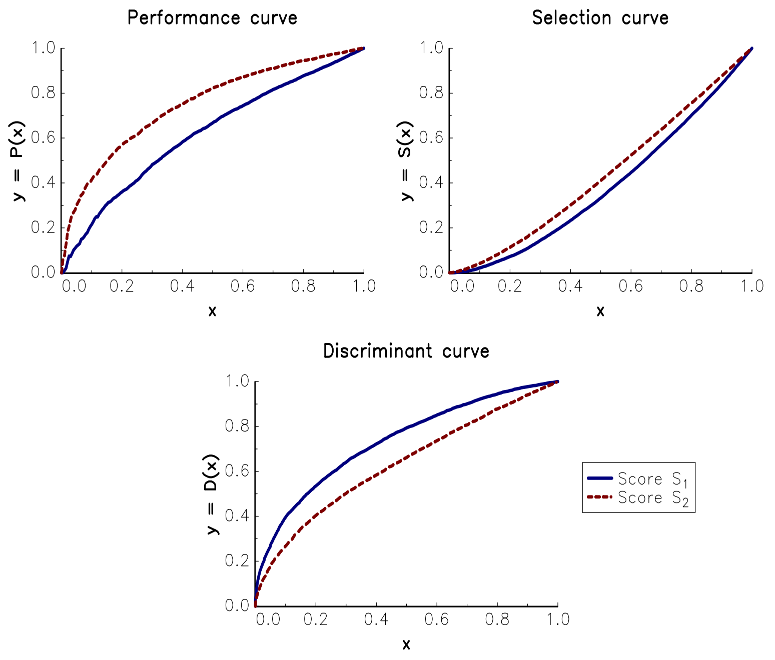

FIGURE 15.40: The score S_1 is better than the score S_2

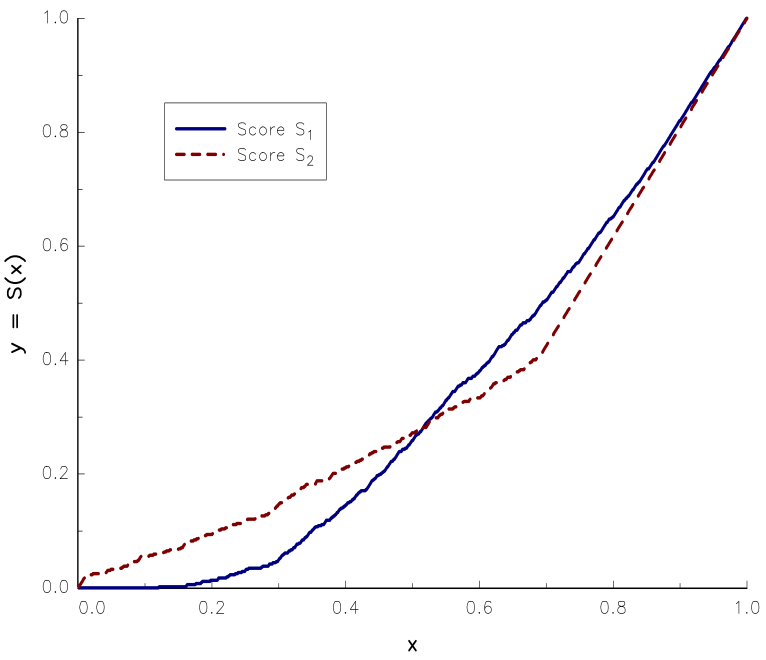

FIGURE 15.41: Illustration of the partial ordering between two scores

15.3.3 Statistical methods

Since the quantitative tools for comparing two scores are numerous, we focus on two non-parametric measures: the Kolmogorov-Smirnov test and the Gini coefficient.

15.3.3.1 Kolmogorov-Smirnov test

We consider the cumulative distribution functions:

$$\mathbf{F}_0(s) = \Pr\{S \leq s \mid Y = 0\}$$

and:

$$\mathbf{F}_1(s) = \Pr\{S \leq s \mid Y = 1\}$$

The score S is relevant if we have the stochastic dominance order $\mathbf{F}_0 \succ \mathbf{F}_1$. In this case, the score quality is measured by the Kolmogorov-Smirnov statistic:

$$\text{KS} = \max_s |\mathbf{F}_0(s) - \mathbf{F}_1(s)|$$

It takes the value 1 if the score is perfect. The KS statistic may be used to verify that the score is not random. We then test the assumption $\mathcal{H}_0 : \text{KS} = 0$ by using the tabulated critical values[74] In Figure 15.42, we give an example with 5 000 observations. The KS statistic is equal to 36%, which implies that $\mathcal{H}_0$ is rejected at the confidence level 1%.

[74]The critical values at the 5% confidence level are equal to:

n	10	50	100	500	5000
CV	40.9%	18.8%	13.4%	6.0%	1.9%

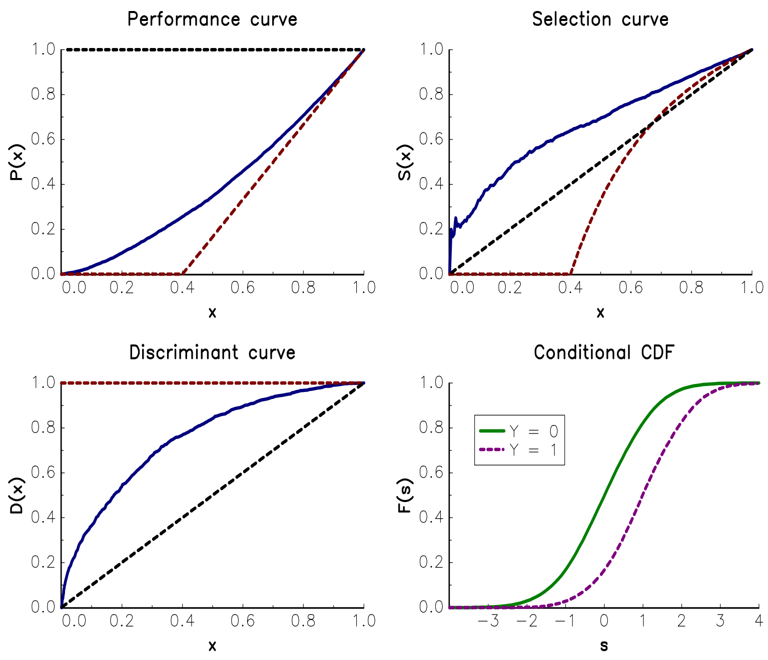

FIGURE 15.42: Comparison of the distributions $\mathbf{F}_0(s)$ and $\mathbf{F}_1(s)$

15.3.3.2 Gini coefficient

The Lorenz curve The Gini coefficient is the statistic, which is the most used for measuring the performance of a score. It is related to the concept of Lorenz curve, which is a graphical representation of the concentration. Let X and Y be two random variables. The Lorenz curve $y = \mathcal{L}(x)$ is the parametric curve defined by:

$$\begin{cases} x = \Pr\{X \leq x\} \\ y = \Pr\{Y \leq y \mid X \leq x\} \end{cases}$$

In economics, x represents the proportion of individuals that are ranked by income while y represents the proportion of income. In this case, the Lorenz curve is a graphical representation of the distribution of income and is used for illustrating inequality of the wealth distribution between individuals. For example, we observe that 70% of individuals have only 34% of total income in Figure 15.43.

Definition of the Gini coefficient The Lorenz curve has two limit cases. If the wealth is perfectly concentrated, one individual holds 100% of the total wealth. If the wealth is perfectly allocated between all the individuals, the corresponding Lorenz curve is the bisecting line. We define the Gini coefficient by:

$$\mathcal{G}ini(\mathcal{L}) = \frac{A}{A+B}$$

where A is the area between the Lorenz curve and the curve of perfect equality, and B is the area between the curve of perfect concentration and the Lorenz curve. By construction, we have $0 \leq \mathcal{G}ini(\mathcal{L}) \leq 1$. The Gini coefficient is equal to zero in the case of perfect equality and one in the case of perfect concentration. We have:

$$\mathcal{G}ini(\mathcal{L}) = 1 - 2\int_0^1 \mathcal{L}(x)\,\mathrm{d}x$$

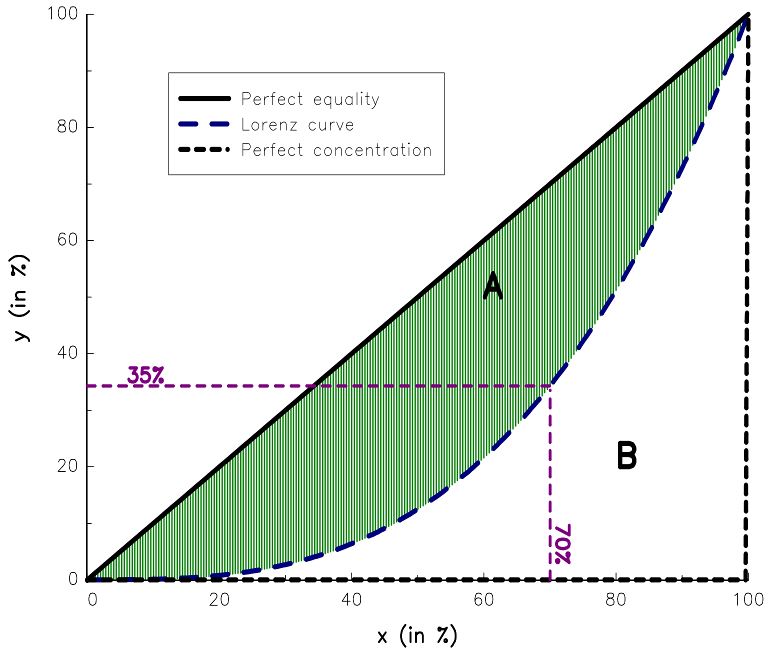

FIGURE 15.43: An example of Lorenz curve

Application to credit scoring We can interpret the selection curve as a Lorenz curve. We recall that $\mathbf{F}(s) = \Pr\{S \leq s\}$, $\mathbf{F}_0(s) = \Pr\{S \leq s \mid Y = 0\}$ and $\mathbf{F}_1(s) = \Pr\{S \leq s \mid Y = 1\}$. The selection curve is defined by the following parametric coordinates:

$$\left\{\begin{array}{l} x(s) = 1 - \mathbf{F}(s) \\ y(s) = 1 - \mathbf{F}_0(s) \end{array}\right.$$

The selection curve measures the capacity of the score for not selecting bad risks. We could also build the Lorenz curve that measures the capacity of the score for selecting good risks:

$$\left\{\begin{array}{l} x(s) = \Pr\{S \geq s\} = 1 - \mathbf{F}(s) \\ y(s) = \Pr\{S \geq s \mid Y = 1\} = 1 - \mathbf{F}_1(s) \end{array}\right.$$

It is called the precision curve. Another popular graphical tool is the receiver operating characteristic (or ROC) curve (Powers, 2011), which is defined by:

$$\left\{\begin{array}{l} x(s) = \Pr\{S \geq s \mid Y = 0\} = 1 - \mathbf{F}_0(s) \\ y(s) = \Pr\{S \geq s \mid Y = 1\} = 1 - \mathbf{F}_1(s) \end{array}\right.$$

An example for a given score S is provided in Figure 15.44. For all the three curves, we can calculate the Gini coefficient. Since the precision and ROC curves are located above the bisecting line, the Gini coefficient associated to the Lorenz curve $\mathcal{L}$ becomes[75]:

$$\mathcal{G}ini(\mathcal{L}) = 2\int_0^1 \mathcal{L}(x)\,\mathrm{d}x - 1$$

[75] An alternative to the Gini coefficient is the AUC measure, which corresponds to the area under the ROC curve. However, they give the same information since they are related by the equation:

$$\mathcal{G}ini(\text{ROC}) = 2 \times \text{AUC}(\text{ROC}) - 1$$

The Gini coefficient of the score S is then computed as follows:

$$\mathcal{G}ini^{\star}(S) = \frac{\mathcal{G}ini(\mathcal{L})}{\mathcal{G}ini(\mathcal{L}^{\star})}$$

where $\mathcal{L}^{\star}$ is the Lorenz curve associated to the perfect score.

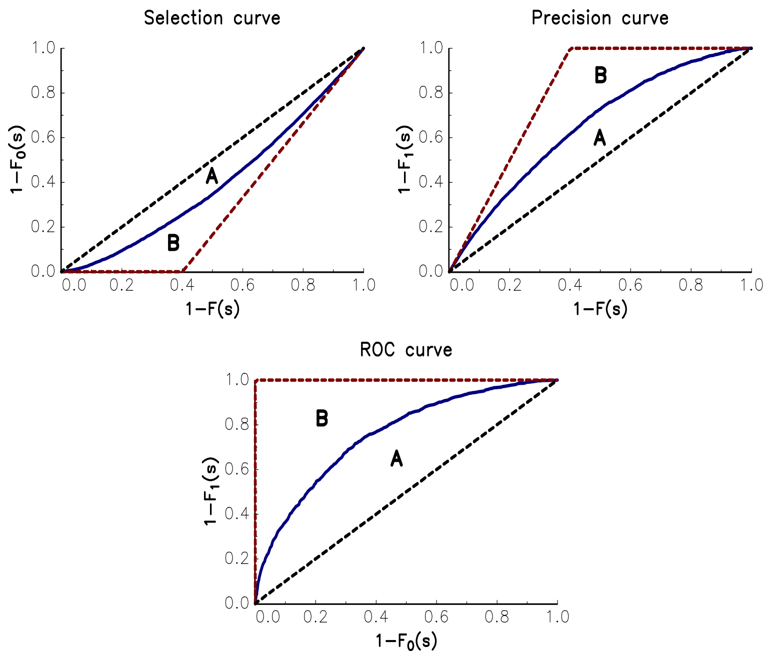

FIGURE 15.44: Selection, precision and ROC curves

Remark 202 *The Gini coefficient is not necessarily the same for the three curves. However, if the population is homogeneous, we generally obtain very similar figures*[76]*.*

15.3.3.3 Choice of the optimal cut-off

The choice of the optimal cut-off $s^{\star}$ depends on the objective function. For instance, we can calibrate $s^{\star}$ in order to achieve a minimum market share. We can also fix a given selection rate. More generally, the objective function can be the profitability of the activity. From a statistical point of view, we must distinguish the construction of the scoring model and the decision rule. In statistical learning, we generally consider three datasets: the training set, the validation set and the test set. The training set is used for calibrating the model and its parameters whereas the validation set helps to avoid overfitting. But the decision rule is based on the test set.

[76]For instance, we obtain the following results with the score S that has been used in Figure 15.44:

Curve	$\mathcal{G}ini(\mathcal{L})$	$\mathcal{G}ini(\mathcal{L}^{\star})$	$\mathcal{G}ini^{\star}(S)$
Selection	20.41%	40.02%	51.01%
Precision	30.62%	59.98%	51.05%
ROC	51.03%	100.00%	51.03%

Confusion matrix A confusion matrix is a special case of contingency matrix. Each row of the matrix represents the frequency in a predicted class while each column represents the frequency in an actual class. Using the test set, it takes the following form:

	$Y = 0$	$Y = 1$
$S < s$	$n_{0,0}$	$n_{0,1}$
$S \geq s$	$n_{1,0}$	$n_{1,1}$
	$n_0 = n_{0,0} + n_{1,0}$	$n_1 = n_{0,1} + n_{1,1}$

where $n_{i,j}$ represents the number of observations of the cell (i, j). We notice that each cell of this table can be interpreted as follows:

	$Y = 0$	$Y = 1$
$S < s$	It is rejected and it is a bad risk (true negative)	It is rejected, but it is a good risk (false negative)
$S \geq s$	It is accepted, but it is a bad risk (false positive)	It is accepted and it is a good risk (true positive)
	(negative)	(positive)

The cells $(S < s, Y = 0)$ and $(S \geq s, Y = 1)$ correspond to observations that are well-classified: true negative (TN) and true positive (TP). The cells $(S \geq s, Y = 0)$ and $(S < s, Y = 1)$ correspond to two types of errors:

1. a false positive (FP) can induce a future loss, because it may default: this is a type I error;
2. a false negative (FN) potentially corresponds to a loss of a future P&L[77]: this is a type II error.

Classification ratios Binary classification defines many metrics for measuring the performance of the classifier[78] (Fawcett, 2006):

$$\begin{array}{ll} \text{True Positive Rate} & \text{TPR} = \dfrac{\text{TP}}{\text{TP} + \text{FN}} \\ \text{False Negative Rate} & \text{FNR} = \dfrac{\text{FN}}{\text{FN} + \text{TP}} = 1 - \text{TPR} \\ \text{True Negative Rate} & \text{TNR} = \dfrac{\text{TN}}{\text{TN} + \text{FP}} \\ \text{False Positive Rate} & \text{FPR} = \dfrac{\text{FP}}{\text{FP} + \text{TN}} = 1 - \text{TNR} \end{array}$$

The true positive rate (TPR) is also known as the sensitivity or the recall. It measures the proportion of real good risks that are correctly predicted good risk. Fawcett (2006) also defines the precision or the positive predictive value (PPV):

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

[77]This is an opportunity cost.

[78] We rewrite the confusion matrix as follows:

	$Y = 0$	$Y = 1$
$S < s$	TN	FN
$S \geq s$	FP	TP
	N = TN + FP	P = FN + TP

It measures the proportion of predicted good risks that are correctly real good risk. Besides these metrics, statisticians also use two generic metrics:

1. the accuracy considers the classification of both negatives and positives:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FN} + \mathrm{TN} + \mathrm{FP}}$$

2. the F_1 score is the harmonic mean of precision and sensitivity:

$$\begin{aligned} F_1 &= \frac{2}{1/\text{precision} + 1/\text{sensitivity}} \\ &= \frac{2 \cdot \mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} \end{aligned}$$

Example 171 *We consider three scoring systems that have been calibrated on a training set. These systems produce a score between* 0 *and* 1 000*. A low value predicts a bad risk while a high value predicts a good risk. In order to calibrate the cut-off, we consider a test set, which is composed of* 10 000 *new observations. In Table 15.23, we report the confusion matrix of each scoring system for different cut-off values (*100*,* 200 *and* 500*).*

TABLE 15.23: Confusion matrix of three scoring systems and three cut-off values s

Score	$s = 100$		$s = 200$		$s = 500$	
S_1	386	616	698	1 304	1 330	3 672
	1 614	7 384	1 302	6 696	670	4 328
S_2	372	632	700	1 304	1 386	3 616
	1 628	7 368	1 300	6 696	614	4 384
S_3	382	616	656	1 344	1 378	3 624
	1 618	7 384	1 344	6 656	622	4 376
Perfect	1 000	0	2 000	0	2 000	3 000
	1 000	8 000	0	8 000	0	5 000

Using confusion matrices given in Table 15.23, we calculate the different classification ratios and report them in Table 15.24. In addition to the three scoring systems, we have also considered a perfect score in order to show what the best value is for each classification ratio. Finally, we indicate the best scoring system in Table 15.25. We notice that it depends on the ratio and on the value of the cut-off. For instance, if we want to maximize the true positive ratio or minimize the false negative ratio, S_1 is the best scoring system for low value of s while S_2 is better when s is equal to 500. For the other ratios, S_1 seems to be the best system when $s = 100$, otherwise S_2 dominates S_1 and S_3 when $s = 200$ or $s = 500$.

Remark 203 *We recall that* $\mathbf{F}_0(s) = \Pr\{S \leq s \mid Y = 0\}$ *and* $\mathbf{F}_1(s) = \Pr\{S \leq s \mid Y = 1\}$*. We deduce that* $\mathrm{TNR} = \mathbf{F}_0(s)$*,* $\mathrm{FNR} = \mathbf{F}_1(s)$*,* $\mathrm{FPR} = 1 - \mathbf{F}_0(s)$ *and* $\mathrm{TPR} = 1 - \mathbf{F}_1(s)$*. Therefore, the ROC curve is the parametric curve, where the x-coordinates are the false positive rates and the y-coordinates are the true positive rates. Generally, we note α and β the type I and II errors. We may also interpret the ROC curve as the relationship of* $1 - \beta(s)$ *with respect to* $\alpha(s)$*.*

TABLE 15.24: Binary classification ratios (in %) of the three scoring systems

Score	s	TPR	FNR	TNR	FPR	PPV	ACC	F_1
	100	92.3	7.7	19.3	80.7	82.1	77.7	86.9
S_1	200	83.7	16.3	34.9	65.1	83.7	73.9	83.7
	500	54.1	45.9	66.5	33.5	86.6	56.6	66.6
	100	92.1	7.9	18.6	81.4	81.9	77.4	86.7
S_2	200	83.7	16.3	35.0	65.0	83.7	74.0	83.7
	500	54.8	45.2	69.3	30.7	87.7	57.7	67.5
	100	92.3	7.7	19.1	80.9	82.0	77.7	86.9
S_3	200	83.2	16.8	32.8	67.2	83.2	73.1	83.2
	500	54.7	45.3	68.9	31.1	87.6	57.5	67.3
	100	100.0	0.0	50.0	50.0	88.9	90.0	94.1
Perfect	200	100.0	0.0	100.0	0.0	100.0	100.0	100.0
	500	62.5	37.5	100.0	0.0	100.0	70.0	76.9

TABLE 15.25: Best scoring system

Cut-off	TPR	FNR	TNR	FPR	PPV	ACC	F_1
100	S_1/S_3	S_1/S_3	S_1	S_1	S_1	S_1	S_1
200	S_1/S_2	S_1/S_2	S_2	S_2	S_2	S_2	S_2
500	S_2	S_2	S_2	S_2	S_2	S_2	S_2

15.4 Exercises

15.4.1 Elastic net regression

We consider the standard linear model:

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{U}$$

where $\mathbf{Y}$ is a $n \times 1$ vector, $\mathbf{X}$ is a $n \times K$ matrix and $\mathbf{U} \sim \mathcal{N}\left(0, \sigma^2 \mathbf{I}_n\right)$. Let $\hat{\beta}$ be the estimator of β, that is the solution of the following least squares problem:

$$\hat{\beta} = \arg\min \frac{1}{2} \left(\mathbf{Y} - \mathbf{X}\beta\right)^\top \left(\mathbf{Y} - \mathbf{X}\beta\right) + \frac{\lambda}{2} \left(\alpha \left\|\beta\right\|_1 + (1-\alpha) \left\|\beta\right\|_2^2\right)$$

where $\lambda \geq 0$ and $\alpha \in [0, 1]$.

1. We consider the case $\alpha = 0$, which corresponds to the ridge regression.
   (a) Find the optimal estimator $\hat{\beta}^{\text{ridge}}$.
   (b) What is the relationship between the ridge estimator $\hat{\beta}^{\text{ridge}}$ and the ordinary least squares $\hat{\beta}^{\text{ols}}$?
   (c) Deduce the expression of $\mathbb{E}\left[\hat{\beta}^{\text{ridge}}\right]$. Show that $\hat{\beta}^{\text{ridge}}$ is a biased estimator except if $\lambda = 0$.
   (d) Demonstrate that the covariance matrix of $\hat{\beta}^{\text{ridge}}$ is equal to:

$$\operatorname{var}\left(\hat{\beta}^{\text{ridge}}\right) = \sigma^2 \left(\mathbf{X}^\top \mathbf{X} + Q\right)^{-1}$$

where Q is a matrix to determine. Deduce that:

$$\operatorname{var}\left(\hat{\beta}^{\text{ols}}\right) \succeq \operatorname{var}\left(\hat{\beta}^{\text{ridge}}\right)$$

where $\succeq$ is the positive definite ordering.

(e) Let $\hat{\mathbf{Y}}$ be the predicted values of $\mathbf{Y}$. If $\hat{\mathbf{Y}} = \mathbf{H}\mathbf{Y}$, the model degree of freedom is equal to the trace of $\mathbf{H}$. Show that the degree of freedom of the ridge model is equal to:

$$\text{df}^{\text{model}} = \sum_{k=1}^{K} \frac{s_k^2}{s_k^2 + \lambda}$$

where $(s_1, \ldots, s_K)$ are the singular values of $\mathbf{X}$ (Hastie *et al.*, 2009).

(f) What does the previous results become when $\mathbf{X}$ is an orthonormal matrix?

2. We consider the case $\alpha > 0$, which corresponds to the elastic net regression (Zou and Hastie, 2005).

(a) Write the corresponding QP program.

(b) Consider the data of Example 164 on page 936. Compare the estimates $\hat{\beta}$ when α is respectively equal to 0, 0.25, 0.5 and 1.0.

15.4.2 Cross-validation of the ridge linear regression

We consider the ridge estimator:

$$\hat{\beta} = \arg\min \frac{1}{2} \left(\mathbf{Y} - \mathbf{X}\beta\right)^\top \left(\mathbf{Y} - \mathbf{X}\beta\right) + \frac{\lambda}{2} \beta^\top \beta$$

where $\mathbf{Y}$ is a $n \times 1$ vector, $\mathbf{X}$ is a $n \times K$ matrix and β is a $K \times 1$ vector.

1. Compute the ridge estimator $\hat{\beta}$.

2. We note $\hat{\beta}_{-i}$ the ridge estimator when leaving out the i^{th} observation:

$$\hat{\beta}_{-i} = \arg\min \frac{1}{2} \left(\mathbf{Y}_{-i} - \mathbf{X}_{-i}\beta\right)^\top \left(\mathbf{Y}_{-i} - \mathbf{X}_{-i}\beta\right) + \frac{\lambda}{2} \beta^\top \beta$$

where $\mathbf{Y}_{-i}$ and $\mathbf{X}_{-i}$ correspond to $\mathbf{Y}$ and $\mathbf{X}$ with the i^{th} row removed. By using the relationships $\mathbf{X}^\top \mathbf{X} = \mathbf{X}_{-i}^\top \mathbf{X}_{-i} + x_i x_i^\top$ and $\mathbf{X}^\top \mathbf{Y} = \mathbf{X}_{-i}^\top \mathbf{Y}_{-i} + x_i y_i$, show that:

$$\hat{\beta}_{-i} = \hat{\beta} - \frac{\left(\mathbf{X}^\top \mathbf{X} + \lambda I_K\right)^{-1} x_i \hat{u}_i}{1 - h_i}$$

where $\hat{u}_i = y_i - x_i^\top \hat{\beta}$ and $h_i = x_i^\top \left(\mathbf{X}^\top \mathbf{X} + \lambda I_K\right)^{-1} x_i$.

3. We note $\hat{y}_{i,-i} = x_i^\top \hat{\beta}_{-i}$ and $\hat{u}_{i,-i} = y_i - \hat{y}_{i,-i}$. Demonstrate that:

$$\hat{u}_{i,-i} = \frac{\hat{u}_i}{1 - h_i}$$

4. Calculate the predicted residual error sum of squares (PRESS) statistic:

$$\mathcal{P}\text{ress} = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{y}_{i,-i}\right)^2$$

where $\hat{y}_{i,-i}$ is the estimate of y_i based on the ridge model when leaving out the i^{th} observation.

5. In the OLS regression, we reiterate that $\mathrm{df}^{(\mathrm{model})} = \operatorname{trace}\mathbf{H} = K$ where $\mathbf{H}$ is the hat matrix for the OLS regression. Define the corresponding hat matrix $\mathbf{H}(\lambda)$ for the ridge regression. Show that:

$$\mathrm{df}^{(\mathrm{model})}(\lambda) = \sum_{k=1}^{K} \frac{s_k^2}{s_k^2 + \lambda}$$

where $(s_1, \ldots, s_K)$ are the singular values of $\mathbf{X}$.

6. The generalized cross-validation (GCV) statistic is defined by:

$$GCV = nK^2 \left(\sum_{k=1}^{K} \frac{(n-K)\, s_k^2 + n\lambda}{s_k^2 + \lambda} \right)^{-2} \mathrm{RSS}\left(\hat{\beta}(\lambda)\right)$$

where $\bar{h} = n^{-1} \sum_{i=1}^{n} \mathbf{H}(\lambda)_{i,i}$ and $\mathrm{RSS}\left(\hat{\beta}(\lambda)\right)$ is the residual sum of squares calculated with the ridge estimator $\hat{\beta}(\lambda)$. What is the relationship between GCV and PRESS statistics? What is the impact of λ?

7. Show that another expression of the GCV statistic is:

$$GCV = n \left(n - K + \sum_{k=1}^{K} \frac{\lambda}{s_k^2 + \lambda} \right)^{-2} \mathrm{RSS}\left(\hat{\beta}(\lambda)\right)$$

8. Using the data of Example 165 on page 940, calculate the estimates $\hat{\beta}_{-i}$ when λ is equal to 3.0. Compute also $\hat{y}_{i,-i}$, $\hat{u}_{i,-i}$, $\hat{u}_i$ and h_i. Deduce then the value of PRESS and GCV statistics.

15.4.3 K-means and the Lloyd's algorithm

1. We consider n observations with K attributes $x_{i,k}$ ($i = 1, \ldots, n$ and $k = 1, \ldots, K$). We note x_i the $K \times 1$ vector $(x_{i,1}, \ldots, x_{i,K})$. Show that:

$$\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left\| x_i - x_j \right\|^2 = n \sum_{i=1}^{n} \left\| x_i - \bar{x} \right\|^2$$

where:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

2. We recall that the loss function of the K-means clustering method is:

$$\mathcal{L}(\mathcal{C}) = \frac{1}{2} \sum_{j=1}^{n_C} \sum_{\mathcal{C}(i)=j} \sum_{\mathcal{C}(i')=j} \left\| x_i - x_{i'} \right\|^2$$

Deduce that:

$$\mathcal{L}(\mathcal{C}) = \sum_{j=1}^{n_C} n_j \sum_{\mathcal{C}(i)=j} \left\| x_i - \bar{x}_j \right\|^2$$

where $\bar{x}_j$ and n_j are two quantities to define.

3. We consider the following optimization function:

$$\left\{\mu_1^\star, \ldots, \mu_{n_C}^\star\right\} = \arg\min \sum_{j=1}^{n_C} n_j \sum_{\mathcal{C}(i)=j} \left\| x_i - \mu_j \right\|^2$$

   Show that $\mu_j^\star = \bar{x}_j$. Comment on this result.

4. Apply the K-means analysis to Example 169 and compare the results with those obtained with the discriminant analysis.

15.4.4 Derivation of the principal component analysis

The following exercise is taken from Chapters 1 and 2 of Jolliffe (2002). Let X be a $K \times 1$ random vector, whose covariance matrix is equal to Σ. We consider the linear transform $Z_j = \beta_j^\top X$ where β_j is a $K \times 1$ vector.

1. Calculate $\operatorname{var}(Z_1)$ and define the PCA objective function to estimate β_1. Show that β_1 is the eigenvector associated to the largest eigenvalue of Σ.

2. Calculate $\operatorname{var}(Z_2)$ and $\operatorname{cov}(Z_1, Z_2)$. Define then the PCA objective function to estimate β_2. Show that β_2 is the eigenvector associated to the second eigenvalue of Σ.

15.4.5 Two-class separation maximization

We note x_i the $K \times 1$ vector of exogenous variables X for the i^{th} observation.

1. We consider the case of J classes. We note $\hat{\mu}_j$ the mean vector for class $\mathcal{C}_j$:

$$\hat{\mu}_j = \frac{1}{n_j} \sum_{i \in \mathcal{C}_j} x_i$$

   and $\hat{\mu}$ the mean vector for the entire sample:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i = \frac{1}{n} \sum_{j=1}^{J} n_j \hat{\mu}_j$$

   Calculate the scatter matrices $\mathbf{S}$, $\mathbf{S}_W$ and $\mathbf{S}_B$. Show that:

$$\mathbf{S} = \mathbf{S}_W + \mathbf{S}_B$$

2. We now consider the two-class problem, and we note $y_i = \beta^\top x_i$. Show that:

$$\beta^\top \mathbf{S}_B \beta = \frac{n_1 n_2}{n_1 + n_2} \left(\tilde{\mu}_1 - \tilde{\mu}_2\right)^2$$

   where:

$$\tilde{\mu}_j = \frac{1}{n_j} \sum_{i \in \mathcal{C}_j} y_i$$

3. Show that:

$$\beta^\top \mathbf{S}_W \beta = \tilde{s}_1^2 + \tilde{s}_2^2$$

   where:

$$\tilde{s}_j^2 = \sum_{i \in \mathcal{C}_j} \left(y_i - \tilde{\mu}_j\right)^2$$

4. Deduce that the Fisher optimization program is:

$$\beta^\star = \arg\max \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

   What is the interpretation of this statistical problem?

5. Find the optimal value $\beta^\star$ and verify that the decision boundary is linear.

6. Using Example 170 on page 967, calculate $\mathbf{S}_W$ and $\mathbf{S}_B$. Find the optimal value $\beta^\star$ and compute the score for each observation. Propose an assignment decision based on the mid-point rule. Comment on these results.

15.4.6 Maximum likelihood estimation of the probit model

1. Given a sample $\{(x_i, y_i), i = 1, \ldots, n\}$, find the log-likelihood function of the probit model.

2. Let $J(\beta)$ be the Jacobian matrix of the log-likelihood vector. Show that:

$$J_{i,k}(\beta) = \frac{\left(y_i - \Phi\left(x_i^\top \beta\right)\right)\phi\left(x_i^\top \beta\right)}{\Phi\left(x_i^\top \beta\right)\left(1 - \Phi\left(x_i^\top \beta\right)\right)} \cdot x_{i,k}$$

   for $i = 1, \ldots, n$ and $k = 1, \ldots, K$. Define the score vector $\mathcal{S}(\beta)$.

3. Let $H(\beta)$ be the Hessian matrix of the log-likelihood function. Show that:

$$H(\beta) = -\sum_{i=1}^{n} H_i \cdot \left(x_i x_i^\top\right)$$

   where:

$$\begin{aligned} H_i &= y_i \frac{\left(\phi\left(x_i^\top \beta\right) + x_i^\top \beta \Phi\left(x_i^\top \beta\right)\right)}{\Phi\left(x_i^\top \beta\right)^2} \phi\left(x_i^\top \beta\right) + \\ &\quad (1 - y_i) \frac{\left(\phi\left(x_i^\top \beta\right) - x_i^\top \beta\left(1 - \Phi\left(x_i^\top \beta\right)\right)\right)}{\left(1 - \Phi\left(x_i^\top \beta\right)\right)^2} \phi\left(x_i^\top \beta\right) \end{aligned}$$

4. Propose a Newton-Raphson algorithm to find the ML estimate.

15.4.7 Computation of feed-forward neural networks

We consider the canonical neural network without constant and direct link.

1. We note X the input matrix of dimension $n \times n_x$ and Y the output matrix of dimension $n \times n_y$. Let $\hat{Y}$ be the prediction of Y. Find the matrix relationship between X and $\hat{Y}$ with respect to the parameter matrices β and γ of dimension $n_z \times n_x$ and $n_y \times n_z$.

2. We assume that the activation functions $f_{x,z}$ and $f_{z,y}$ are the identity function. Demonstrate that the neural network is equivalent to an overidentified linear model or a constrained linear regression.

3. We consider the additive loss function:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{n_y} \mathcal{L}_{i,j}(\theta)$$

   where:

$$\mathcal{L}_{i,j}(\theta) = \xi(y_j(x_i), y_{i,j})$$

   Calculate the matrices $\partial_\gamma \mathcal{L}(\theta)$ and $\partial_\beta \mathcal{L}(\theta)$ of dimension $n_y \times n_z$ and $n_z \times n_x$.

4. We assume that the activation functions $f_{x,z}$ and $f_{z,y}$ correspond to the logistic function and the loss is the least squares error function. Find the matrices $\partial_\gamma \mathcal{L}(\theta)$ and $\partial_\beta \mathcal{L}(\theta)$.

5. Same question if we consider the cross-entropy error loss:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{n} (y_i \ln y(x_i) + (1 - y_i) \ln(1 - y(x_i)))$$

6. Explain why we cannot use the property of additivity in the case of the softmax function.

7. Calculate the matrices $\partial_\gamma \mathcal{L}(\theta)$ and $\partial_\beta \mathcal{L}(\theta)$ when $f_{z,y}$ is the softmax function, $f_{x,z}$ is the identity function, and the loss function is the multi-class error function:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{n} \sum_{j=1}^{n_C} y_{i,j} \ln y_j(x_i)$$

   where n_C is the number of classes[79].

8. Extend the previous results when we consider a constant between the x's and the z's, a constant between the z's and the y's and a direct link between the x's and the y's.

15.4.8 Primal and dual problems of support vector machines

The goal of this exercise is to determine the primal and dual problems of the different SVM models. For each problem, we ask to write the primal problem into a quadratic programming (QP) format:

$$\begin{aligned} \hat{\theta} &= \arg\min \frac{1}{2}\theta^\top Q\theta - \theta^\top R \\ \text{s.t.} & \left\{ \begin{array}{l} A\theta = B \\ C\theta \geq D \\ \theta^- \leq \theta \leq \theta^+ \end{array} \right. \end{aligned}$$

where θ is the vector of parameters. Then, we ask to find the corresponding dual problem and also the associated QP matrix form.

[79] Hint: Use the following decomposition $\mathcal{L}(\theta) = \sum_{i=1}^{n} \mathcal{L}_i(\theta)$.

Hard margin classification

We first begin with the hard margin classifier. We recall that the primal optimization problem is:

$$\begin{aligned} \left\{\hat{\beta}_0, \hat{\beta}\right\} &= \arg\min \frac{1}{2} \|\beta\|_2^2 \\ \text{s.t.} \quad & y_i \left(\beta_0 + x_i^\top \beta\right) \geq 1 \qquad \text{for } i = 1, \ldots, n \end{aligned}$$

1. By noting θ the vector of parameters, write the primal problem in the QP form.
2. We note $\alpha = (\alpha_1, \ldots, \alpha_n)$ the vector of Lagrange coefficients associated to the constraints $y_i \left(\beta_0 + x_i^\top \beta\right) \geq 1$. Write the Lagrange function and find the first-order conditions.
3. Deduce that the dual problem is:

$$\begin{aligned} \hat{\alpha} &= \arg\max \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^\top x_j^\top \\ \text{s.t.} \quad & \alpha \geq \mathbf{0} \end{aligned}$$

4. Write this dual problem as a QP problem.
5. Determine the dual QP problem directly by applying Equation (A.12) on page 1047. What do you observe? How to fix this issue?

Soft margin classification with binary hinge loss

We now consider the soft margin classification problem:

$$\begin{aligned} \left\{\hat{\beta}_0, \hat{\beta}, \hat{\xi}\right\} &= \arg\min \frac{1}{2} \|\beta\|_2^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & \left\{ \begin{array}{l} y_i \left(\beta_0 + x_i^\top \beta\right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{array} \right. \qquad \text{for } i = 1, \ldots, n \end{aligned}$$

1. Write the primal problem as a QP problem.
2. Show that the objective function of the dual problem does not change compared to the hard margin classifier. What does the dual QP problem become?
3. How can we characterize the support vectors?
4. Find the optimal values of ξ_i.
5. We consider the training data set given in Table 15.18 on page 989. Represent the optimal values of β_0, β_1, β_2, $\sum_{i=1}^n \xi_i$ and the margin M with respect to C. Compare the optimal hyperplane when $C = 0.07$ with the optimal hyperplane obtained with the hard margin classifier.

Soft margin classification with squared hinge loss

We replace the binary hinge loss by the squared hinge loss:

$$\begin{aligned} \left\{\hat{\beta}_0, \hat{\beta}, \hat{\xi}\right\} &= \arg\min \frac{1}{2} \|\beta\|_2^2 + C \sum_{i=1}^n \xi_i^2 \\ \text{s.t.} \quad & \left\{ \begin{array}{l} y_i \left(\beta_0 + x_i^\top \beta\right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{array} \right. \qquad \text{for } i = 1, \ldots, n \end{aligned}$$

1. Write the primal problem as a QP problem.

2. Find the dual problem. What do you observe?

3. We consider the training data set given in Table 15.18 on page 989. Study the convergence of the optimal values of β_0, β_1, β_2, $\sum_{i=1}^{n} \xi_i$ and the margin M with respect to C. What is the main difference between binary and squared hinge loss functions?

4. We introduce in the training set two new points $(6.0, 5.0, +1)$ $(i = 16)$ and $(2.0, 2.0, -1)$ $(i = 17)$. Calculate $\hat{\beta}_0$, $\hat{\beta}$, $\hat{\alpha}_i$ and $\hat{\xi}_i$ when the constant C is equal to 1.

Soft margin classification with ramp loss

1. Compare $0-1$, binary hinge, squared hinge and ramp loss functions.

2. Using the property $\min(1, \max(0, a)) = \max(0, a) - \max(0, a-1)$, show that $\mathcal{L}^{\text{ramp}}(x_i, y_i)$ is the difference of two convex functions. Comment on this result.

LS-SVM regression

We consider the following optimization problem:

$$\begin{aligned} \left\{\hat{\beta}_0, \hat{\beta}, \hat{\xi}\right\} &= \arg\min \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^{n} \xi_i^2 \\ &\text{s.t.} \quad y_i = \beta_0 + x_i^\top \beta + \xi_i \qquad \text{for } i = 1, \ldots, n \end{aligned}$$

1. Write the primal problem as a QP problem.

2. Find the dual QP problem.

3. Deduce the expression of $\hat{\beta}_0$ and $\hat{\beta}$. Show that the residuals are centered.

ε-SVM regression

We consider the following optimization problem:

$$\begin{aligned} \left\{\hat{\beta}_0, \hat{\beta}, \hat{\xi}^-, \hat{\xi}^+\right\} &= \arg\min \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^{n} \left(\xi_i^- + \xi_i^+\right) \\ &\text{s.t.} \quad \left\{\begin{array}{l} \beta_0 + x_i^\top \beta - y_i \leq \varepsilon + \xi_i^- \\ y_i - \beta_0 - x_i^\top \beta \leq \varepsilon + \xi_i^+ \\ \xi_i^- \geq 0 \\ \xi_i^+ \geq 0 \end{array}\right. \qquad \text{for } i = 1, \ldots, n \end{aligned}$$

where $\varepsilon \geq 0$.

1. Write the primal problem as a QP problem.

2. Find the dual problem.

3. Write the dual problem as a QP problem.

4. Deduce the expression of $\hat{\beta}_0$ and $\hat{\beta}$.

5. Calculate the optimal values $\hat{\xi}^-$ and $\hat{\xi}^+$.

6. What does the optimization problem becomes when $\varepsilon = 0$?

15.4.9 Derivation of the AdaBoost algorithm as the solution of the additive logit model

We consider a special case of additive models, where the loss function is specified as follows:

$$\mathcal{L}\left(\beta_{(s)}, f_{(s)}\right) = \sum_{i=1}^{n} \mathcal{L}\left(y_i, \hat{g}_{(s-1)}\left(x_i\right) + \beta_{(s)} f_{(s)}\left(x_i\right)\right)$$

$\hat{g}_{(s)}\left(x\right) = \sum_{s'=1}^{s} \hat{\beta}_{(s')} \hat{f}_{(s')}\left(x\right)$, $\hat{f}_{(s)}$ is the s^{th} optimal classification model and $\mathcal{L}\left(y, f\left(x\right)\right) = e^{-y f(x)}$.

1. Show that:
$$\mathcal{L}\left(\beta_{(s)}, f_{(s)}\right) = \sum_{i=1}^{n} w_{i,s} e^{-y_i \beta_{(s)} f_{(s)}(x_i)}$$
where $w_{i,s}$ is a quantity to determine.

2. Find an expression of $\mathcal{L}\left(\beta_{(s)}, f_{(s)}\right)$ that depends on the error rate:
$$\mathcal{L}_{(s)} = \frac{\sum_{i=1}^{n} w_{i,s} \cdot \mathbb{1}\left\{y_i \neq y_{i,s}\right\}}{\sum_{i=1}^{n} w_{i,s}}$$
where $y_{i,s} = f_{(s)}\left(x_i\right)$.

3. We assume that $f_{(s)}$ is known. Verify that the optimal value of $\hat{\beta}_{(s)}$:
$$\hat{\beta}_{(s)} = \arg\min \mathcal{L}\left(\beta_{(s)}, f_{(s)}\right)$$
is equal to:
$$\hat{\beta}_{(s)} = \frac{1}{2} \ln\left(\frac{1 - \mathcal{L}_{(s)}}{\mathcal{L}_{(s)}}\right)$$

4. Suppose that $\hat{f}_{(s)}$ has been already estimated. Show that the normalized observation weights are:
$$w_{i,s+1} = \frac{w_{i,s} e^{w_s \cdot \mathbb{1}\{y_i \neq \hat{y}_{i,s}\}}}{\sum_{i=1}^{n} w_{i',s} e^{w_s \cdot \mathbb{1}\left\{y_{i'} \neq \hat{y}_{i',s}\right\}}}$$
where w_s is a parameter to determine.

5. Conclude on these results.

15.4.10 Weighted estimation

We note $w = \left(w_1, \ldots, w_n\right)$ the vector of observation weights.

1. We consider the weighted log-likelihood function:
$$\boldsymbol{\ell}_w\left(\theta\right) = \sum_{i=1}^{n} w_i \cdot \boldsymbol{\ell}_i\left(\theta\right)$$
   (a) Define the weighted maximum likelihood estimator.
   (b) Find the expression of the Jacobian and Hessian matrices.

2. We consider neural networks (Exercise 15.4.7 on page 1025).

   (a) Define the least squares loss function $\mathcal{L}_w(\theta)$. Give the matrix form of the derivatives $\partial_\gamma \mathcal{L}_w(\theta)$ and $\partial_\beta \mathcal{L}_w(\theta)$.
   (b) Same question if we consider the cross-entropy loss function.

3. We consider the soft margin SVM classification (Exercise 15.4.8 on page 1027).
   (a) Define the optimization problem.
   (b) What is the impact of introducing weights on the primal and dual problems.
   (c) Why weighted hard margin classification does not make sense?