Multivariate Statistical Analysis

Selected Lecture Notes

Radoslav Harman

Faculty of Mathematics, Physics and Informatics, Comenius University

May 9, 2023

Contents

1 Principal Components Analysis

1.1 Mathematical background

We assume that the reader is already familiar with fundamental notions and results of matrix algebra and multivariate probability, but we will give a brief review of some of the facts that are particularly important for multivariate statistics.

Recall that a $p \times p$ matrix Σ is **non-negative definite**[1], if it is symmetric and satisfies $a^T \Sigma a \geq 0$ for any vector $a \in \mathbb{R}^p$. If $\Sigma u = \lambda u$ for some $\lambda \in \mathbb{R}$ and $u \in \mathbb{R}^p$, $u \neq 0$, then u is an **eigenvector** of Σ and λ is the **eigenvalue** of Σ corresponding to u.

Vectors $u_1, \ldots, u_p$ form an **orthonormal system** if $u_1, \ldots, u_p$ are mutually orthogonal and they are normalized such that $\|u_i\| = 1$ for all $i = 1, \ldots, p$. Matrix U of the type $p \times p$ is an **orthogonal matrix** if $UU^T = I_p$, where I_p denotes the $p \times p$ identity matrix. That is, U is orthogonal if and only if the columns of U form an orthonormal system of vectors. The linear mapping corresponding to an orthogonal matrix is a rotation or a composition of reflection and rotation.

Theorem 1.1 (Spectral decomposition of a non-negative definite matrix)**.** *For any non-negative definite $p \times p$ matrix Σ there exists an orthonormal system $u_1, \ldots, u_p$ of eigenvectors such that*

$$\Sigma = \sum_{i=1}^{p} \lambda_i u_i u_i^T = U \Lambda U^T, \tag{1}$$

where λ_i is the eigenvalue of Σ corresponding to the eigenvector u_i for all $i = 1, \ldots, p$, $U = (u_1, \ldots, u_p)$ is the orthogonal matrix of normalized eigenvectors

[1] A non-negative definite matrix is sometimes called "positive semidefinite".

and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$ *is the diagonal matrix with the eigenvalues on the diagonal. If* $\lambda_1 > \lambda_2 > \cdots > \lambda_p$*, then the eigenvectors* $u_1, \ldots, u_p$ *are uniquely defined (up to a possible change of the sign).*

A $p \times p$ matrix Σ is **positive definite**, if it is symmetric and satisfies $a^T \Sigma a > 0$ for any vector $0 \neq a \in \mathbb{R}^p$. A matrix Σ is positive definite if and only if Σ is a non-negative definite non-singular matrix which is if and only if Σ is a non-negative definite matrix with all eigenvalues strictly positive.

An **orthogonal projector** on a k-dimensional linear space $\mathcal{A} \subseteq \mathbb{R}^p$ is the unique symmetric matrix P of the type $p \times p$ such that $Py \in \mathcal{A}$ for all $y \in \mathbb{R}^p$, $Px = x$ for all $x \in \mathcal{A}$, and $x - Px$ is orthogonal to Px for all $x \in \mathbb{R}^p$, which we denote $(x - Px) \perp Px$. If A is a $p \times k$ matrix with rank k, where $k \leq p$, then $A^T A$ is a non-singular matrix and $P = A(A^T A)^{-1} A^T$ is the orthogonal projector on the linear space $\mathcal{C}(\mathcal{A})$ generated by the columns of A.

For a p-dimensional random vector $\mathbf{X} = (X_1, \ldots, X_p)^p$, the **variance-covariance matrix** is a $p \times p$ matrix Σ with elements $\Sigma_{ij} = cov(X_i, X_j)$, $i, j = 1, \ldots, p$. The variance-covariance matrix is always non-negative definite and, typically[2], it is also non-singular, i.e., positive definite. Geometrically, Σ determines the "shape" of the multivariate data generated as independent samples of $\mathbf{X}$.

More generally, by Cov we will denote the matrix of all mutual covariances of the components of a pair of random vectors $\mathbf{X}$ and $\mathbf{Z}$. For multivariate statistical analysis, it is very important to know how the variance-covariance matrix and the matrix Cov changes under linear transformations of the random vector(s).

Theorem 1.2 (The effect of a linear transformation on the variance-covariance matrix)**.** *If* $\mathbf{X}$ *is a* p*-dimensional random vector with covariance matrix* Σ *and* A *is an* $m \times p$ *matrix, then the* m*-dimensional random vector* $\mathbf{Y} = A\mathbf{X}$ *has variance-covariance matrix* $A\Sigma A^T$*. More generally: If* $\mathbf{X}$ *is a* p*-dimensional random vector,* $\mathbf{Z}$ *is an* r*-dimensional random vector,* A *is an* $m \times p$ *matrix and* B *is an* $k \times r$ *matrix, then* $Cov(A\mathbf{X}, B\mathbf{Z}) = ACov(\mathbf{X}, \mathbf{Z})B^T$*.*

Principal components are based on a rotation (i.e., a specific linear transformation) of an underlying random vector as detailed in the next section.

[2]For instance if the random vector $\mathbf{X}$ has a distribution continuous with respect to the Lebesgue measure in $\mathbb{R}^p$.

1.2 Theoretical principal components

Let μ be the mean value vector and let Σ be the variance-covariance matrix of a random vector $\mathbf{X} = (X_1, \ldots, X_p)^T$ which corresponds to p-dimensional measurements or observations on n objects[3].

Let $u_1, \ldots, u_p$ be an orthonormal system of eigenvectors of Σ, let $\lambda_1 \geq \ldots \geq \lambda_p$ be the corresponding eigenvalues and let $U = (u_1, \ldots, u_p)$; cf. Theorem 1.1. Vectors $u_1, \ldots, u_p$ determine what we will call "principal variance directions" and $\lambda_1, \ldots, \lambda_p$ determine the variances in the principal directions.

Principal variance directions can be illustrated on an example of a random vector with a two-dimensional normal distribution. For instance, if

$$\Sigma = \begin{pmatrix} 3.25 & 1.30 \\ 1.30 & 1.75 \end{pmatrix},$$

then $u_1 \approx (0.87; 0.50)^T$, $u_2 \approx (-0.50; 0.87)^T$, $\lambda_1 \approx 4$ and $\lambda_2 \approx 1$. In Figure 1 we see points randomly generated from $N_2((0,0)^T, \Sigma)$. Arrows denote the directions of vectors u_1 a u_2. The length of the arrows is proportional to $\sqrt{\lambda_1}$ and $\sqrt{\lambda_2}$.

Clearly, eigenvectors and eigenvalues of the variance-covariance matrix Σ capture important aspects of the "shape" of the distribution of $\mathbf{X}$. The essence of principal components analysis is the rotation of $\mathbf{X}$ (or a random sample), to the coordinate system determined by the eigenvectors of Σ (or by the eigenvectors of the sample variance-covariance matrix S_n, see Subsection 1.3).

An immediate consequence of Theorems 1.1 and 1.2 is:

Theorem 1.3 (De-correlation of a random vector)**.** *Random vector* $\mathbf{Y} = U^T(\mathbf{X} - \mu)$ *has the zero mean value and its variance-covariance matrix is* $Var(\mathbf{Y}) = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$*. That is, the components of* $\mathbf{Y}$ *are uncorrelated, their variances are* $\lambda_1 \geq \cdots \geq \lambda_p$ *and their standard deviations are* $\sqrt{\lambda_1} \geq \cdots \geq \sqrt{\lambda_p}$*.*

The previous theorem gives a theoretical basis for the following definition.

[3]We will only consider random variables with finite mean values and variances, that is, we will assume that all random vectors have well-defined, finite variance-covariance matrices.

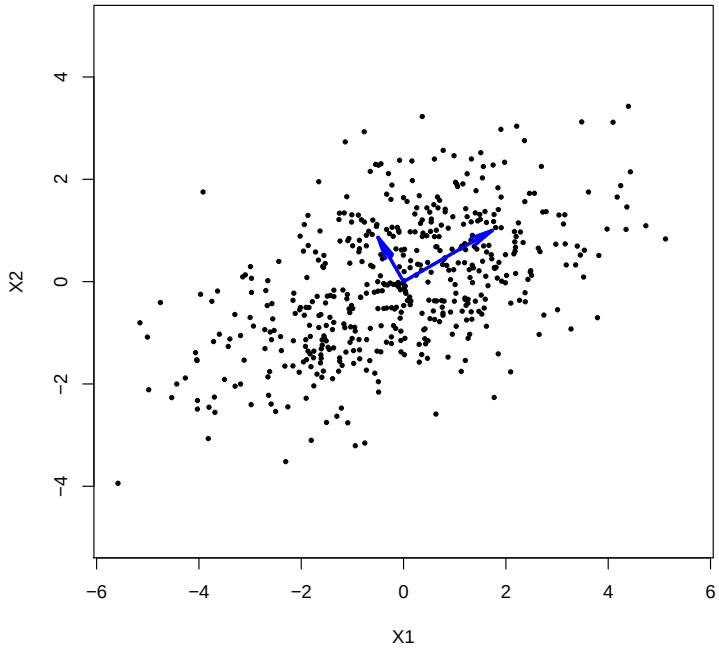

Figure 1: Principal variance directions defined by a pair of orthogonal eigenvectors of a two-dimensional normal distribution. In the figure, the lengths of the eigenvectors is proportional to the standard deviation of the corresponding principal components.

Definition 1.1 (Principal components of a random vector)**.** *Random vector* $\mathbf{Y} = (Y_1, \ldots, Y_p)^T$ *from Theorem 1.3 is called the* ***vector of*** *(theoretical)* ***principal components*** *of the random vector* $\mathbf{X}$*. For* $i = 1, \ldots, p$*, the random variable* $Y_i = u_i^T(\mathbf{X} - \mu)$ *is called the* i***-th principal component*** *of the random vector* $\mathbf{X}$*.*[4]

It is simple to show that for all $i \in \{1, \ldots, p\}$ the random vector $Y_i u_i$ is the orthogonal projection of $\mathbf{X}$ onto the 1-dimensional subspace (i.e., a line) defined by all real multiples of u_i. In other words, principal components $Y_1, \ldots, Y_p$ form (random) coordinates of $\boldsymbol{X}$ in the coordinate system defined by the orthonormal vectors $u_1, \ldots, u_p$.

Theorem 1.3 states that principal components $Y_1, \ldots, Y_p$ of a random

[4]Note that if some of the eigenvalues of Σ are equal, then there is an infinite number of possible choices of the corresponding eigenvectors, i.e., feasible definitions of the vectors of principal components.

vector $\mathbf{X}$ are uncorrelated. Thus, the transformation to principal components is occasionally referred to as "de-correlation" of $\mathbf{X}$. Importantly, variances of principal components are in decreasing order. Note also that the sum of variances of principal components is the same as the sum of sample variances of variables $X_1, \ldots, X_p$ which is probably the basis for the expression that (all) principal components "explain" (all) variation in the data.

Mutual relations of the coordinates of the original random vector $\mathbf{X}$ and the principal components of $\mathbf{X}$ are given by the following theorem.

Theorem 1.4 (Relations of original variables and principal components)**.** *Let $\mathbf{Y} = (Y_1, \ldots, Y_p)^T$ be the vector of principal components of the random vector $\mathbf{X}$ from Theorem 1.3. Then, for any pair $i, j \in \{1, \ldots, p\}$, we have*

$$cov(X_i, Y_j) = u_{ij}\lambda_j,\ \rho(X_i, Y_j) = u_{ij}\sqrt{\lambda_j}/\sigma_i, \tag{2}$$

where $u_{ij} = (u_j)_i$ is the i,j-th element of the matrix U (that is, the i-th coordinate of the eigenvector u_j), and $\sigma_i = \sqrt{DX_i}$.

Proof. From $X_i = e_i^T\mathbf{X}$, where e_i is the i-the standard unit vector and from $Y_j = u_j^T(\mathbf{X} - \mu)$ we obtain

$$\begin{aligned} cov(X_i, Y_j) &= cov(e_i^T\mathbf{X}, u_j^T(\mathbf{X} - \mu)) = e_i^T Var(\mathbf{X})u_j = \\ &= e_i^T\left(\sum_{k=1}^{p}\lambda_k u_k u_k^T\right)u_j =^* e_i^T\lambda_j u_j = \lambda_j e_i^T u_j = \lambda_j u_{ij}. \end{aligned}$$

The equality denoted by the asterisk follows from the fact that $u_1, \ldots, u_p$ are mutually orthogonal and have the unit length. □

Exercise 1.1. *Prove the following claim. Let $b_1, b_2 \in \mathbb{R}^p$, $b_1, b_2 \neq 0$ and let $\mathbf{X}$ be a p-dimensional random vector with positive definite covariance matrix. Then the random variables $b_1^T\mathbf{X}$ and $b_2^T\mathbf{X}$ are uncorrelated if and only if $b_1 \perp b_2$.*

The following theorem provides an important optimization/probabilistic interpretation of principal components.

Theorem 1.5 (Maximum variance justification of principal components)**.** *The first principal component Y_1 of the p-dimensional random vector $\mathbf{X}$ has the largest variance from all normed linear combinations of the components of $\mathbf{X}$. Formally:*

$$\mathrm{Var}(Y_1) = \max\{\mathrm{Var}(b^T\mathbf{X}) : b \in \mathbb{R}^p, \|b\| = 1\}$$

For $k \geq 2$, the k-th principal component Y_k of the random vector $\mathbf{X}$ has the largest variance from all normed linear combinations of the components of $\mathbf{X}$ that are uncorrelated with $Y_1, \ldots, Y_{k-1}$. Formally[5]

$$\mathrm{Var}(Y_k) = \max\{\mathrm{Var}(b^T\mathbf{X}) : b \in \mathbb{R}^p, \|b\| = 1, b \perp u_1, \ldots, b \perp u_{k-1}\}.$$

Proof. Let $u_1, \ldots, u_p$ be an orthonormal system of eigenvectors of the covariance matrix Σ of the p-dimensional random vector $\mathbf{X}$, and let $\lambda_1 > \ldots > \lambda_p$ be the corresponding eigenvalues. Let $b \in \mathbb{R}^p$, $\|b\| = 1$, and let $b = \sum_{i=1}^p c_i u_i$ be the expression of b in the orthonormal v basis $u_1, \ldots, u_p$ of the space $\mathbb{R}^p$. Since the vectors $u_1, \ldots, u_p$ are orthonormal, we obtain $u_j^T b = \sum_i c_i u_j^T u_i = c_j$ for any $j \in \{1, \ldots, p\}$, and $\sum_i c_i^2 = \sum_i c_i^2 u_i^T u_i = \sum_i (c_i u_i)^T \sum_k (c_k u_k) = b^T b = 1$. Therefore

$$\mathrm{Var}(b^T\mathbf{X}) = b^T \left(\sum_{i=1}^p \lambda_i u_i u_i^T \right) b = \sum_{i=1}^p \lambda_i c_i^2 \leq^* \lambda_1 = \mathrm{Var}(Y_1). \tag{3}$$

The inequality denoted by the asterisk follows from the fact that $\sum_{i=1}^p c_i^2 = 1$, i.e., $\sum_{i=1}^p \lambda_i c_i^2$ is a weighted average of eigenvalues, which of course cannot be larger than the largest eigenvalue λ_1. Since $Y_1 = u_1\mathbf{X}$, that is, Y_1 is itself a normed linear combination of the coordinates of $\mathbf{X}$, we obtain the first part of the theorem.

If we have $2 \leq k \leq p$ and an additional condition $b \perp u_1, \ldots, b \perp u_{k-1}$, then $c_i = 0$ for all $i = 1, \ldots, k-1$, which implies

$$\mathrm{Var}(b^T\mathbf{X}) = \sum_{i=1}^p \lambda_i c_i^2 = \sum_{i=k}^p \lambda_i c_i^2 \leq^* \lambda_k = \mathrm{Var}(Y_k),$$

in a way analogous to (3). □

The transformation of a random vector to principal components has several alternative geometric/optimization interpretations. For instance, let $\mathbf{X} \sim N_p(0, \Sigma)$ and by $P_{\mathcal{A}}$ denote the orthogonal projector on a linear space $\mathcal{A}$. Let $\mathcal{A}^* \subseteq \mathbb{R}^p$ be the k-dimensional hyperplane that optimally fits the distribution of $\mathbf{X}$ in the sense of least squares, i.e., $\mathcal{A}^*$ minimizes $E(\|X - P_{\mathcal{A}^*}X\|^2)$. Then, it is possible to show that $\mathcal{A}^*$ is spanned by the eigenvectors $u_1, \ldots, u_k$ of Σ corresponding to the k largest eigenvalues.

A measure of **proportion of variance** "explained" by the first k principal components $Y_1, \ldots, Y_k$, $k \leq p$, or a "goodness of fit" measure of the

[5] See also Exercise 1.1.

k-dimensional hyperplane $\mathcal{A}^*$ from the previous paragraph, is the quantity

$$\alpha_k = \frac{\lambda_1 + \cdots + \lambda_k}{\lambda_1 + \cdots + \lambda_p}.$$

The quantity α_k is important for the selection of the number of principal components that capture significant amount of variability of the original p-dimensional data. It turns out that the variance-covariance matrix of multidimensional data is often such that α_k is close to 1 for values of k that are small relative to p. Geometrically, this means that the ellipsoid of dispersion is "thick" in a few orthogonal directions and "thin" in all others.

1.3 Sample principal components

In real applications, the mean value μ and the variance-covariance matrix Σ are rarely known, and the "theoretical" principal components from Definition 1.1 cannot be calculated. Usually, we only have a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_n$, $n \geq p$, from an otherwise unknown distribution, and the parameters μ, Σ need to be estimated by the **vector of mean values** $\bar{\mathbf{X}}_n = \frac{1}{n}\sum_{i=1}^n \mathbf{X}_i$ and the **sample variance-covariance matrix**

$$S_n = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})^T.$$

Definition 1.2 (Sample eigenvalues and eigenvectors)**.** *Ordered eigenvalues of S_n will be called **sample eigenvalues** and denoted by $\hat{\lambda}_1^{(n)} > \ldots > \hat{\lambda}_p^{(n)}$. The corresponding normalized eigenvectors of S_n will be called **sample eigenvectors** and denoted by $\hat{u}_1^{(n)}, \ldots, \hat{u}_p^{(n)}$.*[6]

Note that (from the point of view before collecting the data) the sample eigenvalues $\hat{\lambda}_i^{(n)}$ are random variables and the sample eigenvectors $\hat{u}_i^{(n)}$ are random vectors. In general, $\hat{\lambda}_i^{(n)} \to \lambda_i$ and $\hat{u}_i^{(n)} \to u_i$ as $n \to \infty$, but the

[6] In theory, some of the eigenvalues of S_n could be equal, but for the case of an independent sample of size $n \geq p$ from a continuous p-dimensional distribution, which is typical for applications, the probability of this event is zero. Moreover, even in the case of distinct eigenvalues $\hat{\lambda}_i^{(n)}$, the eigenvectors are not uniquely defined in the sense that if $\hat{u}_i^{(n)}$ is a normalized eigenvector, so is $-\hat{u}_i^{(n)}$. We assume that we have some consistent way of selecting which of these two eigenvectors is *the* normalized eigenvector. It would be counter-productive to be too mathematically rigorous at this point.

analysis of the stochastic convergence is usually very non-trivial. For the case of a normal sample, the convergence is given by the following result.[7]

Theorem 1.6. *For any $i \in \{1, \dots, p\}$, we have the following convergence in distribution:*

$$\begin{aligned}\sqrt{n-1}\,(\hat{\lambda}_i^{(n)} - \lambda_i) &\to N(0, 2\lambda_i^2),\\ \sqrt{n-1}\,(\hat{u}_i^{(n)} - u_i) &\to N_p\left(0, \sum_{i\neq j} \lambda_i\lambda_j(\lambda_i-\lambda_j)^{-2}u_j u_j^T\right).\end{aligned}$$

Thus, if the sample size is large enough ($n \gg p$), the values of $\hat{\lambda}_i^{(n)}$ and $\hat{u}_i^{(n)}$ can usually be taken as reliable approximations of the eigenvalues and eigenvectors of Σ, and the "speed of convergence" of the estimators is approximately $\sqrt{n}$. Note, however, that the variances and covariances of the limiting distributions depend on the estimated parameters. Moreover, the variance covariance matrix of $\hat{u}_i^{(n)}$'s tends to be large, if some of the eigenvalues are very similar.

For actual realizations $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ of the random vectors $\mathbf{X}_1, \dots, \mathbf{X}_n$[8], it is often useful to compute the vectors $\mathbf{y}_i = (\hat{u}_1^{(n)}, \dots, \hat{u}_p^{(n)})^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$, $i = 1, \dots, n$, which are called **principal components scores**. These vectors are estimates of the coordinates determined by the principal component directions of $\boldsymbol{x}_i$, $i = 1, \dots, n$.

In other words, assume that $\mathcal{A}^*$ is the k-dimensional hyperplane that best fits the data in the sense of least squares in the p-dimensional space[9] and let $\mathbf{z}_1, \dots, \mathbf{z}_n$ be the orthogonal projections of the feature vectors $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ onto $\mathcal{A}^*$. Then, the first k coordinates of the scores $\mathbf{y}_1, \dots, \mathbf{y}_n$ correspond to the coordinates of $\mathbf{z}_1, \dots, \mathbf{z}_n$ in the hyperplane $\mathcal{A}^*$. They can be used to represent the n objects in the k-dimensional space, usually in plane ($k = 2$).

The proportion α_k, $k \in \{1, \dots, p\}$, of the variance explained by the first k principal components can be estimated by

$$\hat{\alpha}_k^{(n)} = \frac{\hat{\lambda}_1^{(n)} + \dots + \hat{\lambda}_k^{(n)}}{\hat{\lambda}_1^{(n)} + \dots + \hat{\lambda}_p^{(n)}}.$$

[7] Note, however, that in applications the normality is usually not required; the aim of the principal component analysis is almost always a reduction of dimensionality, with the aim to compress the data or to discover new knowledge, see Subsection 1.4, not hypothesis testing.

[8] The realizations $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ are sometimes called feature vectors of objects.

[9] Hyperplane $\mathcal{A}^*$ is sometimes called "perpendicular regression" hyperplane.

For the random variables $\hat{\alpha}_k^{(n)}$, a theorem similar to 1.6 can be proved for the case of normality, stating that the random variables $\hat{\alpha}_k^{(n)}$ converge to the values α_k with a speed proportional to $\sqrt{n}$.

The values $\hat{\lambda}_i^{(n)}$, and $\hat{\alpha}_k^{(n)}$ form the basis of several rules of thumb for choosing an appropriate number k, i.e., for selecting how many principal components to retain to capture most variability in the data. Often, k is simply chosen to be the smallest number with the property $\hat{\alpha}_k^{(n)} > c$, where c is some constant, for instance 0.8. The **Kaiser's rule** suggests to take the smallest k, such that all values $\hat{\lambda}_k^{(n)}$ are larger than the average of all values $\hat{\lambda}_1^{(n)}, \ldots, \hat{\lambda}_p^{(n)}$[10]. Another popular method is to draw the so-called **scree plot** which is a piece-wise linear function connecting the points $(0, 0)$, $(1, \hat{\alpha}_1^{(n)})$,...,$(p, \hat{\alpha}_p^{(n)})$. If this line forms an **"elbow"** at a point k, it suggests that k could be an appropriate number of principal components to summarize the original data.

1.4 Applications of principal components

The simplest principal component analysis, as described in this chapter, is intended to be applied on a random sample of p-dimensional vectors without specific structure and without division of variables into the sets of dependent and independent variables. Although some theoretical results about principal components assume normality, it is routinely applied also to non-normal data.[11]

The results of principal component analysis are sometimes used as an end to itself, for instance as a means for visualization of multidimensional data using the first two[12] components of the vectors of scores or for getting an insight into the data based on a possible interpretation of coefficient vectors $\hat{u}_1^{(n)}, \ldots, \hat{u}_k^{(n)}$. Note, however, that there are also many other methods for the visualisation of multidimensional data, such as various projection pursuit methods, or the so-called multidimensional scaling, which we will describe in the next section.

[10] Note that the average of eigenvalues $\hat{\lambda}_i^{(n)}$, $i = 1, \ldots, p$, is equal to the average of variances of variables $X_1, \ldots, X_p$, which is in turn equal to $\mathrm{tr}(S_n)/p$.

[11] Indeed, when the sample principal components are used for visual display of data, i.e., for exploration of the structure of data, assumption of normality makes little sense. Moreover, real-world, highly multidimensional data rarely follow a multivariate normal distribution.

[12] It can be expected that in near future 3-dimensional data visualisation based on principal components will also become common.

Sometimes the results of principal component analysis (usually the vectors of first k coordinates of scores) form an input to a subsequent statistical procedure that benefits from a small-dimensional representation of the originally large-dimensional dataset; an example is reducing the number of explanatory variables for a regression analysis. Principal components analysis is used across all fields that handle multivariate data. A particularly famous application uses principal components for face recognition, see, e.g.,

`http://en.wikipedia.org/wiki/Eigenface`

The most problematic aspect of principal components is its dependence on the scale of individual variables, i.e., principal components are not scale invariant. For instance, if we change the units with which we measure some distance variable (say, from meters to millimetres), the principal components can significantly change. This is particularly problematic if the variables have very different magnitudes or if we simultaneously use variables of completely different nature (such as distances, times and weights) where it is impossible to express all variables in the same units. This is the reason why principal components analysis is sometimes based on the correlation matrix, instead of variance-covariance matrix, that is, all variables are scaled to unit standard deviation before the application of principal components.

2 Multidimensional Scaling

Suppose that we study a set of n objects and the relations of the objects are described by an $n \times n$ matrix D of their mutual **dissimilarities**. Multidimensional scaling is a class of methods that assign vectors $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n \in \mathbb{R}^k$ to the objects, such that the mutual distances of pairs $\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j$ are "close" to dissimilarities D_{ij} of the objects i and j. The usual aim is to discover a hidden structure in the data by means of visualizing the "map" of dissimilarities in a two or three dimensional space, i.e., $k = 2$ or $k = 3$.

As we saw in the previous section, a reasonable k-dimensional representation of the data can be obtained from sample principal components, but for the direct application of principal components analysis, we need to know the p-dimensional vectors of features of all objects. The so-called metric multidimensional scaling[13] is closely related to the principal component analysis; here, however, only the matrix D of dissimilarities is required. Note that methods of multidimensional scaling usually do not make assumptions about the probability distribution that generated the data.

[13] There is also an interesting variant called non-metric multidimensional scaling.

2.1 Theory for Classical Multidimensional Scaling

First, we will describe a classical solution of the following problem: How do we construct an n-touple $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n$ of points in $\mathbb{R}^k$, $k \leq n$, if we only know Euclidean distances between these points? Clearly, the solution is not unique, because for any solution, an orthogonal rotation or a shift also provides a feasible solution. Therefore, we can search for a solution $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n$ with the center of mass in $\mathbf{0}_k$, that is, $\sum_{i=1}^n \tilde{\boldsymbol{x}}_i = \mathbf{0}_k$.

It is simple to prove the following lemma.

Lemma 2.1. *Let $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$ be an $n \times k$ matrix, let $\sum_{i=1}^n \boldsymbol{x}_i = \mathbf{0}_k$, and let $B = XX^T$. Then (i) the sum of all elements of any row (and any column) of B is zero. (ii) $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = B_{ii} + B_{jj} - 2B_{ij}$.*

Matrix $B = XX^T$ from the previous lemma is called the **Gram matrix** of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ (its ij-th element is the scalar product of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$). The following theorem[14] shows that the Gram matrix of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ can be computed from the mutual distances of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, element by element.

Theorem 2.1. *Let $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$ be a matrix of the type $n \times k$, let $\sum_{i=1}^n \boldsymbol{x}_i = \mathbf{0}_k$ and let $B = XX^T$. Denote $D_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ for $i, j \in \{1, \ldots, n\}$. Then*

$$B_{ij} = -\frac{1}{2}\left(D_{ij}^2 - \frac{1}{n}\sum_{r=1}^n D_{ir}^2 - \frac{1}{n}\sum_{l=1}^n D_{lj}^2 + \frac{1}{n^2}\sum_{l=1}^n\sum_{r=1}^n D_{lr}^2\right), \tag{4}$$

for all $i, j \in \{1, \ldots, n\}$.

Proof. Fix $i, j \in \{1, \ldots, n\}$. Using Lemma 2.1, we have

$$\sum_{r=1}^n D_{ir}^2 = \sum_{r=1}^n (B_{ii} + B_{rr} - 2B_{ir}) = nB_{ii} + \mathrm{tr}(B), \tag{5}$$

$$\sum_{l=1}^n D_{lj}^2 = \sum_{l=1}^n (B_{ll} + B_{jj} - 2B_{lj}) = nB_{jj} + \mathrm{tr}(B), \tag{6}$$

$$\sum_{r=1}^n\sum_{l=1}^n D_{lr}^2 = \sum_{r=1}^n (nB_{rr} + \mathrm{tr}(B)) = 2n\mathrm{tr}(B). \tag{7}$$

[14] sometimes called the Young-Householder theorem

From (7) we obtain $\mathrm{tr}(B) = (2n)^{-1}\sum_{r=1}^{n}\sum_{l=1}^{n} D_{lr}^2$, which can be substituted to (5) and (6), yielding

$$B_{ii} = \frac{1}{n}\sum_{r=1}^{n} D_{ir}^2 - \frac{1}{2n^2}\sum_{r=1}^{n}\sum_{l=1}^{n} D_{lr}^2, \tag{8}$$

$$B_{jj} = \frac{1}{n}\sum_{l=1}^{n} D_{lj}^2 - \frac{1}{2n^2}\sum_{r=1}^{n}\sum_{l=1}^{n} D_{lr}^2. \tag{9}$$

From Lemma 2.1 we see that $B_{ij} = -\frac{1}{2}\left(D_{ij}^2 - B_{ii} - B_{jj}\right)$, which together with (8) and (9) provides the equality from the statement of the theorem. □

Exercise 2.1. *Show that the matrix B from the previous theorem can be obtained using the following matrix computation. Let $P = I_n - \frac{1}{n}1_n 1_n^T$*[15] *and let A be the matrix with elements $A_{ij} = -\frac{1}{2}D_{ij}^2$, $i, j \in \{1, \ldots, n\}$. Then $B = PAP$.*

Therefore, distances of vectors directly provide the Gram matrix of the vectors. We will show that from the Gram matrix it is a simple step to obtain a solution $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n$ of the original problem. In other words, if $B = XX^T$ for some $n \times k$ matrix X, then we can easily find some $n \times p$ matrix $\tilde{X}$, such that $B = \tilde{X}\tilde{X}^T$.

Clearly, the Gram matrix $B = XX^T$ is non-negative definite, that is, $B = U\Lambda U^T$, where $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$ is an orthogonal matrix of eigenvectors of B and Λ is a diagonal matrix with eigenvalues $\lambda_1 \geq \ldots \geq \lambda_n \geq 0$ on the diagonal. Since X is of the type $n \times k$, where $k \leq n$, the rank of B is at most k, i.e., $\lambda_{k+1} = \ldots = \lambda_n = 0$. Now we can easily verify that the matrix $\tilde{X} = (\sqrt{\lambda_1}\boldsymbol{u}_1, \ldots, \sqrt{\lambda_k}\boldsymbol{u}_k)$ satisfies $B = \tilde{X}\tilde{X}^T$. That is, the k-dimensional columns $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n$ of $\tilde{X}^T$ are the required solutions.

It is interesting to note that as a method of "dimensionality reduction", multidimensional scaling is essentially equivalent to the principal components analysis. In other words, if we do have p-dimensional vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ of features of n objects (or measurements of p variables on n objects), we can decide to compute the matrix D of mutual Euclidean distances of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, and then use the classical multidimensional scaling with some $k \leq p$, as described above. Then, the resulting vectors $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n \in \mathbb{R}^k$ are just orthogonally rotated vectors of the first k-coordinates of the principal component scores.

[15] Note that P is a projector.

2.2 Application of Classical Multidimensional Scaling

For real objects, the matrix D of dissimilarities is usually not based on Euclidean distances of some unknown vectors. Nevertheless, if we tentatively assume that D is a matrix of Euclidean distances of some vectors in $\mathbb{R}^k$ (or in a k-dimensional subspace of $\mathbb{R}^p$) and imitate the theoretical construction of the vectors $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n$ from the previous subsection, we often obtain a reasonable fit with the dissimilarities given by D.

If we use the matrix D of dissimilarities, it is always possible to compute the symmetric matrix B with elements given in Theorem 2.1. If D is not a perfect matrix of Euclidean distances, then B is not necessarily non-negative definite. However, if k largest eigenvalues $\lambda_1, \ldots, \lambda_k$ of B are positive and large compared to the absolute values of the remaining eigenvalues[16], then the mutual Euclidean distances of the k-dimensional rows of $(\sqrt{\lambda_1}\boldsymbol{u}_1, \ldots, \sqrt{\lambda_k}\boldsymbol{u}_k)$ give a good fit with dissimilarities D_{ij}.

Classical multidimensional scaling, as a means to visualize data given by their matrix of dissimilarities, have been used for instance in psychology to create a "perceptual map" of stimuli, in marketing to crate a "product (diss)similarity map", in social networks to create "friendship/collaboration maps" and so on.

Note that there are several more advanced alternatives to the classical multidimensional scaling: the so-called general metric multidimensional scaling and non-metric multidimensional scaling. These methods are beyond the scope of this introductory lecture.

3 Canonical Correlations

3.1 Mathematical background for Canonical Correlations

The theory of canonical correlation can be easily explained using the notion of the square-root matrix. Let Σ be a non-negative definite $p \times p$ matrix and let $\Sigma = \sum_{i=1}^{p} \lambda_i u_i u_i^T$ be the decomposition of Σ from Theorem 1.1. Let $\Sigma^{1/2} := \sum_{i=1}^{p} \sqrt{\lambda_i} u_i u_i^T$. Clearly, $\Sigma^{1/2}$ is a non-negative definite matrix satisfying $\Sigma^{1/2}\Sigma^{1/2} = \Sigma$, i.e., it is natural to call it the **square-root matrix** of Σ[17]. If Σ is positive definite, its square-root matrix is also positive definite

[16] A common rule of thumb is that $\sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{n} |\lambda_i| > 0.8$.

[17] It is also possible to show that $\Sigma^{1/2}$ is unique even in the case when the orthonormal system $u_1, \ldots, u_p$ of eigenvectors of Σ is not uniquely defined.

and its inverse $\Sigma^{-1/2} := (\Sigma^{1/2})^{-1}$ satisfies $\Sigma^{-1/2} = \sum_{i=1}^{p} \lambda_i^{-1/2} u_i u_i^T$.

Let $u \in \mathbb{R}^q$ be a vector of norm 1 and let M be a non-negative definite matrix of the type $q \times q$. Then $u^T M u \leq \lambda_1(M)$, where $\lambda_1(M)$ is the largest eigenvalue of M, which is sometimes called the **Railegh-Ritz theorem**. Similarly, assume that M has $m \leq p$ distinct positive eigenvalues $\lambda_1(M) > \ldots > \lambda_p(M)$ (and $p-m$ zero eigenvalues). Let $2 \leq k \leq m$. If u is orthogonal on $k-1$ eigenvectors of M corresponding to the $k-1$ largest eigenvalues of M, then $u^T M u \leq \lambda_k(M)$.

Recall also that, if F is any matrix, then the linear space generated by the columns of F is the same as the linear space generated by the columns of the matrix FF^T. If A, B are $s \times r$ matrices then $\mathrm{tr}(A^T B) = \mathrm{tr}(BA^T) = \mathrm{tr}(AB^T) = \mathrm{tr}(B^T A)$. Note that the largest eigenvalue of a non-negative definite matrix of rank 1 is equal to the trace of the matrix.

3.2 Theoretical Canonical Correlations

Consider the random vector $\boldsymbol{X} = (\boldsymbol{X}_{(1)}^T, \boldsymbol{X}_{(2)}^T)^T$ where $\boldsymbol{X}_{(1)}$ is the subvector of dimension p and $\boldsymbol{X}_{(2)}$ is the subvector of dimension q. Correspondingly, let the covariance matrix of $\boldsymbol{X}$ be divided into sub-blocks as follows:

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}.$$

We will assume that Σ_{11} and Σ_{22} are positive definite, i.e., there exist matrices $\Sigma_{11}^{-1/2}$ and $\Sigma_{22}^{-1/2}$. Let

$$\begin{aligned} B &:= \Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1/2}, \\ N_1 &:= BB^T = \Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21} \Sigma_{11}^{-1/2}, \\ N_2 &:= B^T B = \Sigma_{22}^{-1/2} \Sigma_{21} \Sigma_{11}^{-1} \Sigma_{12} \Sigma_{22}^{-1/2}. \end{aligned}$$

Clearly, N_1, N_2 are both non-negative definite. Moreover, N_1, N_2 have the same rank, because $\mathrm{rank}(N_1) = \mathrm{rank}(BB^T) = \mathrm{rank}(B) = \mathrm{rank}(B^T) = \mathrm{rank}(B^T B) = \mathrm{rank}(N_2)$. We will denote the common rank of matrices N_1, N_2 by the symbol m.

Lemma 3.1. *Let $\alpha_1, \ldots, \alpha_m$ be an orthonormal system of eigenvectors of N_1, with corresponding eigenvalues $\lambda_1(N_1) > \cdots > \lambda_m(N_1) > 0$. Similarly, let $\beta_1, \ldots, \beta_m$ be the orthonormal system of eigenvectors of N_2, with corresponding eigenvalues $\lambda_1(N_2) > \cdots > \lambda_m(N_2) > 0$. Then, for every $i = 1, \ldots, m$, we have $\lambda_i(N_1) = \lambda_i(N_2) =: \lambda_i$ and*

$$\beta_i = \frac{B^T \alpha_i}{\sqrt{\lambda_i}}, \quad \textit{or} \quad \beta_i = -\frac{B^T \alpha_i}{\sqrt{\lambda_i}}.$$

Proof. For the proof that all non-zero eigenvalues of N_1 and N_2 are equal, it is enough to show that each eigenvalue of N_1 is an eigenvalue of N_2[18]. For any $i \in \{1, \ldots, m\}$, a normalized eigenvector α_i of $N_1 = BB^T$ and corresponding eigenvalue $\lambda_i(N_1)$ we can write:

$$\begin{aligned} BB^T\alpha_i &= \lambda_i(N_1)\alpha_i, \\ B^TBB^T\alpha_i &= \lambda_i(N_1)B^T\alpha_i, \\ N_2B^T\alpha_i &= \lambda_i(N_1)B^T\alpha_i \end{aligned} \tag{10}$$

In addition, observe that

$$\|B^T\alpha_i\|^2 = \alpha_i^TBB^T\alpha_i = \alpha_i^TN_1\alpha_i = \alpha_i^T\lambda_i(N_1)\alpha_i = \lambda_i(N_1) > 0, \tag{11}$$

which implies $B^T\alpha_i \neq 0$. That is (10) entails that $\lambda_i(N_1)$ is an eigenvalue of N_2 and $B^T\alpha_i$ is a corresponding eigenvector of N_2 (not necessarily normalised). Moreover, since all eigenvalues of N_2 are assumed to be distinct, the corresponding orthonormal system of eigenvectors of N_2 is uniquely determined, up to the possible reversal of the direction. Therefore

$$\beta_i = \pm\frac{\boldsymbol{B}^T\alpha_i}{\|\boldsymbol{B}^T\alpha_i\|} = \pm\frac{\boldsymbol{B}^T\alpha_i}{\sqrt{\lambda_i}},$$

where the second equality follows from (11).

□

Definition 3.1 (Canonical variables and canonical correlations)**.** *For $i = 1, \ldots, m$ let*

$$\begin{aligned} a_i = \Sigma_{11}^{-1/2}\alpha_i,\ b_i = \Sigma_{22}^{-1/2}\beta_i, \\ U_i = a_i^T\boldsymbol{X}_{(1)},\ V_i = b_i^T\boldsymbol{X}_{(2)}, \\ \rho_i = \sqrt{\lambda_i}, \end{aligned}$$

where α_i, β_i a λ_i are defined in Lemma 3.1, such that $\beta_i = \frac{\boldsymbol{B}^T\alpha_i}{\sqrt{\lambda_i}}$. Then random variables $U_1, \ldots, U_m$, $V_1, \ldots, V_m$ are called ***canonical variables*** *and numbers $\rho_1, \ldots, \rho_m, 0, \ldots, 0$*[19] *are called* ***canonical correlations****.*

It is possible to show that the vectors a_i of coefficients that define the first group of canonical variables are eigenvectors of $\Sigma_{11}^{-1}\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$, and canonical

[18] Clearly, then, the symmetry of the problem implies that each eigenvalue of N_2 will be an eigenvalue of N_1.

[19] The number of zeros here is $\min(p, q) - m$, that is all canonical correlations of order higher than $m = \text{rank}(N_1) = \text{rank}(N_2)$ are defined to be zero.

correlations ρ_i are square roots of corresponding eigenvalues. Analogously, vectors b_i of coefficients that define the second group of canonical variables are eigenvectors of $\Sigma_{22}^{-1}\Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}$ and, as above, canonical correlations ρ_i are square roots of corresponding eigenvalues. That is, we can obtain canonical variables and correlations without actually computing the square root matrices; however, the square root matrices are useful for the theoretical results, because they allow us to work exclusively with non-negative definite matrices.

Theorem 3.1 (Mutual correlations of canonical variables)**.** *For all* $i, j \in \{1, \ldots, m\}$ *we have*

$$\begin{aligned} cov(U_i, U_j) &= \rho(U_i, U_j) = \delta_{ij}, \\ cov(V_i, V_j) &= \rho(V_i, V_j) = \delta_{ij}, \\ cov(U_i, V_j) &= \rho(U_i, V_j) = \delta_{ij}\rho_i, \end{aligned}$$

where δ_{ij} *is the Kronecker delta*[20]*.*

Proof.

$$\begin{aligned} cov(U_i, U_j) &= cov(a_i^T \boldsymbol{X}_{(1)}, a_j^T \boldsymbol{X}_{(1)}) = a_i^T \Sigma_{11} a_j \\ &= \alpha_i^T \Sigma_{11}^{-1/2} \Sigma_{11} \Sigma_{11}^{-1/2} \alpha_j = \alpha_i^T \alpha_j = \delta_{ij} \end{aligned}$$

and, similarly, we obtain $cov(V_i, V_j) = \delta_{ij}$. In particular, this implies $Var(U_i) = Var(V_i) = 1$, that is, the covariances are equal to correlations. We can conclude the proof by observing that:

$$\begin{aligned} cov(U_i, V_j) &= cov(a_i^T \boldsymbol{X}_{(1)}, b_j^T \boldsymbol{X}_{(2)}) = a_i^T \Sigma_{12} b_j = \alpha_i^T \Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1/2} \beta_j \\ &= \alpha_i^T \boldsymbol{B} \beta_j =^* \sqrt{\lambda_i} \beta_i^T \beta_j = \delta_{ij}\rho_i, \end{aligned}$$

where for the equality denoted by the asterisk we used $\beta_i = \frac{\boldsymbol{B}^T \alpha_i}{\sqrt{\lambda_i}}$. □

In a manner similar to principal components, canonical variables and canonical correlations have a probabilistic justification:

Theorem 3.2 (Maximum correlation justification of canonical variables and correlations)**.** *Canonical variables* $U_1 = a_1^T \boldsymbol{X}_{(1)}$ *a* $V_1 = b_1^T \boldsymbol{X}_{(2)}$ *have the maximum possible correlation among all pairs* $a^T \boldsymbol{X}_{(1)}$*,* $b^T \boldsymbol{X}_{(2)}$ *of non-zero linear combinations of the components of vectors* $\boldsymbol{X}_{(1)}$ *and* $\boldsymbol{X}_{(2)}$*. Formally:*

$$\rho(U_1, V_1) \geq \rho(a^T \boldsymbol{X}_{(1)}, b^T \boldsymbol{X}_{(2)}) \text{ for all } 0 \neq a \in \mathbb{R}^p, 0 \neq b \in \mathbb{R}^q.$$

[20] $\delta_{ii} = 1$ and $\delta_{ij} = 0$ if $i \neq j$.

For $k \geq 2$, the canonical correlations $U_k = a_k^T \boldsymbol{X}_{(1)}$ and $V_k = b_k^T \boldsymbol{X}_{(2)}$ have the largest correlation coefficient among all pairs $a^T \boldsymbol{X}_{(1)}$, $b^T \boldsymbol{X}_{(2)}$ of non-zero linear combinations of the components of vectors $\boldsymbol{X}_{(1)}$ a $\boldsymbol{X}_{(2)}$ that are uncorrelated with $U_1, \ldots, U_{k-1}$ and $V_1, \ldots, V_{k-1}$, respectively. Formally:

$$\rho(U_k, V_k) \geq \rho(a^T \boldsymbol{X}_{(1)}, b^T \boldsymbol{X}_{(2)}) \text{ for all } 0 \neq a \in \mathbb{R}^p, 0 \neq b \in \mathbb{R}^q$$
$$\text{s.t. } a^T \Sigma_{11} a_i = b^T \Sigma_{22} b_i = 0 \text{ for all } i = 1, \ldots, k-1.$$

Proof. As $\rho(U_1, V_1) = \sqrt{\lambda_1}$ and

$$\rho(a^T \boldsymbol{X}_{(1)}, b^T \boldsymbol{X}_{(2)}) = \frac{a^T \Sigma_{12} b}{\sqrt{a^T \Sigma_{11} a} \sqrt{b^T \Sigma_{22} b}},$$

which is invariant with respect to multiples of vectors a, b, for the proof of the first part of the theorem it is enough to show that $\lambda_1 \geq (a^T \Sigma_{12} b)^2$ under the constraint $a^T \Sigma_{11} a = b^T \Sigma_{22} b = 1$. Denote $u := \Sigma_{11}^{1/2} a$ and $v := \Sigma_{22}^{1/2} b$ and note that $a^T \Sigma_{11} a = b^T \Sigma_{22} b = 1$ implies $\|u\| = \|v\| = 1$. We obtain:

$$\begin{aligned}
(a^T \Sigma_{12} b)^2 &= (u^T \Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1/2} v)^2 \\
&= u^T \Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1/2} v v^T \Sigma_{22}^{-1/2} \Sigma_{21} \Sigma_{11}^{-1/2} u \\
&\leq^* \lambda_1(\Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1/2} v v^T \Sigma_{22}^{-1/2} \Sigma_{21} \Sigma_{11}^{-1/2}) \\
&= \mathrm{tr}(\Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1/2} v v^T \Sigma_{22}^{-1/2} \Sigma_{21} \Sigma_{11}^{-1/2}) \\
&= v^T \Sigma_{22}^{-1/2} \Sigma_{21} \Sigma_{11}^{-1/2} \Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1/2} v \\
&= v^T N_2 v \leq^* \lambda_1(N_2) = \lambda_1.
\end{aligned}$$

The inequalities denoted by the asterisk follow from the Raileigh-Ritz inequality.

The second part of the theorem follows similarly as the first one, observing that the conditions $b^T \Sigma_{22} b_i = 0$ for all $i = 1, \ldots, k-1$ imply that v is orthogonal to the eigenvectors of N_2 that correspond to $k-1$ largest eigenvalues, hence $v^T N_2 v \leq \lambda_k(N_2) = \lambda_k$. □

Note that unlike principal components, canonical correlations $\rho_1, \ldots, \rho_m$ do not depend on the units of individual variables, i.e., they are scale invariant.

3.3 Sample Canonical Correlations

In practice, we do not know the theoretical variance-covariance matrices of the random vector that we observe and we need to based the estimate of the canonical correlations on a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_n$ from the underlying $(p+q)$-dimensional distribution. The following definition is motivated by the remark after Definition 3.1.

Definition 3.2 (Sample canonical correlations). *Let $m = \min(p, q)$. Sample canonical correlations $\hat{\rho}_1, \ldots, \hat{\rho}_m$ are defined to be the square roots of the m largest eigenvalues of $\boldsymbol{S}_{22}^{-1}\boldsymbol{S}_{21}\boldsymbol{S}_{11}^{-1}\boldsymbol{S}_{12}$ (or, equivalently, of $\boldsymbol{S}_{11}^{-1}\boldsymbol{S}_{12}\boldsymbol{S}_{22}^{-1}\boldsymbol{S}_{21}$), where $\boldsymbol{S}_{11}$,$\boldsymbol{S}_{12}$, $\boldsymbol{S}_{21}$, and $\boldsymbol{S}_{22}$ are the $p \times p$, $p \times q$, $q \times p$, and $q \times q$ sub-blocks of the sample variance-covariance matrix*

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{S}_{11} & \boldsymbol{S}_{12} \\ \boldsymbol{S}_{21} & \boldsymbol{S}_{22} \end{pmatrix}.$$

In the case of normal errors and a large sample size, we can test the hypothesis that all theoretical canonical correlations are zero, i.e., that the sub-vector of the first p variables is independent with the sub-vector of the last q variables:

Theorem 3.3. *If $\mathbf{X}_1, \ldots, \mathbf{X}_n$ follow $N_{p+q}(\mu, \Sigma)$ and the upper-right $p \times q$ submatrix Σ_{12} of Σ is zero*[21]*, then the statistics*

$$W = \det(\mathbf{I} - \boldsymbol{S}_{22}^{-1}\boldsymbol{S}_{21}\boldsymbol{S}_{11}^{-1}\boldsymbol{S}_{12})$$

has asymptotically the Wilks distribution $\Lambda(p, n-1-q, q)$ and the statistics

$$Z = -\left(n - (p+q+3)/2\right)\ln(W)$$

has asymptotically the distribution χ^2_{pq}.

The tests using the Z statistics from the previous theorem is called the Bartlett ξ^2 test.

3.4 Applications of Canonical Correlations

Canonical correlations are used in the situations where variables can be logically divided into two distinct groups. For instance, in a psychological survey we can have a set of variables measuring one "personality dimension", and a set of variables measuring a second "personality dimension". The first canonical correlation ρ_1 then measures the overall degree of correlation of the first group of variables with the second group of variables.

Note that for $q = 1$ the first canonical correlation corresponds to the so-called coefficient of multiple correlation, which, in the context of linear regression, is closely related to the square of the coefficient of determination. (It is a useful theoretical exercise to simplify the theory of canonical correlations for $q = 1$.)

[21] This is of course if and only if the lower-left $q \times p$ submatrix Σ_{21} of Σ is zero.

4 Factor Analysis

In factor analysis, we assume that the random vector $\boldsymbol{X} = (X_1, \ldots, X_p)^T$ of observed (or "manifest") variables can be described in terms of an m-dimensional random vector $\boldsymbol{F} = (F_1, \ldots, F_m)^T$ of hidden (or "latent") variables, where m is substantially smaller than p. More precisely, we assume that the following statistical model holds:

$$X_i = \sum_{j=1}^{m} a_{ij} F_j + U_i, \quad i = 1, \ldots, p, \tag{12}$$

or, in a matrix form,

$$\boldsymbol{X} = A\boldsymbol{F} + \boldsymbol{U}. \tag{13}$$

The elements a_{ij} of A are called **factor loadings**[22], the random vector $\boldsymbol{F}$ is called the vector of **common factors**, and the random vector $\boldsymbol{U}$ is called the vector of **specific factors**, specific variates, or uniqueness. The factor loadings are assumed to be unknown parameters of the model.

Hence, we consider a model similar to the multivariate regression, but in factor analysis the "regressors" or "explanatory variables" $F_1, \ldots, F_m$ are assumed to be random, and cannot be directly observed[23].

Usual theoretical assumptions are that the common factors $\boldsymbol{F}$ and uniqueness $\boldsymbol{U}$ are uncorrelated, i.e., $Cov(\boldsymbol{F}, \boldsymbol{U}) = 0_{m \times p}$, the common factors themselves are standardized and uncorrelated, which means that the variance-covariance matrix of $\boldsymbol{F}$ is I_m[24], and the variance-covariance matrix of $\boldsymbol{U}$ is assumed to be a diagonal matrix $D = \mathrm{diag}(d_1, \ldots, d_p)$. Note that the variances $d_1, \ldots, d_p$ are additional unknown parameters of the model.

The mean values of $\boldsymbol{F}$ and $\boldsymbol{U}$ are assumed to be 0, which directly implies that the mean value of $\boldsymbol{X}$ is 0. In this text, we also adopt a simplifying assumption that the manifest variables $X_1, \ldots, X_p$ are normalized, i.e., $Var(X_i) = 1$, which is common in applications[25].

[22] That it, the $p \times m$ matrix A is called the matrix of factor loadings.

[23] Another difference is that in factor analysis, we allow the variances of the components of the "error vector" $\boldsymbol{U}$ to differ, while in regression analysis the errors are usually assumed to be homoscedastic.

[24] This is then called the orthogonal model, as opposed to a non-orthogonal model, where no restrictions on the variance-covariance matrix of $\boldsymbol{F}$ are imposed.

[25] Especially in some behavioral sciences, where X_i represent standardized responses of subjects.

The theoretical assumptions, together with the standard transformation rules stated in Theorem 1.2 imply that the correlation matrix Ψ of the vector $\boldsymbol{X}$[26] is

$$\Psi = AA^T + D. \tag{14}$$

Note that the elements of the matrix A are simply correlations between the manifest variables and hidden factors:

Theorem 4.1 (Interpretation of factor loadings)**.** *Under the model of factor analysis described above, we have* $a_{ij} = \rho(X_i, F_j)$ *for all* $i = 1, \ldots, p$ *and* $j = 1, \ldots, m$.

Proof. Consider the model (13). We have $Cov(\boldsymbol{X}, \boldsymbol{F}) = Cov(A\boldsymbol{F} + \boldsymbol{U}, \boldsymbol{F}) = ACov(\boldsymbol{F}, \boldsymbol{F}) + Cov(\boldsymbol{U}, \boldsymbol{F}) = A$, taking into account assumptions $Cov(\boldsymbol{F}, \boldsymbol{F}) = I_m$ and $Cov(\boldsymbol{U}, \boldsymbol{F}) = 0_{p\times m}$. Since the variables of $\boldsymbol{X}$ as well as of $\boldsymbol{F}$ are normalized, we obtain that $\rho(X_i, F_j) = cov(X_i, F_j) = a_{ij}$. □

A crucial observation is that the model (13) is over-parametrized: Based on the observations of the manifest variables $\boldsymbol{X}$, we will never be able to distinguish between models $\boldsymbol{X} = A\boldsymbol{F} + \boldsymbol{U}$ and $\boldsymbol{X} = (AV)(V^T\boldsymbol{F}) + \boldsymbol{U}$, where V is any $m \times m$ orthogonal matrix. In other words, the factor loadings A with factors $\boldsymbol{F}$ provide as good a fit to the observations as factor loadings AV with "rotated" factors $V^T\boldsymbol{F}$[27].

The previous point is the crux of the main criticism of factor analysis from the point of view of traditional statistical philosophy. Note that statisticians traditionally assume a single, fixed model, which is unknown, yet possible to identify with increasing precision as the sample size increases. Clearly, the model of factor analysis does not fit into this view. Rather, in factor analysis, we can adopt the attitude that *all* models of the form $\boldsymbol{X} = (AV)(V^T\boldsymbol{F}) + \boldsymbol{U}$, where V is an orthogonal matrix, are equally good representations of reality, and it is perfectly justifiable to select one of these models that leads to our understanding of the origin of data, based on a clear interpretation of what do the factors $V^T\boldsymbol{F}$ represent. In fact, there is a large body of literature on methods called "rotation of factors" that help us find a suitable matrix V; see the section on rotation of factors.

[26] Note that since the manifest variables are standardized, the correlation matrix of $\boldsymbol{X}$ coincides with the variance-covariance matrix Σ of $\boldsymbol{X}$.

[27] Note that if $\boldsymbol{F}$ satisfies the assumptions of factor analysis, then so does $V^T\boldsymbol{F}$.

4.1 Estimation of factor loadings

The first step in factor analysis is to use the observations of the manifest variables to find some appropriate estimates of the parameters a_{ij} and d_i. Here, two methods are most popular: **the method of maximal likelihood** and **the method of principal factors** which we very briefly describe.

Consider the formula $\Psi = AA^T + D$. If we knew $D = \mathrm{diag}(d_1, \ldots, d_p)$, then we would be able to evaluate the so-called **reduced correlation matrix** $\Psi - D = AA^T$, which would enable us the computation of a matrix A of factor loadings. Note that $d_i = 1 - h_i^2$, where $h_i^2 = \sum_{j=1}^m a_{ij}^2$ are called **communalities**, i.e., the estimation of the variances of specific factors is essentially equivalent to the estimation of communalities.

The method of principal factors uses the sample correlation matrix R as an estimator of Ψ, and $\hat{d}_1 = 1/(R^{-1})_{11}, \ldots, \hat{d}_p = 1/(R^{-1})_{pp}$ as (initial) estimators of $d_1, \ldots, d_p$. Thus, we can obtain an estimate

$$R^* = R - \mathrm{diag}(\hat{d}_1, \ldots, \hat{d}_p)$$

of the reduced correlation matrix, which can be viewed as the sample correlation matrix with diagonal elements replaced by the estimates $\hat{h}_1^2 = 1 - 1/(R^{-1})_{11}, \ldots, \hat{h}_p^2 = 1 - 1/(R^{-1})_{pp}$ of the communalities. Then we calculate the eigenvalues $\lambda_1 \geq \ldots \geq \lambda_p$ of R^* and the corresponding orthonormal eigenvectors $u_1, \ldots, u_p$. If the first m eigenvalues of R^* are non-negative, and the remaining eigenvalues are "small", we can estimate the matrix of factor loadings as

$$\hat{A} = (\sqrt{\lambda_1}u_1, \ldots, \sqrt{\lambda_m}u_m).$$

We remark that once we have $\hat{A}$, we can repeat the process using new estimates $\sum_{j=1}^m (\hat{A})_{ij}^2$, $i = 1, \ldots, p$, of communalities and iterate it until possible convergence.

In the method of principal factors the number m of factors is considered satisfactory based on analogous criteria as in the method of principal components, for instance, if $\sum_{j=1}^m \lambda_j / \sum_{i=1}^p |\lambda_i| > 0.8$.

4.2 Rotation of Factors

As mentioned above, if we have any estimate $\hat{A}$ of A, and V is any $m \times m$ orthogonal matrix, then $\tilde{A} = \hat{A}V$ is as good an estimate of A as $\hat{A}$. For a given estimate $\hat{A}$, the methods of factors rotation provide, in a sense, optimal rotation V^* in order to achieve a suitable form of the new matrix $\hat{A}V^*$ of

factor loadings. A general aim is to achieve the so called **simple structure** of the matrix of factor loadings. Omitting details, a matrix of factor loadings has a simple structure, if it contains many entries close to 0, 1 and -1, because, viewed as correlations, such factor loadings usually lead to a simple interpretation of common factors.

Definition 4.1 (Varimax and Quartimax rotation methods)**.** *Varimax rotation of the $p \times m$ matrix $\hat{A}$ of factor loadings is the orthogonal matrix V of the type $m \times m$ that maximizes the value of*

$$f_v(V) = \frac{1}{m}\sum_{t=1}^{m}\left[\frac{1}{p}\sum_{j=1}^{p}\left((\hat{A}V)_{jt}^2 - \frac{1}{p}\sum_{k=1}^{p}(\hat{A}V)_{kt}^2\right)^2\right]. \tag{15}$$

Quartimax rotation of the $p \times m$ matrix $\hat{A}$ of factor loadings is the orthogonal matrix V of the type $m \times m$ that maximizes the value of

$$f_q(V) = \frac{1}{mp}\sum_{t=1}^{m}\sum_{j=1}^{p}\left((\hat{A}V)_{jt}^2 - \frac{1}{mp}\sum_{s=1}^{m}\sum_{k=1}^{p}(\hat{A}V)_{ks}^2\right)^2. \tag{16}$$

Let B_V be the $m \times p$ matrix with elements corresponding to the squares of the elements of $\hat{A}V$, that is, $B_V = \hat{A}V \odot \hat{A}V$, where $\odot$ is the Hadamard (entry-wise) product of matrices. Observe that it is possible to interpret (15) as the average "sample variance" of the entries of the columns of B_V, and (16) can be interpreted as the "sample variance" of all elements of B_V. Clearly, the elements of $\hat{A}V$ are correlations, i.e., the elements of B_V are bounded by 0 from below and by 1 from above. Therefore, the maximization of the "variance" of the elements of B_V forces the elements to be close to 0 or 1, i.e., forces the elements of $\hat{A}V$ to be close to the numbers 0, 1, and -1.

Using the formula $\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \frac{1}{N}\sum_{i=1}^{N}x_i^2 - \bar{x}^2$ for any $x_1, \ldots, x_N$ and the observation

$$\sum_{s=1}^{m}\sum_{k=1}^{p}(\hat{A}V)_{ks}^2 = \|\hat{A}V\|^2 = \mathrm{tr}(\hat{A}V(\hat{A}V)^T) = \mathrm{tr}(\hat{A}\hat{A}^T) = \|\hat{A}\|^2,$$

the quartimax utility function can be simplified as follows:

$$f_q(V) = \frac{1}{mp}\sum_{t=1}^{m}\sum_{j=1}^{p}(\hat{A}V)_{jt}^4 - \frac{1}{(mp)^2}\|\hat{A}\|^4.$$

Therefore, the quartimax method maximizes the sum of the fourth powers of factor loadings, hence the name.

The problem of finding optimal V in the sense of the varimax or the quartimax rotation method is a difficult problem of mathematical programming and description of the appropriate numerical methods goes beyond this text. However, all advanced statistical packages implement some form of factor rotation optimization.

4.3 Estimation of Factor Scores

Lemma 4.1. *Let the random vector $(\boldsymbol{X}^T, \boldsymbol{F}^T)^T$ follows the $p+m$-dimensional normal distribution with the zero mean value. Then the conditional distribution of $\boldsymbol{F}$ given $\boldsymbol{X} = \boldsymbol{x}$ is*

$$N_m\left(Cov(\boldsymbol{F},\boldsymbol{X})\Sigma_{\boldsymbol{X}}^{-1}\boldsymbol{x},\ \Sigma_{\boldsymbol{F}} - Cov(\boldsymbol{F},\boldsymbol{X})\Sigma_{\boldsymbol{X}}^{-1}Cov(\boldsymbol{X},\boldsymbol{F})\right) = N_m\left(A^T(AA^T+D)^{-1}\boldsymbol{x},\ I_m - A^T(AA^T+D)^{-1}A\right).$$

Consider a random sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ of the p-dimensional observable variables on n objects, satisfying the model of factor analysis. Suppose that our aim is to estimate the realizations of the common factors $\boldsymbol{F}_1, \ldots, \boldsymbol{F}_n$ for individual objects; these realizations are called **factor scores**. Lemma 4.1 motivates the following estimators:

$$\hat{\boldsymbol{F}}_r = \hat{A}^T(\hat{A}\hat{A}^T + \hat{D})^{-1}\boldsymbol{X}_r,\ r = 1, \ldots, n, \tag{17}$$

where $\hat{A}$ is the selected estimate of the matrix of factor loadings and $\hat{D}$ is the estimate of the diagonal matrix of variances of specific variates. This kind of estimation of factor scores is sometimes called the **method of regression analysis**[28].

If the number p is very large, it may be difficult to compute the inverse of the $p \times p$ correlation matrix $\hat{R} = \hat{A}\hat{A}^T + \hat{D}$[29]. However, the matrix $\hat{A}^T\hat{R}^{-1}$ that appears in the formula (17) can be computed by only inverting a usually much smaller $m \times m$ matrix, and a $p \times p$ diagonal matrix:

Lemma 4.2. *Let $R = AA^T + D$ be a non-singular matrix, where A is a $p \times m$ matrix and D is a $p \times p$ matrix. Then $A^TR^{-1} = (I_m + A^TD^{-1}A)^{-1}A^TD^{-1}$.*

Proof.

$$\begin{aligned}(I_m + A^TD^{-1}A)^{-1}A^TD^{-1}R &= \\ (I_m + A^TD^{-1}A)^{-1}A^TD^{-1}(AA^T + D) &= \\ (I_m + A^TD^{-1}A)^{-1}(A^TD^{-1}A + I_m)A^T &= A^T.\end{aligned}$$

□

[28] The reason is that it can also be motivated by techniques similar to multivariate regression analysis.

[29] It was especially difficult in the past.

5 Partitioning Methods of Cluster Analysis

The general aim of clustering is to reveal relationships or similarities of n objects, typically characterized by either multidimensional real-valued vectors of features or by a matrix of mutual dissimilarities. In particular, the aim of the *partitioning* methods of cluster analysis[30] is to divide the objects into k groups called clusters, in such a way that the elements within the same cluster are as similar as possible and the elements of different clusters are as dissimilar as possible. It is the specification of the notions of "as similar as possible" and "as dissimilar as possible" that distinguishes different partitioning methods.

First, let us introduce some notation. Let n be the number of objects and let k be the number of clusters. That is, the objects will be denoted by the numbers $1, \dots, n$ and the clusters by the numbers $1, \dots, k$. We will assume that $n \geq k$; the case $n < k$ is meaningless and in real situations, the number n of clusters is usually greater than k by several orders of magnitude.

Definition 5.1 (Clustering and clusters). *Let $\Gamma_{n,k}$ be the set of all vectors $\gamma \in \mathbb{R}^n$ with elements from $\{1, \dots, k\}$, such that each of the values $1, \dots, k$ occurs at least once in γ*[31]*. Any vector $\gamma \in \Gamma_{n,k}$ will be called a "clustering" (or "partitioning"). The set $C_j(\gamma) = \{i \in \{1, \dots, n\} : \gamma_i = j\}$ will be called the j-th cluster for the clustering γ*[32]*. The number $n_j(\gamma)$ of elements of $C_j(\gamma)$ will be called the size of the j-th cluster for the clustering γ.*

If the objects are characterized by features $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^p$, we can define the following characteristics for each cluster $C_j(\gamma)$, $\gamma \in \Gamma_{n,k}$.

Definition 5.2 (Centroid of a cluster). *Let $\gamma \in \Gamma_{n,k}$ and let $j \in \{1, \dots, k\}$. The centroid of the cluster $C_j(\gamma)$ is*[33]

$$\bar{\boldsymbol{x}}_j(\gamma) = n_j^{-1}(\gamma) \sum_{i \in C_j(\gamma)} \boldsymbol{x}_i.$$

[30] Partitioning methods are sometimes called non-hierarchical methods to highlight the contrast with the hierarchical methods of cluster analysis.

[31] As we will see, the interpretation of this requirement is that all clusters contain at least one object.

[32] That is, for $j \in \{1, \dots, k\}$ and $i \in \{1, \dots, n\}$ the equality $\gamma_i = j$ means that the clustering γ assigns the i-th object into the j-th cluster.

[33] Note that the centroid of a cluster is its "center of mass", if the objects are assumed to have mass 1 and $\boldsymbol{x}_i$ is the position of the i-object, $i = 1, \dots, n$.

Definition 5.3 (Variance-covariance matrix of a cluster)**.** *Let $\gamma \in \Gamma_{n,k}$ and let $j \in \{1, \dots, k\}$. The variance-covariance matrix of the cluster $C_j(\gamma)$ is*[34]

$$S_j(\gamma) = n_j^{-1}(\gamma) \sum_{i \in C_j(\gamma)} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma))(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma))^T.$$

Partitioning methods differ in the way they define the notion of "optimal" clustering $\gamma^* \in \Gamma_{n,k}$ and in the way they search for this clustering[35].

5.1 Clustering using k-means and k-medoids

The best known and popular partitioning method is called "k-means". The basic idea of k-means is to first fix k[36] and then select $\gamma_k^* \in \Gamma_{n,k}$ that minimizes

$$\mathrm{Err}_k(\gamma) := \sum_{j=1}^{k} \sum_{i \in C_j(\gamma)} d^2(\boldsymbol{x}_i, \bar{\boldsymbol{x}}_j(\gamma)), \tag{18}$$

where d is some distance measure, usually the Euclidean distance $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$, which we will further assume. In words, k-means tries to find the partitioning γ_k^* that minimizes the sums of squared distances of objects to the centroid of their clusters.

The definition of Err_k implies that k-means can only produce "convex" clusters with a tendency towards a spherical shape. An important drawback of k-means is that its result depends on the units in which the individual components of the feature-vectors are expressed. Moreover, the computation of the minimum of Err_k is in general a difficult problem of discrete optimization. In some special cases, the optimal γ^* is not unique and there is a large number of local optima of the objective function[37], some of which can be very different from what we would intuitively deem to be a good partitioning. However, k-means often does provide a reasonable clustering and there exist simple optimization heuristics that rapidly lead to small values of Err_k.

[34] The Variance-covariance matrix is a matrix representation of the "shape" of the cluster. Note, however, that in some pathological cases it can be singular.

[35] Finding the optimal optimal clustering is typically a difficult algorithmic problem, or a problem of discrete optimization.

[36] The number k of clusters can be known or it can be determined using various methods that we mention later. For now, we assume that k is given.

[37] Of course, the notion of a local optimum requires that we define what we mean a "neighbourhood" of a partitioning γ, but the claim that the problem can have many local optima is valid for most natural definitions of the structure of neighbourhoods.

One such heuristic is the so-called Lloyd's algorithm, which creates a sequence $\gamma_0, \gamma_1, \ldots, \gamma_{\text{stop}}$ of partitionings as follows:

1. Set $t \leftarrow 0$ and create γ_0 randomly.
2. Until a given limit $t_{\max}$ on t is reached or no reassignment of objects occurs[38] repeat the following steps:
   (a) Compute the centroids $\bar{\boldsymbol{x}}_j(\gamma_t)$ for all clusters $j = 1, \ldots, k$.
   (b) Create γ_{t+1} by assigning objects $i = 1, \ldots, n$ to the nearest centroid $\bar{\boldsymbol{x}}_j(\gamma_t)$ of the previous clustering[39].
   (c) Set $t \leftarrow t + 1$.

Since the Lloyd's algorithm can get stuck in a local optimum, it is common to run it multiple times (with multiple initial clusterings γ_0), and select the one of the final clusterings γ_{stop} that provides the minimum value of Err_k. Note that there exist some more efficient (and more complex) heuristics for minimizing Err_k than the basic Lloyd's method.

An interesting modification of k-means is the method called k-medians. Note that k-means requires that we compute the centroids of clusters, which in turn assumes that each object is characterised by a real-valued vector of features. In some cases, however, we only have an $n \times n$ matrix D of mutual "dissimilarites" of all pairs of objects.

The matrix D (with elements D_{ij}) is enough to define the so-called medoid of a set $C \subseteq \{1, \ldots, n\}$. A medoid of C is any[40] object $i^C \in C$ minimizing the average dissimilarity to all other objects of C, i.e.,

$$\sum_{i \in C} D_{ii^C} = \min_{l \in C} \sum_{i \in C} D_{il}.$$

The method of k-medoids then tries to choose the clustering γ_k^D that minimizes

$$\text{Err}_k^D(\gamma) := \sum_{j=1}^{k} \sum_{i \in C_j(\gamma)} D_{ii^{C_j(\gamma)}}.$$

[38] That is $\gamma_t = \gamma_{t-1}$. Note that this must occur sooner or later.

[39] Formally, this means that $\gamma_{t+1}(i) = j$ if $\|\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma_t)\| \leq \min_{l=1,\ldots,k} \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}_l(\gamma_t)\|$. If the current centroid is one of the centroids closest to $\boldsymbol{x}_i$, then we do not change the cluster of i. All other ambiguities in the assignment of the closest centroid are resolved randomly.

[40] usually uniquely defined

Note that the medoids of the optimal clustering γ_k^D can be considered as "the most typical representatives" of individual clusters. Compared to k-means, the partitioning obtained by k-medians can be more robust to outliers. Computing optimal clustering with respect to the method of k-medoids is a difficult problem, similarly to k-means. An analogue of the Lloyd's algorithm for is here a method called "partitioning around medoids", or PAM.

Both k-means and k-medoids assume that we already know the appropriate number k of clusters. However, in most application, k is not known in advance[41]. There are several rules of thumb of selecting the "best" k. The most common is an "elbow" diagram, similar to the one that we used for selecting an appropriate number of principal components. In this approach, we plot the piece-wise linear function interpolating the points

$$(1, \mathrm{Err}_1(\gamma_1^*)), (2, \mathrm{Err}_2(\gamma_2^*)), \ldots, (K, \mathrm{Err}_K(\gamma_K^*))$$

for some reasonably large K. We will then choose the value of k^* that corresponds to an "elbow" in this diagram. An intuitive explanation of this procedure is that increasing the number of clusters from k^* to k^*+1 has only a small effect on the improvement of the "fit" of the data by the centroids.

Naturally we can also plot the elbow diagram for k-medoids, with Err_k^D instead of Err_k. Nevertheless, a graphical method typically used to find the appropriate k for k-medoids (and provide additional information about the quality of a given k-medoid clustering) is the so-called silhouette plot. This is a simple heuristic method an interested readers are referred to abundant online materials on this topic.

5.2 Model-based Clustering

In this approach, the optimal clustering γ^* is defined by means of an underlying model assumption. More precisely, we assume that the n objects are characterized by feature vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$ that are realizations of random vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ which are independent, with a distribution that depends on γ, i.e., γ is considered to be a model parameter. In the model-based clustering, we obtain the optimal clustering as an estimate $\hat{\gamma}$ of the parameter γ, typically by the principle of maximum likelihood.

First, let us analyse the following very simple model: $\boldsymbol{X}_i \sim N_p(\mu_{\gamma_i}, \sigma^2 I_p)$ for all $i = 1, \ldots, k$, where $\gamma \in \Gamma_{n,k}$, $\sigma > 0$ is known and I_p denotes the $p \times p$

[41] As as exercise, try to find some problems, where the number k of clusters is known in advance.

identity matrix. Here, the unknown model parameters are $\hat{\gamma}, \hat{\mu}_1, \ldots, \hat{\mu}_k$ and the parametric space is $\Gamma_{n,k} \times \mathbb{R}^p \times \cdots \times \mathbb{R}^p$. The likelihood function is

$$L(\gamma, \mu_1, \ldots, \mu_k | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \prod_{i=1}^{n} f_{\mu_{\gamma_i}}(\boldsymbol{x}_i), \tag{19}$$

where

$$f_\mu(\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \mu\|^2\right)}{(2\pi\sigma^2)^{p/2}};\ \boldsymbol{x} \in \mathbb{R}^p$$

is the density of $N_p(\mu_{\gamma_i}, \sigma^2 I_p)$. The log-likelihood of (19) is

$$\ln L(\gamma, \mu_1, \ldots, \mu_k | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \ln \prod_{i=1}^{n} f_{\mu_{\gamma_i}}(\boldsymbol{x}_i) =$$

$$\sum_{i=1}^{n} \left\{ -\frac{p}{2} \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \|\boldsymbol{x}_i - \mu_{\gamma_i}\|^2 \right\} = A - B \sum_{i=1}^{n} \|\boldsymbol{x}_i - \mu_{\gamma_i}\|^2,$$

where A, B are positive constants. Therefore, the maximum likelihood estimate $\hat{\gamma}, \hat{\mu}_1, \ldots, \hat{\mu}_k$ minimizes $\sum_{i=1}^{n} \|\boldsymbol{x}_i - \mu_{\gamma_i}\|^2$.

For a fixed $\gamma \in \Gamma_{n,k}$ we have

$$\sum_{i=1}^{n} \|\boldsymbol{x}_i - \mu_{\gamma_i}\|^2 = \sum_{j=1}^{k} \sum_{i \in C_j(\gamma)} \|\boldsymbol{x}_i - \mu_{\gamma_i}\|^2 = \sum_{j=1}^{k} \sum_{i \in C_j(\gamma)} \|\boldsymbol{x}_i - \mu_j\|^2.$$

Now, recall that the point that minimizes the sum of the squared Euclidean distances to the points of a finite set C is the center of mass of C. That is, minimizing $\sum_{i \in C_j(\gamma)} \|\boldsymbol{x}_i - \mu_j\|^2$ with respect to μ_j leads to the solution $\hat{\mu}_j = \bar{\boldsymbol{x}}_j(\gamma)$, i.e., the centroid. Consequently, to find the maximum likelihood estimate of $\gamma \in \Gamma_{n,k}$, we need to minimize

$$\sum_{j=1}^{k} \sum_{i \in C_j(\gamma)} \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma)\|^2, \tag{20}$$

which is exactly the problem (18) of k-means. Therefore, the clustering based on the simplest meaningful multivariate normal model is exactly the k-means clustering. Incidentally, this clarifies why the k-means method has the tendency to create not only spherical clusters, but spherical clusters of similar size.

Now, we could gradually relax the assumptions on the covariance matrices of the normal distributions of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$, i.e., taking σ^2 as unknown, taking

the covariance matrices equal to $\sigma_{\gamma_1} I_p, \ldots, \sigma_{\gamma_n} I_p$ with unknown and possibly different $\sigma_1, \ldots, \sigma_k > 0$, etc. However, here we will directly move to the most general assumption: $\boldsymbol{X}_i \sim N_p(\mu_{\gamma_i}, \Sigma_{\gamma_i})$, where $\gamma \in \Gamma_{n,k}$, $\mu_1, \ldots, \mu_k \in \mathbb{R}^p$, and $\Sigma_1, \ldots, \Sigma_k \in \mathcal{S}^p_{++}$, all unknown.

By $\Gamma^R_{n,k}$ denote the set of all "regular" clusterings, i.e., clusterings $\gamma \in \Gamma_{n,k}$ such that the variance-covariance cluster matrices $S_1(\gamma), \ldots, S_k(\gamma)$ are non-singular. By

$$\hat{\gamma}, \hat{\mu}_1, \ldots, \hat{\mu}_k, \hat{\Sigma}_1, \ldots, \hat{\Sigma}_k$$

denote the model parameters that, on the set $\Gamma^R_{n,k} \times \mathbb{R}^p \times \cdots \times \mathbb{R}^p \times \mathcal{S}^p_{++} \times \cdots \times \mathcal{S}^p_{++}$ [42], maximize the likelihood function

$$L(\gamma, \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \prod_{i=1}^{n} f_{\mu_{\gamma_i}, \Sigma_{\gamma_i}}(\boldsymbol{x}_i), \tag{21}$$

where

$$f_{\mu,\Sigma}(\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu)^T \Sigma^{-1}(\boldsymbol{x}-\mu)\right)}{(2\pi)^{p/2}\sqrt{\det(\Sigma)}};\ \boldsymbol{x} \in \mathbb{R}^p$$

is the density of the distribution $N_p(\mu, \Sigma)$ with a positive definite variance-covariance matrix Σ. In the sequel, we will again derive an optimization problem of a simple form which provides the maximum likelihood estimate $\hat{\gamma}$, without the need to explicitly compute $\hat{\mu}_1, \ldots, \hat{\mu}_k, \hat{\Sigma}_1, \ldots, \hat{\Sigma}_k$.

Consider the log-likelihood function of (21):

$$\begin{aligned}
&\ln L(\gamma, \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \ln \prod_{i=1}^{n} f_{\mu_{\gamma_i}, \Sigma_{\gamma_i}}(\boldsymbol{x}_i) \\
&= \sum_{i=1}^{n} \left[-\frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln\det(\Sigma_{\gamma_i}) - \frac{1}{2}(\boldsymbol{x}_i - \mu_{\gamma_i})^T \Sigma_{\gamma_i}^{-1}(\boldsymbol{x}_i - \mu_{\gamma_i}) \right] \\
&= \sum_{j=1}^{k} \sum_{i \in C_j(\gamma)} \left[-\frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln\det(\Sigma_j) - \frac{1}{2}(\boldsymbol{x}_i - \mu_j)^T \Sigma_j^{-1}(\boldsymbol{x}_i - \mu_j) \right].
\end{aligned}$$

For any fixed $\gamma \in \Gamma^R_{n,k}$ and $j \in \{1, \ldots, k\}$ the sum

$$\sum_{i \in C_j(\gamma)} \left[-\frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln\det(\Sigma_j) - \frac{1}{2}(\boldsymbol{x}_i - \mu_j)^T \Sigma_j^{-1}(\boldsymbol{x}_i - \mu_j) \right],$$

[42] $\mathcal{S}^p_{++}$ is the set of all positive-definite $p \times p$ matrices.

attains the maximum for $\hat{\mu}_j = \bar{\boldsymbol{x}}_j(\gamma)$ a $\hat{\Sigma}_j = S_j(\gamma)$[43]. Moreover, the constant $-\frac{p}{2}\ln(2\pi)$ plays no role in our optimization problem, therefore $\hat{\gamma}$ is the solution of the problem:

$$\text{argmax}_{\gamma\in\Gamma^R_{n,k}} \sum_{j=1}^{k} \left[-n_j(\gamma)\ln\det(S_j(\gamma)) - \sum_{i\in C_j(\gamma)} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma))^T S_j^{-1}(\gamma)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma)) \right] \tag{22}$$

Using the basic properties of the trace we obtain that for any $\gamma \in \Gamma^R_{n,k}$ and $j \in \{1,\ldots,k\}$:

$$\begin{aligned}
&\sum_{i\in C_j(\gamma)} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma))^T S_j^{-1}(\gamma)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma)) \\
&= \sum_{i\in C_j(\gamma)} \text{tr}\left[(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma))^T S_j^{-1}(\gamma)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma))\right] \\
&= \text{tr}\left[S_j^{-1}(\gamma) \sum_{i\in C_j(\gamma)} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma))(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j(\gamma))^T \right] \\
&= \text{tr}\left[S_j^{-1}(\gamma) n_j(\gamma) S_j(\gamma)\right] = \text{tr}\left[n_j(\gamma) I_p\right] = p n_j(\gamma),
\end{aligned}$$

which means that the optimization problem (22) is equivalent to

$$\text{argmin}_{\gamma\in\Gamma^R_{n,k}} \sum_{j=1}^{k} n_j(\gamma)\ln\det(S_j(\gamma)) + \frac{p}{2}\sum_{j=1}^{k} n_j(\gamma).$$

Clearly, $\sum_{j=1}^{k} n_j(\gamma) = n$, which does not depend on γ, therefore

$$\hat{\gamma} = \text{argmin}_{\gamma\in\Gamma^R_{n,k}} \sum_{j=1}^{k} n_j(\gamma)\ln\det(S_j(\gamma)). \tag{23}$$

Note that, roughly speaking, the optimal clustering is the one that minimizes a weighted average of logarithms of "volumes" of clusters[44]. The form of the problem (23) is simple, but its generally difficult to find its solution. Often, the algorithm used to compute $\hat{\gamma}$ is the so-called EM algorithm[45] or some algorithm of stochastic combinatorial optimization.

[43] The proof of this statement is analogous to the proof that the sample mean and the sample variance-covariance matrix are the maximum likelihood estimates based on the random sample from the multivariate normal distribution.

[44] As an exercise, justify the view that $\det(S_j(\gamma))$ is relates to the "volume" of $C_j(\gamma)$.

[45] EM means "Expectation-Maximization".

Model-based clustering (under the most general assumptions) is significantly more difficult – theoretically and computationally – than the standard clustering methods; however, it has several advantages. For instance, the version of the model based clustering that we described above[46] is invariant under changes of the scale of variables (changes of the units of measurements), and can detect "clusters inside other clusters". One disadvantage of the model-based clustering is that it cannot be directly used if we only know the matrix of dissimilarities of the objects (we will see examples of this phenomenon).[47]

5.3 Clustering using the density-based scan

A powerful partitioning method of completely different kind is the so-called density-based scan (DBScan). Here, the clustering is not defined as a solution of an optimization problem, but as a split of objects into implicitly defined classes.[48]

Let $\epsilon > 0$ and minPts ≥ 3 be parameters. Let us define the classes as follows:

1. An object i^* is a "core object" if the feature vectors of at least minPts of objects are within the distance ϵ from the feature vector of i^*, i.e., if

$$|N_\epsilon(\boldsymbol{x}_{i^*})| := |\{i \in \{1,\dots,n\} : \|\boldsymbol{x}_i - \boldsymbol{x}_{i^*}\| \leq \epsilon\}| \geq \text{minPts},$$

   where $|\cdot|$ denotes the size of a set.

2. An object i is "directly reachable" from i^* if i^* is a core point and $\|\boldsymbol{x}_i - \boldsymbol{x}_{i^*}\| \leq \epsilon$.

3. An object i_r is "reachable" from an object i_1^*, if there is a sequence $i_1^*, i_2^*, \dots, i_{r-1}^*, i_r$ of objects such that i_{l+1}^* is directly reachable from i_l^* for all l such that $1 \leq l \leq r-2$, and i_r is directly reachable from i_{r-1}^*.

[46] Note that there are many types of "model-based clustering", for instance there are methods that impose restrictions on the underlying variance-covariance matrices of the clusters.

[47] Note, however, that we can be creative and suggest the following application of the model based clustering even in the case that we only know D: first, use some form of the multidimensional scaling (which uses only D) to embed the object to $\mathbb{R}^k$. Then apply the model based clustering to this embedding.

[48] An alternative view is that the DBScan clustering is defined via an *algorithm* that computes this split.

4. All objects that are not reachable from any other object are classified as "noise".

5. Two objects i_a and i_b are "density-connected" if there is a point i^*, such that i_a is reachable from i^* as well as i_b is reachable from i^*.

6. It is clear that density-connectedness is an symmetric relation[49] on the set "non-noise" objects. DBScan clustering is any partitioning of non-noise objects into classes such that two objects are in the same class only if they are density-connected.

Let us compare DBScan to other clustering methods that we already introduced.

First, a unique advantage of DBScan is that it can produce a special class of "noise" objects, which can be viewed as outliers; this makes DBScan robust to outliers. Second, DBScan does not require to choose the number k of clusters; the number of clusters is automatically determined by ϵ and minPts. Third, DBScan can identify clusters of various shapes, for instance two interlocked banana-shaped clusters, which is unattainable using the previous methods. Fourth, DBScan can be easily modified to work with only a dissimilarity matrix, similarly to k-medoids. Fifth, there exists an efficient algorithm that computes the DBScan clustering.[50]

The most important disadvantage of DBScan is that it can be very sensitive to the choice of the two parameters; it may be difficult to choose them properly.[51] Moreover, DBScan cannot split the objects into clusters of significantly different densities and the DBScan clustering generally depends on the units of variables (units of measurement of individual features); these two problems do not occur with the model based clustering in the form that we presented in Subsection 5.2. A minor disadvantage is that the result of DBScan is not fully deterministic (some "border" points may end up in different clusters depending on the actual algorithm and the order of the objects in the database).

[49] It is not, however, an equivalence relation, because it is not transitive. There are some variants of the DBScan principle that lead to the relation of equivalence, and therefore a uniquely defined split into the equivalence classes.

[50] We will not describe the algorithm in this text, and for applications it is not in fact necessary to know it (unlike the principle of the method itself).

[51] There is an entire body of literature on choosing ϵ and minPts which goes beyond the scope of the current version of this text.

A related algorithm is called "Optics" (ordering points to identify the clustering structure). This algorithm also depends on two parameters ϵ and minPts; however, the result of Optics is not a split into clusters (plus noise objects), but a specific *ordering* of the objects, such that consecutive points are "close"[52] in the feature space. This ordering is then used to construct the so-called "reachability plot", as follows.

For $i \in \{1, \ldots, n\}$ we set the "core distance" $d_{core}(\boldsymbol{x}_i)$ as undefined if $|N_\epsilon(\boldsymbol{x}_i)| <$ minPts and $d_{core}(\boldsymbol{x}_i) = \|\boldsymbol{x}_i - \boldsymbol{x}_{i^+}\|$, where $i^+ \in N_\epsilon(\boldsymbol{x}_i)$ is the index of the minPts-th closest point to $\boldsymbol{x}_{i^*}$. If $i \neq j \in \{1, \ldots, n\}$ we set the "reachability distance" $d_{reach}(\boldsymbol{x}_j|\boldsymbol{x}_i)$ of $\boldsymbol{x}_j$ from $\boldsymbol{x}_i$ as undefined if $|N_\epsilon(\boldsymbol{x}_i)| <$ minPts and

$$d_{reach}(\boldsymbol{x}_j|\boldsymbol{x}_i) = \max\{d_{core}(\boldsymbol{x}_i), \|\boldsymbol{x}_j - \boldsymbol{x}_i\|\}.$$

Let $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n$ be the sequence of feature vectors in the ordering generated by Optics. To each $i \in \{2, \ldots, n\}$ we will assign the reachability distance $r_i = d_{reach}(\tilde{\boldsymbol{x}}_i|\tilde{\boldsymbol{x}}_{i-1})$. The "reachability plot" is then the plot of values r_i as they depend on $i = 2, \ldots, n$. The "valleys" in this plot suggest blocks of nearby points possibly forming clusters and the "peaks" suggest transitions between separated clusters.

Optics has somewhat similar properties to DBScan. Compared to DBScan, Optics is better equipped to identify clusters of various densities. However, Optics is still not fully invariant even under linear transformations of the variables. For instance, the reachability plot (and hence the clustering) may change if we change the units of the measurement of a single variable.

5.4 Using clustering for anomaly detection

The identification of anomalies (or outliers) is directly inbuilt into the DBScan method - the noise objects are "anomalous" by definition. By decreasing the value of ϵ, we can obtain a list of candidates, with decreasing "magnitude of anomalousness". Such an ordering is exactly what we require if our aim is to closely inspect anomalies one by one but our inspection capacities may be limited, for example in the process of insurance fraud detection.

However, other clustering methods can also be used for anomaly detection. For instance, we can deem small clusters to be automatically anomalous, or order the objects based on the distance to the nearest centroid/medoid of a cluster.

[52] Again, we will not go into the very complex specifics of the algorithm itself.

Alternatively, we can order the objects according to the estimated density of the mixture of normal distributions (in the case of the model-based clustering): If γ^* is the optimal clustering based on the mixture of normal distributions, then an approximation of the data-generating density can be

$$\hat{f}(\boldsymbol{x}) = \sum_{j=1}^{k} \frac{n_j(\gamma^*)}{n} f_{\bar{\boldsymbol{x}}_j(\gamma^*), S_j(\gamma^*)}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^p,$$

where $f_{\mu,\Sigma}$ is the density of $N_p(\mu, \Sigma)$. If $\hat{f}(\boldsymbol{x}_i)$ is very small, then the object i can be an outlier or anomaly.

Several techniques of identification of outliers and anomalies can also be based on discrimination/classification methods which we study next.

6 Classification methods in general

The aim of the classification analysis is to construct a **classifier** $\mathcal{C}$ for **predicting** the **classes** of objects based on the **vectors of features** of the objects.[53] To this end, we must specify m classes, or categories. Often, the classification is **binary**[54], that is, $m = 2$, but more than two categories, the so-called **multiclass**[55] classification, are also common.

The construction of $\mathcal{C}$ is based on the **training data set**, which consists of n objects described by their vectors of features $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$[56], together with their known correct classifications $c_1, \ldots, c_n \in \{1, \ldots, m\}$. The classifications of the objects in the training set are pre-determined by an expert or by another reliable method, which, however, tends to be costly or time consuming. Our goal is to develop an inexpensive and rapid classification rule, usually implemented as a computer program, that can categorize future objects using only their known vectors of features[57].

[53] The classes of objects are sometimes called **predicted variables** and the vectors of features **explanatory variables**.

[54] For instance "acceptable/unacceptable", "female/male", "infected/noninfected", etc.

[55] For example the age of a person from m age categories, the soil type from m possible types, a disease from m possible diseases, etc.

[56] Note that if we do not directly have real vectors of features of objects, we can usually **embed** the objects into $\mathbb{R}^p$ using various methods. For instance, if we can meaningfully measure the distances between the objects, we can use the multidimensional scaling for the embedding. Also, if the number of features is too large to process, we can use various techniques of **dimensionality reduction** or **feature extraction**.

[57] Naturally, it is rarely the case that we are able to construct an error-free classification rule, but we can often achieve an acceptably small error rate. The reliability of a classifier is an important topic and will be discussed in the next sections.

Note the distinction between partitioning clustering and classification methods. While clustering methods work only with the data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$ representing the vectors of features and try to devise a "natural" split of objects into categories without any additional information[58], classification methods try to mimic the categorization $c_1, \ldots, c_n \in \{1, \ldots, m\}$ provided by a "teacher" in order to automatically classify new objects, without further help from the "teacher". That is why the clustering methods are sometimes called **unsupervised learning**, while the classification methods are called **supervised learning**.

6.1 Representations of classifiers

Often, a classifier is formally represented via real-valued **discrimination functions** $l_1, \ldots, l_m$ defined on $\mathbb{R}^p$. In an abstract way, the value $l_j(\boldsymbol{x})$ quantifies the "degree of belief" of the classifier that the object with features $\boldsymbol{x}$ belongs to the class j.[59] Such a classifier is sometimes called **soft**. If we systematically convert the values $l_1(\boldsymbol{x}), \ldots, l_m(\boldsymbol{x})$ into a single "winning" category $j \in \{1, \ldots, m\}$ that maximizes[60] $l_j(\boldsymbol{x})$, we obtain a **hard** classifier.[61] It is important to add that $l_1, \ldots, l_m$ can be very complicated functions, sometimes requiring a complex algorithm to evaluate.

Any hard classifier[62] can also be represented by a decomposition of $\mathbb{R}^p$ into m **classification regions** $R_1, \ldots, R_m$, $1 \leq i < j \leq m \Rightarrow R_i \cap R_j = \emptyset$. Here, the object with features $\boldsymbol{x}$ is classified into the i-th category if and only if $\boldsymbol{x}$ falls into R_i.

Clearly, any system $l_1, \ldots, l_m$ of discrimination functions automatically generates classification regions $R_1, \ldots, R_m$; a possible choice can be formally

[58] Except for the *number* of classes. Some clustering techniques, such as the DBScan, do not even require the number of classes.

[59] If $l_1(\boldsymbol{x}), \ldots, l_m(\boldsymbol{x})$ are non-negative and sum to one, they can be viewed as probabilities of individual categories. However, note that the decision that the object be classified into the class i can also be based on other factors, not only on the probability that it originated from the class i, but also on the degree of loss incurred by an incorrect classification.

[60] An observant student could ask what if there are two distinct categories j_1 and j_2 such that both $l_{j_1}(\boldsymbol{x})$ and $l_{j_2}(\boldsymbol{x})$ are maximal. In that case we usually select the winning category at random. In this text, we can for simplicity assume that we select the smallest of the maximizing indices. In practice, this does not happen too frequently.

[61] Of course, by converting a soft classifier into a hard classifier we loose some information. The soft classifier is always more informative than the corresponding hard classifier, but in most applications we are ultimately forced to make a definitive decision about the class of the object.

[62] In this study material, we will mostly focus on the hard classifiers.

defined by $\boldsymbol{x} \in R_j \Leftrightarrow j = \min\left(\text{argmax}_i l_i(\boldsymbol{x})\right)$, $\boldsymbol{x} \in \mathbb{R}^p$. The inverse, that is, converting $R_1, \ldots, R_m$ into a system of discrimination functions is also simple; we can just use the indicator functions of the regions. However, such a system of discriminant functions does not provide nuanced differences in the degrees of beliefs.

Various classification methods differ in the way they utilize the training set of data to construct either the discrimination functions, or the decomposition $R_1, \ldots, R_m$.

6.2 Reliability of a classifier

The most prominent characteristic of a classifier is its "reliability".[63] Assessing the reliability of a classifier is in fact an intricate problem, both theoretically and practically. To put it on a mathematically firm ground, let us adopt the following model of how the data are produced: Let us assume that for each object its class $j \in \{1, \ldots, m\}$ is first generated with **prior class probabilities** $q_1, \ldots, q_m$[64] and then, depending on j (and independent of the previous objects) its vector of features is generated from the **class density** p_j on $\mathbb{R}^p$.[65] For this section we will formalize the classifier via the decomposition $R_1, \ldots, R_m$ of $\mathbb{R}^p$.[66]

Now, the reliability of our classifier can be represented by two related matrices: First, we could define the matrix with elements

$$P(j|i) = \int_{R_j} p_i(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \quad i, j \in \{1, \ldots, m\},$$

which is the probability that an object from class i will be categorized into class j. That is, $P(j|i)$ is the "conditional" probability that the object will be categorized into the class j provided we (only) know that it belongs to the class i.

[63] Many sources use the term "performance" to denote what I call reliability. However, performance is a too general term which would presumably cover for instance the speed of the classification and other characteristics of the classifier. The term reliability indicates that here we only focus on the frequency of errors.

[64] Less formally, the probabilities $q_1, \ldots, q_m$ represent the frequencies of the m classes.

[65] It is straightforward to use general probability distributions of features, which could also formally describe discrete (or any) random variables. We choose to only work with densities for simplicity.

[66] That is, we will only deal with hard classifiers in this section; it more difficult to estimate the precision of a soft classifier. One method of measuring the performance of a predictor which predicts *probabilities* of classes if the so-called Brier score, which is out of the scope of this text.

However, the most commonly used is the so-called **confusion matrix** matrix $\mathbf{C}$, in which the element $\mathbf{C}_{ij}$ is the probability that a "generic" object will be from the class i and, at the same time, it will be categorized into the class j.[67] Clearly, $\mathbf{C}_{ij} = q_i P(j|i)$ for any $i, j \in \{1, \ldots, m\}$ and, for a fixed i, the row sums of $\mathbf{C}$ is the prior frequency of the class i:

$$\sum_{j=1}^{m} \mathbf{C}_{ij} = \sum_{j=1}^{m} q_i P(j|i) = q_i \sum_{j=1}^{m} P(j|i) = q_i.$$

We could also define the column sums of $\mathbf{C}$, say $r_j = \sum_{i=1}^{m} \mathbf{C}_{ij}$. The interpretation is that r_i is the probability that a generic object will be *classified* to the class j. The sum of all elements of $\mathbf{C}$ is 1.

The off-diagonal elements of $\mathbf{C}$ are called theoretical **miss-classification rates**. Clearly, large values of the diagonal elements of $\mathbf{C}$ and small values of the non-diagonal elements of $\mathbf{C}$, i.e., small miss-classification rates, signify a reliable classifier. However, there is no universal way of comparing confusion matrices. That is, we often cannot unambiguously decide which of the classifiers A and B is better, based only on their confusion matrices $\mathbf{C}_A$ and $\mathbf{C}_B$. To this end, we need a real-valued measure defined on the set of possible confusion matrices.

One such measure is based on the expected "gain" from the classification. Let $G(j|i)$ be the estimated gain that we obtain from classifying an object into the category j, if the correct classification of the object is the category i.[68] Then, the expected gain is $\sum_{i=1}^{m} \sum_{j=1}^{m} G(j|i)\mathbf{C}_{ij}$.

If we are unsure about the gains, we can set $G(i|i) = 1$ for all i and $G(j|i) = 0$ for all $i \neq j$. This measure is sometimes call the **accuracy** of the classifier, and can also be interpreted as the probability of the correct classification of a generic new object. Note however, that sometimes this is a misleading measure.[69] There are many other possible measures of precision of a classifier, most of which are specialized to binary classification (that is $m = 2$); we will mention them later.

[67] This is a "normalized" version of the confusion matrix, which can be interpreted via probabilities. Often, a non-normalized version of the confusion matrix is used representing the frequencies, i.e., $n\mathbf{C}$, where n is the number of objects. Or, more precisely, an estimate of $\mathbf{C}$ or $n\mathbf{C}$ is used, as described in the next subsection.

[68] Notice that this is closely related to the loss L from the section on the Bayes classifier.

[69] For instance if one category is frequent, for instance if $q_1 = 0.9$, then the trivial classifier that classifies all objects into the category 1 has accuracy 0.9.

6.3 Estimation of misclassification rates

We could in principle calculate the matrix $\mathbf{C}$ from the probabilities $q_1, \ldots, q_m$ and the densities $p_1, \ldots, p_m$. However, a fundamental problem is that in practical applications we do not know these characteristics. Therefore, our only hope is to use some method of estimating the rates of correct classification and miss-classification rates from the available data. To this end, the most popular are the so-called **resampling methods**.

Suppose that we have a method $\mathcal{A}$, often a computer algorithm, of constructing a classifier from any training set $(\mathbf{X}, \mathbf{c})$.[70] Here, $\mathbf{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)^T$ is the matrix of the feature vectors of the objects in the training set, and $\mathbf{c} = (c_1, \ldots, c_k)$ is the vector of "correct" classifications of the objects, sometimes called **labels**. That is, given any training set $(\mathbf{X}, \mathbf{c})$ as the input, the method $\mathcal{A}$ produces some classifier $\mathcal{C} = \mathcal{C}(\mathbf{X}, \mathbf{c}, \mathcal{A})$ represented by a decomposition $R_1, \ldots, R_m$ of $\mathbb{R}^p$. How do we estimate the real misclassification rates of $\mathcal{C}$ applied to future data?

The simplest approach, sometimes called the **in-sample method**, or the "re-substitution" method, is the following: We first apply $\mathcal{A}$ to the full training dataset to obtain our classifier $\mathcal{C}$. Next, we go through all objects of the training set one by one and predict their classes using $\mathcal{C}$; let us say that the predicted classes are $\tilde{c}_1, \ldots, \tilde{c}_n$. Then, the estimate of the element $(c, \tilde{c})$ of $\mathbf{C}$ is the proportion of the pairs $(c_i, \tilde{c}_i)$, $i = 1, \ldots, n$, equal to $(c, \tilde{c})$. This method typically yields overly optimistic estimates of $\mathbf{C}$.[71]

From the category of the **out-of-sample methods**, the simplest is the **validation-set method**:[72] We first split the full training dataset into two parts, the training part $(\mathbf{X}^t, \mathbf{c}^t)$ and the validation part $(\mathbf{X}^v, \mathbf{c}^v)$, and use $\mathcal{A}$ to create a classifier $\mathcal{C} = \mathcal{C}(\mathbf{X}^t, \mathbf{c}^t, \mathcal{A})$.[73] Next, we use $\mathcal{C}$ to classify the objects from the validation set; let us say that the validation set is composed of the last k objects of the full training set[74] and their categories predicted by $\mathcal{C}$ are

[70] $(\mathbf{X}, \mathbf{c})$ can be for instance the original, full training set, or its proper subset.

[71] Intuitively, the reason is that we test the classifier based on the same data that had been used for its training.

[72] Some literature uses a different and more nuanced terminology: the validation set is a set of data used in the course of training and the so-called "test set" is a completely independent dataset used for the evaluation of the final model. Confusingly, sometimes the terms are used in the opposite meaning.

[73] $(\mathbf{X}^t, \mathbf{c}^t)$ usually contains a fixed proportion, say 80% of objects selected from the full training set.

[74] Often, the training set is produced by a random selection of $n - k$ objects, but for the notational simplicity we assume here that it is the first $n - k$ objects.

$\tilde{c}_{n-k+1}, \ldots, \tilde{c}_n$. Then, the estimate of the element $(c, \tilde{c})$ of $\mathbf{C}$ is the proportion of the pairs $(c_i, \tilde{c}_i)$, $i = n-k+1, \ldots, n$, equal to $(c, \tilde{c})$. This method often yields good estimates of the real miss-classification rates of $\mathcal{C}$. However, its disadvantage is that it is not trained on the full dataset, i.e., it may be less precise than ít could have been if trained on the the full dataset.[75]

Finally, we will mention advanced and more computationally demanding out-of-sample estimation methods, called **cross-validation**. In cross-validation, we do construct the classifier $\mathcal{C}$ from the entire training dataset, but to estimate its miss-classification rates, we effectively rely on systematic aspects of the general method $\mathcal{A}$ that we used for constructing $\mathcal{C}$.

A simple cross-validation method is called **leave-one-out**. Here, we repeat the validation-set method n times. In the i-th loop, $i = 1, \ldots, n$, we take the object i to be a single-point validation set, construct the classifier $\mathcal{C}_i$ on the training data excluding the object i, and produce an estimate $\hat{\mathbf{C}}_i$ of the matrix $\mathbf{C}$.[76] The final estimate of $\mathbf{C}$ is the average of matrices $\hat{\mathbf{C}}_i$. The leave-one-out method generally leads to better estimates of $\mathbf{C}$ than the in-sample method and does not waste training data. However, a big disadvantage is that we need to construct n classifiers. In many applications, the number n of objects is thousands or even millions, and the computation time of this validation method would be prohibitive.

Probably the most frequently used cross-validation method is called the **k-fold cross-validation**, where the validation-set method is repeated k times. We split the training data randomly into k subsets[77] $(\mathbf{X}^1, \mathbf{c}^1), \ldots, (\mathbf{X}^k, \mathbf{c}^k)$. In the t-th loop, $t = 1, \ldots, k$, we take $(\mathbf{X}^t, \mathbf{c}^t)$ to be the validation set, construct the classifier $\mathcal{C}_t$ on all training data excluding $(\mathbf{X}^t, \mathbf{c}^t)$, and produce an estimate $\hat{\mathbf{C}}_t$ of the matrix $\mathbf{C}$. Similarly as in the leave-one-out method, the final estimate of $\mathbf{C}$ is the average of the matrices $\hat{\mathbf{C}}_t$. Clearly, leave-one-out is a special case of the k-fold cross-validation for $k = n$. However, in practice we usually choose k to be much smaller than n.[78] The k-fold cross-validation seems to be a good compromise between the precision of the estimation of $\mathbf{C}$ and the computation time to produce the estimate.

There are many other variants of cross-validation, for instance leave-p-out, repeated random sub-sampling cross-validation, and so on. Also, some

[75] In a sense, we exchanged the precision of the classifier for the precision of the estimation of its miss-classification rates.

[76] This will be a very coarse estimate with just zeros, except for one number 1 at the position determined by the real and the predicted class of the i-th object.

[77] For simplicity, we assume that n is a multiple of k.

[78] A common choice is $k = 10$.

classification methods may have their own, specific, techniques for the estimation of the miss-classification rates.

6.4 Other important properties of a classifier

While the precision of a classifier is an important characteristic, there is a number of other factors that we must take into account when choosing a classifier for our application. Here are some of them:[79]

- How difficult is to construct/prepare the classifier? That is, how complex is the method $\mathcal{A}$? Some classifiers are based on straightforward algebra, such as the classifier corresponding to the classical linear discriminant analysis (LDA), others require relatively advanced algorithms to construct, for example the classification trees (CTs), or classification forests (CFs). Still other classifiers need solving a difficult optimization problem, a typical examples are the artificial neural networks (ANNs). Note that in the machine learning literature, the process of construction of the classifier is informally called **training**.

- How simple/rapid is the process of classification of new objects once the classifier is ready to use? That is, how fast can we evaluate the discriminant functions or find the region R_i that contains a given $\boldsymbol{x}$? Some classifiers are fast to apply, for instance the LDA, or the support-vectors machine (SVM) classifier. A shallow CT can be applied rapidly even without a computer. However, some methods require time-consuming computation to classify new objects (for instance the k-nearest neighbours classifier, KNN, especially without advanced tricks and for a large training set).

- How easy or difficult is to "understand" the inner workings of the classifier? Can we easily summarize the reason why did the classifier decide for one or the other class; can *we* learn something from the classifier? In some methods, such as LDA and CT, we can directly "see" the classification rule, while the reason for the chosen classification of some methods, most famously the ANNs, are notoriously difficult to understand. In the literature, this problem is called the **problem of interpretability** and the classifiers that are difficult to understand are often called **black-box classifiers**. The black-box classifiers are not suitable for some specific applications, for instance if we must be sure about the robustness or "fairness" of the results.

[79] You may wish to review this list later when you will known details about several kinds of classifiers.

There are many other considerations when choosing the best classifier: Do we require a soft classifier or a hard one is enough?[80] Can the classifier be applied only to binary classification or to a multi-class classification?[81] Can unequal class frequencies $q_1, \ldots, q_m$ be incorporated into the process of training of the classifier? How easy is to update the classifier if we happen to obtain new training data? Can the classifier directly handle the categorical explanatory variables? Can it deal with missing data (i.e., missing parts of the feature vector of the object to be classified)? And so on.

In general, the appropriateness of the classifier strongly depends on various specifics of the application as well as on the training data. It is often impossible to say in advance which classification method is the best one; we usually attempt to use several of them and select a well-performing method empirically. However, a very rough rule of thumb is that the simplest methods (CT, LDA, linear SVM) should be chosen for small training data with a simple structure, while huge and complicated training data are best handled with more complex methods (KNN, CF, ANN).

6.5 Binary classification

An important special case is the binary classification ($m = 2$). A (hard) binary classifier is fully determined by a decomposition R_1, R_2, where $R_1 \cup R_2 = \mathbb{R}^p$, $R_1 \cap R_2 = \emptyset$. The boundary between the sets R_1 and R_2 is called the **decision boundary**.

Let us call the two categories "positives" and "negatives".[82] It is usual to name the elements of the normalized confusion matrix $\mathbf{C}$ as follows: $\mathbf{C}_{11}$ is the **true positives** rate, $\mathbf{C}_{12}$ is the **false negatives** rate, $\mathbf{C}_{21}$ is the **false positives** rate, and $\mathbf{C}_{22}$ is the **true negatives** rate.[83] Note that the sum of the true positive and the false negative rates is q_1, i.e., the overall frequency of the positives in the "population" and the sum of the true negative and the false positive rates is q_2, i.e., the overall frequency of the negatives in the population.

[80] For instance the classical CTs are hard classifiers and it is not completely straightforward to modify them for soft classification.

[81] For standard SVM is just for the binary classification, although there exist sophisticated multi-class SVMs.

[82] This is a terminology particularly common in medical applications, where the medical test is in fact a (physical) classifier.

[83] In the Babylon of the machine learning literature, the true positive rate is sometimes call the "probability of detection" and the false positive rate is called the "probability of false alarm". If instead of the normalized confusion matrix we use a non-normalized version, we simply use the terms true positives, true negatives, false positives, false negatives.

The ratio of the true positives rate to q_1, schematically $P(+|+)$ is called the **sensitivity** of the classifier/test. The ratio of the true negatives rate to q_2, that is, $P(-|-)$, is called the **specificity** of the classifier/test.[84]

Besides sensitivity and specificity, there exists a plethora of other measures of the reliability of a classifier. We mentioned some of them, in particular the accuracy, in subsection 6.2. Another popular measure is the **Pearson-Yule phi coefficient**, that is the standard correlation coefficient calculated from $\mathbf{C}$ seen as the probability mass function of two binary random variables.[85]

Let us also mention the so-called "positive predicted value", or **precision**, formally defined as $\mathbf{C}_{11}/(\mathbf{C}_{11}+\mathbf{C}_{21})$. Precision is analogous to sensitivity: while sensitivity is the probability that an object will be classified to the class 1 provided that it is from the class 1, precision is the probability that an object is from the class 1 provided that it has been classified to the class 1. A good classifier has high values of both sensitivity and precision. Therefore, to quantify the "reliability" of a classifier by a single number, some researchers use the so-called **F1-score**, which is simply the harmonic mean of sensitivity and precision.[86]

For some classification methods, the classifier is fully determined by a single discrimination function l defined on $\mathbb{R}^p$ and a threshold $t \in \mathbb{R}$. That is, the object characterized by $\boldsymbol{x}$ is classified into the first category if $l(\boldsymbol{x}) \geq t$ and into the second category if $l(\boldsymbol{x}) < t$.[87] Compatible classification regions are then $R_1 = \{\boldsymbol{x} \in \mathbb{R}^p : l(\boldsymbol{x}) \geq t\}$ and $R_2 = \{\boldsymbol{x} \in \mathbb{R}^p : l(\boldsymbol{x}) < t\}$. The decision boundary is then the set $\{\boldsymbol{x} \in \mathbb{R}^p : l(\boldsymbol{x}) = t\}$. If l is linear (and the sets R_1 and R_2 are half-spaces and the decision boundary is a hyperplane), such a classifier is usually called **linear**, in the opposite case it is called **non-linear**.

Suppose that the function l is known[88], but we are allowed to select the threshold t. By changing t we can obtain a continuum of classifiers $\mathcal{C}_t$, each

[84] In some fields of application, the authors use the term "recall" or "hit rate" instead of sensitivity and the term "selectivity" instead of specificity.

[85] In machine learning literature, this is called the "Matthews correlation coefficient" in honour of some Matthews, who reinvented the coefficient more than half a century later than Pearson and Yule.

[86] Note that the harmonic mean of two numbers approaches 0 provided that *any* of the two numbers approaches 0.

[87] That is, we can formally define the first discriminant function from the previous sections by $l_1(\boldsymbol{x}) = l(\boldsymbol{x}) - t$ and the second one by $l_2(\boldsymbol{x}) = t - l(\boldsymbol{x})$.

[88] That is, we assume that the classification method at hand provides the function l per se, not only the partition R_1, R_2 defined by some automatically determined t.

with its own miss-classification rates. Note that if t decreases, the region R_1 expands. This means that with decreasing t, the rate of the true positives increases (ultimately it reaches 1), but the rate of the false positives also increases (also reaches 1 in the limit).[89] Similarly, with increasing t, the rate of the true positives decreases (ultimately it reaches 0), and the rate of the false positives also decreases (reaches 0 in the limit).

Let us now make a plot of all points with coordinates "(1 - specificity, sensitivity)", formally $(\mathbf{C}_{21}/q_2, \mathbf{C}_{11}/q_1)$.[90] The set of points will form a curve in the square $[0,1] \times [0,1]$ called **receiver operating characteristic curve**[91], or the **ROC curve**. Note that the ROC curve will be non-decreasing, it starts at $(0,0)$ and ends at $(1,1)$.[92] We can visually inspect the ROC curve and select the most appropriate t, depending on whether we prefer a high sensitivity or a high specificity, or, most likely, a reasonable compromise. Of course, the theoretically ideal situation is if there is some t for which the ROC curve hits the point $(0,1)$, because it signifies a perfect classifier. While this is usually unattainable in practice, this idea inspires a simple method of measuring the quality of the set of classifiers parametrized by the threshold t (it can be viewed as the quality of the function l itself) by a single number: compute the **area under the ROC curve**, abbreviated by **AUC**. The closer is AUC to 1, the better overall quality of the classifier.[93]

7 Bayes classifier

7.1 The general Bayes classifier

The practical performance of our classifier can be influenced by the "cost" $L(i|j)$ of miss-classification of an object from the class j into the class i. For example, if $L(i|j)$ is huge[94] for some pair of classes i, j, then we should be very careful about the miss-classification rate $\mathbf{C}_{ji}$.

[89] Since q_1 and q_2 are fixed, decreasing t implies increasing the sensitivity but decreasing the specificity.

[90] Note again that $\mathbf{C}_{21}, \mathbf{C}_{11}$ depend on t, although it is not explicit in our notation.

[91] The term is so strange because this curve was first used by electrical engineers.

[92] Except for some pathological cases.

[93] The AUC has also a nice probabilistic interpretation, but we will skip it for now.

[94] Suppose that we have a classifier that reads the signal from the traffic lights to inform an autonomous car. The possible classes are now "green", "yellow" and "red". Clearly, the miss-classification "red → green" incurs potentially catastrophic costs, therefore the number $L(\text{green}|\text{red})$ is very large.

For the theoretical development, suppose that we know the underlying probabilistic distributions $p_1, \ldots, p_m$ of the feature vectors of objects from categories $1, \ldots, m$, and also suppose that we know the probabilities $q_1, \ldots, q_m$ of individual classes, that is, q_i is the frequency with which we observe the objects from the class i. In this hypothetical case, it is possible to construct a theoretically optimal classification rule that takes the costs $L(i|j)$ into account, which we will call the **general Bayes classifier**.

To this end, we use the entire matrix $m \times m$ matrix $\mathbf{L}$ of losses from miss-classification, that is $\mathbf{L}_{ii} =: L(i|i) = 0$ for all i and $\mathbf{L}_{ij} =: L(i|j) > 0$ for $i \neq j$, which is the loss that we suffer if we classify an object into the category i, provided that the correct classification of the object is the category j.[95]

For a fixed discrimination rule $R_1, \ldots, R_m$, the probability that an object from category j will be classified into category i is

$$P(i|j) = \int_{R_i} p_j(\boldsymbol{x}) d\boldsymbol{x},$$

therefore the probability of the even "the classified object is from the class j and, at the same time, it will be classified to the class i" is

$$\mathbf{C}_{ji} = q_j \int_{R_i} p_j(\boldsymbol{x}) d\boldsymbol{x}.$$

Hence, the mean loss from the classifier $R_1, \ldots, R_m$ is

$$E_{R_1,\ldots,R_m} = \sum_{i=1}^{m} \sum_{j=1}^{m} L(i|j)\, q_j \int_{R_i} p_j(\boldsymbol{x}) d\boldsymbol{x}.$$

Surprisingly, it turns out that it is simple to minimize the mean loss $E_{R_1,\ldots,R_m}$ with respect to the choice of the decomposition $R_1, \ldots, R_m$, obtaining thus the theoretical Bayes rule. Indeed, note that

$$E_{R_1,\ldots,R_m} = \sum_{i=1}^{m} \int_{R_i} h_i(\boldsymbol{x}) d\boldsymbol{x}, \tag{24}$$

where

$$h_i(\boldsymbol{x}) = \sum_{j=1}^{m} L(i|j)\, q_j p_j(\boldsymbol{x}) \text{ for all } i = 1, \ldots, m$$

[95] Note that the matrix $\mathbf{L}$ does not need to be symmetric; it is often the case that the loss $L(i|j)$ is very different from the loss $L(j|i)$ as in the example with the traffic lights.

are fixed functions. Clearly, in order to minimize (24) it is enough to choose any measurable decomposition $R_1^*, \ldots, R_m^*$ satisfying

$$R_i^* \subseteq \{\boldsymbol{x} \in \mathbb{R}^p : h_i(\boldsymbol{x}) = \min_k h_k(\boldsymbol{x})\} \text{ for all } i = 1, \ldots, m. \tag{25}$$

The selection of optimal rule in (25) can be written more explicitly for two (not mutually exclusive) special cases. First, if $m = 2$, we can choose[96]

$$\begin{aligned} R_1^* &= \{\boldsymbol{x} \in \mathbb{R}^p : h_1(\boldsymbol{x}) \leq h_2(\boldsymbol{x})\} \\ &= \{\boldsymbol{x} \in \mathbb{R}^p : q_2 L(1|2) p_2(\boldsymbol{x}) \leq q_1 L(2|1) p_1(\boldsymbol{x})\} \\ &= \left\{\boldsymbol{x} \in \mathbb{R}^p : \frac{p_1(\boldsymbol{x})}{p_2(\boldsymbol{x})} \geq \frac{q_2 L(1|2)}{q_1 L(2|1)}\right\}, \end{aligned} \tag{26}$$

and $R_2^* = \mathbb{R}^p \setminus R_1^*$. Thus, the binary Bayes classification rule classifies an object with features $\boldsymbol{x}$ into category 1 if the **likelihood ratio** $p_1(\boldsymbol{x})/p_2(\boldsymbol{x})$ exceeds some fixed threshold determined by the prior probabilities q_1, q_2 and losses $L(1|2), L(2|1)$. The likelihood ratio can thus be viewed as the discrimination function l from Section 6.5 and the term $\frac{q_2 L(1|2)}{q_1 L(2|1)}$ as a threshold chosen optimally, provided that the aim is to minimize the expected loss.[97]

Second, assume a general number m of categories with the simplest loses given by $L(i|j) = 1$ for $i \neq j$.[98] We obtain from (25) that for any $i = 1, \ldots, m$:

$$\begin{aligned} R_i^* &\subseteq \{\boldsymbol{x} \in \mathbb{R}^p : \sum_{j=1, j\neq i}^{m} q_j p_j(\boldsymbol{x}) \leq \sum_{j=1, j\neq k}^{m} q_j p_j(\boldsymbol{x}) \text{ for all } k\} \\ &= \{\boldsymbol{x} \in \mathbb{R}^p : q_k p_k(\boldsymbol{x}) \leq q_i p_i(\boldsymbol{x}) \text{ for all } k\} \\ &= \{\boldsymbol{x} \in \mathbb{R}^p : q_i p_i(\boldsymbol{x}) = \max_k q_k p_k(\boldsymbol{x})\}. \end{aligned} \tag{27}$$

That is, for a vector $\boldsymbol{x}$ of features, the Bayes classifier selects the category i such that $q_i p_i(\boldsymbol{x})$ is maximal. Here, the functions $q_i p_i$ can be viewed as discrimination functions l_i from Section 6.1.

For this second special case (i.e., $L(i|j) = 1$ for $i \neq j$), we can also derive the same classifier by the following reasoning. Let Y be the categorical

[96] We can choose the class for those $\boldsymbol{x}$ such that $h_1(x) = h_2(\boldsymbol{x})$; any such choice leadts to the optimal, Bayes, classifier. For simplicity we will also assume that $p_2(\boldsymbol{x}) > 0$ for all $\boldsymbol{x}$.

[97] However, we can also vary the threshold and obtain the ROC as explained in Section 6.5. The threshold $t = \frac{q_2 L(1|2)}{q_1 L(2|1)}$ will provide a specific point on the ROC.

[98] Note that for these simplest losses the minimization of the expected loss $E_{R_1,\ldots,R_m}$ is the same as the maximization of the classifier's reliability measure called accuracy. That is, in this case the Bayes classifier maximizes the accuracy.

random variable that represents the class of a generic object to be classified in the future. Assume that we observe, for a given object, that the random vector $\boldsymbol{X}$ representing its features attained the vector $\boldsymbol{x} \in \mathbb{R}^p$. Then, the posterior probability of the event $[Y = i]$ is given by the Bayes theorem as

$$P(Y = i|\boldsymbol{X} = \boldsymbol{x}) = \frac{q_i p_i(\boldsymbol{x})}{\sum_{k=1}^{m} q_k p_k(\boldsymbol{x})}. \tag{28}$$

This means that the (hard) classifier (27) is exactly the one that selects the class with the maximum posterior probability; that is why it is also called the **maximum a posteriori classifier**. The functions at the right-hand side of (28) can be used for the more nuanced information about the class, i.e., what we called the "soft" classifier.

In practice we do not know the densities $p_1, \ldots, p_m$ nor the prior probabilities $q_1, \ldots, q_m$, but we can use several reasonable methods of estimating the densities[99], as well as the prior probabilities.

7.2 Naive Bayes classifier

The so-called **naive Bayes classifier** makes an assumption that all the variables forming vectors of features are independent.[100] In our notation, this means that, for each i the density p_i is a product of one-dimensional marginal densities $p_i^{(1)}, \ldots, p_i^{(p)}$:

$$p_i(x_1, \ldots, x_p) = \prod_{\ell=1}^{p} p_i^{(\ell)}(x_\ell), \text{ for all } i = 1, \ldots, m. \tag{29}$$

Admittedly, the assumption that all the features behave as independent random variables is unrealistic in all but the simplest cases. However, as a practical tool, the resulting classifier often performs acceptably well and can be our method of choice, if we prefer simplicity over performance.

The simplicity of the naive Bayes classifier is based on the fact that we only need to estimate the marginal densities to obtain an estimate of the joint density. The marginal densities can be estimated for instance by the univariate normal distribution

$$\widehat{p_i^{(\ell)}}(x_\ell) = f(x_\ell \mid \widehat{\mu_{i,\ell}}, \widehat{\sigma_{i,\ell}}^2), \tag{30}$$

[99] Often, it is enough to estimate the likelihood ratios $p_i(\boldsymbol{x})/p_j(\boldsymbol{x})$, or the position of the boundaries of the sets R_i^*.

[100] The independence is understood "conditionally for a given class", that is, provided that we know the class of an object, all its p features, i.e., explanatory random variables, are stochastically independent.

where $f(x \mid \mu, \sigma^2)$ denotes the density of the univariate normal distribution with mean μ and variance σ^2, and the parameter estimates $\widehat{\mu_{i,\ell}}$, $\widehat{\sigma_{i,\ell}}^2$ are computed in a standard way from the training data.[101] This is sometimes called **the Gaussian naive Bayes classifier**. Less coarse estimates of the marginal densities can be obtained via the so-called **kernel density estimation**.

In the literature and in practice, the naive Bayes classifier is typically applied with an implicit assumption that $L(i|j) = 1$ for $i \neq j$; in that case (27) implies that the object characterized by $\boldsymbol{x} = (x_1, \ldots, x_p)$ is classified to the category

$$i^* = \text{argmax}_i \left\{ q_i \prod_{\ell=1}^{p} \widehat{p}_i^{(\ell)}(x_\ell) \right\}. \tag{31}$$

Note that it is simple to modify the naive Bayes classifier as explained above to the case of discrete (even categorical) features.

7.3 Linear and quadratic discriminant analyses

Another classical simplification is to assume that the densities $p_1, \ldots, p_m$ of feature vectors are full-blooded multivariate normal, but all with the same, non-singular variance-covariance matrix Σ. That is,

$$p_i(\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu_i)^T\Sigma^{-1}(\boldsymbol{x}-\mu_i)\right)}{(2\pi)^{p/2}\sqrt{\det(\Sigma)}} \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^p,$$

where $\mu_1, \ldots, \mu_m \in \mathbb{R}^p$ are the mean value vectors of distributions $p_1, \ldots, p_m$. Under this assumption, the sets $R_1, \ldots, R_m$ turn out to be (possibly unbounded) polyhedrons[102]. We will describe the case of two categories in more detail.

Let $m = 2$. The log-likelihood ratio from (26) is for any $\boldsymbol{x} \in \mathbb{R}^p$

$$\begin{aligned} \ln\left(\frac{p_1(\boldsymbol{x})}{p_2(\boldsymbol{x})}\right) &= \frac{1}{2}(\boldsymbol{x}-\mu_2)^T\Sigma^{-1}(\boldsymbol{x}-\mu_2) - \frac{1}{2}(\boldsymbol{x}-\mu_1)^T\Sigma^{-1}(\boldsymbol{x}-\mu_1) \\ &= (\mu_1-\mu_2)^T\Sigma^{-1}\boldsymbol{x} + \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1. \end{aligned}$$

[101] That is: take the set of the values of the ℓ-th explanatory variable (the ℓ-th feature) of all objects from the i-th class. Compute the standard mean and the standard sample variance of this set of real numbers.

[102] In some cases, the optimal choice of $R_1, \ldots, R_m$ is not unique, but the sets can always be chosen as polytopes. The situation is, however, markedly different if the variance-covariance matrices of the normal distributions can be different. In that case, we obtain a "quadratic" classification rule.

Thus, if we denote[103] $\boldsymbol{a} := \Sigma^{-1}(\mu_1 - \mu_2)$, $b := \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1$, and $k := \frac{q_2 L(1|2)}{q_1 L(2|1)}$, we obtain that

$$R_1^* = \left\{\boldsymbol{x} \in \mathbb{R}^p : \boldsymbol{a}^T\boldsymbol{x} + b - \ln(k) \geq 0\right\}, \; R_2^* = \mathbb{R}^p \setminus R_1^*. \tag{32}$$

We obtain a linear classification rule, i.e., R_1^* and R_2^* are half-spaces and the decision boundary is a hyperplane.

Some special cases of the classifier (32) are worth a special attention. First, if the prior probabilities of the two classes are the same, i.e., $q_1 = q_2 = 0.5$ and the losses are symmetric, that is, $L(1|2) = L(2|1)$, then $k = 1$ and $\ln(k) = 0$ in (32). If, moreover, $\Sigma = \sigma^2 I_p$ for some $\sigma^2 > 0$, the hyperplane that sets R_1^* and R_2^* apart passes through the center of mass $(\mu_1 + \mu_2)/2$ of the mean value vectors μ_1, μ_2, and is orthogonal to $\mu_1 - \mu_2$, i.e., an object is classified into category 1 (or 2) if its feature vector $\boldsymbol{x}$ is closer to μ_1 than to μ_2 (or closer to μ_2 than to μ_1).

The expression (32) allows us to find simple explicit formulas for the misclassification probabilities $P(2|1)$ and $P(1|2)$. Let $\boldsymbol{X}_1 \sim N_p(\mu_1, \Sigma)$. Observing that $\boldsymbol{a}^T\boldsymbol{X}_1 \sim N(\boldsymbol{a}^T\mu_1, \boldsymbol{a}^T\Sigma\boldsymbol{a})$ we obtain

$$\begin{aligned} P(2|1) &= \int_{R_2} p_1(\boldsymbol{x})d\boldsymbol{x} = P[\boldsymbol{a}^T\boldsymbol{X}_1 < \ln k - b] = \\ &= P\left[\frac{\boldsymbol{a}^T\boldsymbol{X}_1 - \boldsymbol{a}^T\mu_1}{\sqrt{\boldsymbol{a}^T\Sigma\boldsymbol{a}}} < \frac{\ln k - b - \boldsymbol{a}^T\mu_1}{\sqrt{\boldsymbol{a}^T\Sigma\boldsymbol{a}}}\right] = \Phi\left(\frac{\ln k - b - \boldsymbol{a}^T\mu_1}{\sqrt{\boldsymbol{a}^T\Sigma\boldsymbol{a}}}\right), \end{aligned}$$

where Φ is the distribution function of the standardized normal. Denoting $\alpha = \sqrt{(\mu_1 - \mu_2)^T\Sigma^{-1}(\mu_1 - \mu_2)}$, which is sometimes called the **Mahalanobis distance** of μ_1 and μ_2, we can express the misclassification probability as

$$P(2|1) = \Phi\left(\frac{\ln k - \alpha^2/2}{\alpha}\right). \tag{33}$$

An analogous derivation yields

$$P(1|2) = 1 - \Phi\left(\frac{\ln k + \alpha^2/2}{\alpha}\right). \tag{34}$$

Naturally, $P(1|1) = 1 - P(2|1)$ and $P(2|2) = 1 - P(1|2)$ therefore we can obtain all 4 elements of $\mathbf{C}$ from $\mathbf{C}_{ij} = q_i P(j|i)$.

[103] We assume that $\mu_1 \neq \mu_2$.

As an interesting exercise related to $m \geq 3$, one can show that for the special case of $L(i|j) = 1$ for $i \neq j$ and $\Sigma = \sigma^2 I_p$, the classification sets R_i are the **Voronoi cells** of the expected value vectors $\mu_1, \ldots, \mu_k$. That is, the corresponding classifier selects the category for $\boldsymbol{x}$ based on the nearest expected value vector.

In practice, the parameters $\mu_1, \ldots, \mu_m$ and Σ are estimated from the training set as follows: $\hat{\mu}_i$ is the mean of the feature vectors assigned to the class c_i in the training set and $\mathbf{S}$, the estimate of Σ is **pooled** from the matrices $\mathbf{S}_1, \ldots, \mathbf{S}_m$ via

$$\mathbf{S} = \frac{(n_1 - 1)\mathbf{S}_1 + \ldots + (n_m - 1)\mathbf{S}_m}{n_1 + \ldots + n_m - m},$$

where n_i is the number, and $\mathbf{S}_i$ is the sample covariance matrix of the objects assigned to the class c_i in the training set.[104]

Even if the assumptions of normality are not satisfied[105], this approach provides *some* classifier which may perform reasonably well; we can estimate its miss-classification rates using the methods from Section 6.3.[106]

The methods described above lead to the so-called **Linear discriminant analysis** (LDA). More complex classification methods can be obtained by relaxing the rather stringent assumptions of multivariate normality with the same covariance matrix. For instance, we can assume that the classes have non-unique covariance matrices, i.e., the data are generated form a general mixture of normals. Then we obtain the method called the **Quadratic discriminant analysis** (QDA). Here, we separately estimate the means as well as covariance matrices of the explanatory vectors of each class. In the QDA the classification regions $R_1, \ldots, R_m$ are in general no longer polytopic, not even in the binary case ($m = 2$).[107]

7.4 K nearest neighbours

K nearest neighbours (KNN) is a simple but powerful approach to constructing a classifier: the object with features $\boldsymbol{x}$ is classified according to the classes

[104] Note once again that here we assume that the covariance matrices of all feature distributions p_i are the same. We also assume that each class has at least 2 representatives in the training set.

[105] Which is usually the case in real applications.

[106] If we have a binary classification problem and we are sure about the theoretical assumptions, we can use $\hat{\mu}_i$ and $\mathbf{S}$ to estimate the Mahalanobis distance of μ_1 and μ_2, and then use the formulas (34) and (33) to estimate the probabilities of false classification.

[107] Convince yourself that if $\Sigma_1 \neq \Sigma_2$ then $\ln(p_1(\boldsymbol{x})/p_2(\boldsymbol{x}))$ is a quadratic form in $\boldsymbol{x}$.

of the k objects of the training set that have features closest to $\boldsymbol{x}$. More precisely, for each feature vector $\boldsymbol{x}$ to be classified, the KNN classifier finds the k closest vectors $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}$ from the training set, and classifies $\boldsymbol{x}$ to the class which occurs most frequently in the set $c_{i_1}, \ldots, c_{i_k}$;[108] we call this the **majority vote**. The idea is very simple, but (for a large training set) it may be exceedingly time consuming to find the k nearest neighbours for a given $\boldsymbol{x}$.[109]

It is useful to view the KNN as a method inspired by the Bayes classification rule. In its simplest multi-class form ($L(i|j) = 1$ for all $i \neq j$ and $q_i = 1/m$ for all i), the Bayes rule (27) says: for $\boldsymbol{x}$ choose the category j with maximal $p_j(\boldsymbol{x})$. If the training set is approximately balanced with respect to the classes,[110] the KNN is a straightforward method of estimating exactly this - which class has the largest density of features in the point $\boldsymbol{x}$.[111]

8 Classification trees

The underlying structure of a standard classification tree[112] is that of a directed rooted binary[113] tree. There are two main categories of nodes of the classification tree: **non-terminal nodes** and **terminal nodes**. Each non-terminal node is a **parent** of exactly two **descendants**, and each node has exactly one parent, except for the **root node**, which has no parents[114]. To each non-terminal node we assign a condition (sometimes called a **split**) involving values of a single variable. Moreover, we assign the label "True" to one of the two edges emanating from the non-terminal node, and the label "False" to the other one. To each terminal node we assign a class label from $\{1, \ldots, m\}$, where m is the number of classes.[115]

[108] If there are several classes that are most frequent in the set $c_{i_1}, \ldots, c_{i_k}$, we select one at random. In the binary classification, we can choose k to be an odd number, to avoid such "ties".

[109] There are methods of how to significantly seed up the search for the k-nearest neighbours, especially if we allow occasional errors in their identification. This is a fascinating topic from the point of view of computer science, but we will skip it.

[110] That is, all classes have approximately same number of objects in the training set.

[111] Try to invent alternative methods for predicting which class has the largest density in $\boldsymbol{x}$. Use the optimal Bayes classifier to modify the basic KNN to take the probabilities q_i or the miss-classification costs C_{ij} into account.

[112] A classification tree is a special case of the structure called a **decision tree**.

[113] There exist non-binary classification trees but we will omit them in this text.

[114] Most often, the root node is non-terminal, but for some extreme cases the root node can be terminal, i.e., the classification tree classifies every object into a single category.

[115] Each terminal node can be assigned a more nuanced output information, for instance frequencies of classes, effectively resulting in the classifier which we call "soft". Here we

The process of classification of an object using a classification tree starts in the root node. The object is then passed down the tree via a path directed by the validity (or non-validity) of the conditions in the non-terminal nodes, until the terminal node is hit. The object is assigned to the class corresponding to the label of the terminal node.

Geometrically, the resulting classification $R_1, \ldots, R_m$ is formed by p-dimensional polytopic sets (some of them unbounded) with $(p-1)$-dimensional facets perpendicular to the basic unit vectors $\mathbf{e}_1, \ldots, \mathbf{e}_p$.

8.1 Constructing a classification tree using recursive partitioning

The construction of a classification tree is based on the objects in the training set, that is, objects with known true/correct classifications. We will say that a node τ (of a completely or partially constructed tree) "contains" a training set object, if the classification of the training object passes through τ.

The standard procedure for creating a classification tree is called **growing** or **recursive partitioning**[116] and can be outlined as follows:

1. Create a new "empty" node τ which is either the root node or a descendant of an existing non-terminal node.

2. Decide whether τ should be terminal or non-terminal. The node is declared to be terminal if it contains only objects of a single class, or if it contains less than some critical number of training objects, or if the node reached a pre-specified depth (distance from the root node).

3. If τ is terminal, assign a class label to τ. The class label is usually assigned based on the **majority vote**, i.e., the class label is the most frequent label of the objects contained in τ.[117]

4. If τ is non-terminal, assign a condition to τ. The common strategy of this step is:

will only consider the classification trees that assign just a single class to each terminal node.

[116] Naturally, we could formulate an optimization problem implicitly defining an "optimal" classification tree; recursive partitioning can then be viewed as a "greedy" method to obtain a possibly sub-optimal but practically often useful classification tree.

[117] Of course, the method for assigning the class to the terminal node can also take the expected frequencies of the classes and the miss-classification costs into account.

(a) Create a finite number of possible conditions that could be assigned to the node τ.

(b) Define a "discrimination quality" of each of the conditions for τ (sometimes called the **goodness of split**) and then select the best condition.

Step 4(a) is an easy one, and can be completed before the algorithm is started. For each variable X_i, $i = 1, \ldots, p$, do the following:

- If X_i is quantitative or ordinal, select a finite number of thresholds $\theta_i^{(j)}$, $j = 1, \ldots, n_i$, and add n_i conditions $X_i < \theta_i^{(1)}$, $\ldots$, $X_i < \theta_i^{(n_i)}$ to the pool of all permissible conditions.

- If X_i is nominal with possible values from a finite set M_i, add a system of conditions of the form $X_i \in M_i^{(j)}$, $j = 1, \ldots, n_i$, where $n_i = 2^{|M_i|-1} - 1$, such that $\emptyset, M_i^{(1)}, \ldots, M_i^{(n_i)}, M \setminus M_i^{(1)}, \ldots, M \setminus M_i^{(n_i)}, M$ are all $2^{|M|}$ subsets of M.

Thus, we obtained $\sum_{i=1}^{p} n_i$ permissible conditions to be assigned to τ. The more difficult question is how to construct a natural measure of the discrimination quality of each condition in the node τ.

Let CND be a fixed condition from the pool of all conditions permissible for a non-terminal node τ. Temporarily, assume that CND is assigned to τ. As the node τ is non-terminal, it will have two descendants, let us denote them as τ_L and τ_R. Suppose that all objects that satisfy CND at τ will be directed to τ_L and all objects that do not satisfy CND at τ will be directed to τ_R. For the sake of simplicity, we will assume that we only have two classes of objects - $CLS1$ and $CLS2$ (i.e., $m = 2$). Define the following numbers based on the training set[118]:

- n_{**} is the total number of objects that pass through τ.

- n_{*1} is the number of objects of $CLS1$ that pass through τ.

- n_{*2} is the number of objects of $CLS2$ that pass through τ.

- n_{1*} is the number of objects contained in τ that satisfy CND (i.e., they will be directed to τ_L).

- n_{2*} is the number of objects contained in τ that do not satisfy CND (i.e., they will be directed to τ_R).

[118] Draw a contingency table for a clarification.

- n_{11} is the number of objects of $CLS1$ that satisfy CND.
- n_{12} is the number of objects of $CLS2$ that satisfy CND.
- n_{21} is the number of objects of $CLS1$ that do not satisfy CND.
- n_{22} is the number of objects of $CLS2$ that do not satisfy CND.

A useful measure of the goodness of the split CND at τ can be defined as

$$\iota(\tau) - p_L\iota(\tau_L) - p_R\iota(\tau_R), \tag{35}$$

where $p_L = \frac{n_{1*}}{n_{**}}$, $p_R = \frac{n_{2*}}{n_{**}}$, and $\iota(\tau)$, called an **impurity index** is, loosely speaking, a measure of how unclear the classification would be provided that τ is a terminal node (similarly $\iota(\tau_L)$ and $\iota(\tau_R)$). Two common measures of impurity are the **entropy index** and the so-called **Gini index**. The entropy index is defined as:

$$\iota(\tau) = -\frac{n_{*1}}{n_{**}}\log_2\frac{n_{*1}}{n_{**}} - \frac{n_{*2}}{n_{**}}\log_2\frac{n_{*2}}{n_{**}}.$$

Observe that $\iota(\tau)$ is in fact the entropy of the discrete random variable that attains two values with probabilities $\frac{n_{*1}}{n_{**}}$ and $\frac{n_{*2}}{n_{**}}$. If one of the numbers $\frac{n_{*1}}{n_{**}}$ is equal to 1 (and the other number is equal to 0[119]), then the impurity $\iota(\tau)$ is 0. That is, if all objects in τ are from the same class, it has the minimal possible impurity. On the other hand, if $\frac{n_{*1}}{n_{**}} = \frac{n_{*2}}{n_{**}} = 1/2$ then the impurity $\iota(\tau)$ attains its maximal possible value.

Analogously, we set

$$\begin{aligned}
\iota(\tau_L) &= -\frac{n_{11}}{n_{1*}}\log_2\frac{n_{11}}{n_{1*}} - \frac{n_{12}}{n_{1*}}\log_2\frac{n_{12}}{n_{1*}},\\
\iota(\tau_R) &= -\frac{n_{21}}{n_{2*}}\log_2\frac{n_{21}}{n_{2*}} - \frac{n_{22}}{n_{2*}}\log_2\frac{n_{22}}{n_{2*}}.
\end{aligned}$$

For the Gini index, the impurities are defined by numerically somewhat simpler formulas that approximate the entropy:

$$\begin{aligned}
\iota(\tau) &= 1 - \left(\frac{n_{*1}}{n_{**}}\right)^2 - \left(\frac{n_{*2}}{n_{**}}\right)^2,\\
\iota(\tau_L) &= 1 - \left(\frac{n_{11}}{n_{1*}}\right)^2 - \left(\frac{n_{12}}{n_{1*}}\right)^2,\\
\iota(\tau_R) &= 1 - \left(\frac{n_{21}}{n_{2*}}\right)^2 - \left(\frac{n_{22}}{n_{2*}}\right)^2.
\end{aligned}$$

[119] By continuity, we define $0\ln(0) = 0$.

Hence, the condition CND^* that maximizes (35) is the one that splits the relevant training set objects into two subsets, such that the classification in these two subsets would be "as clear-cut as possible".

8.2 Pruning of a classification tree

An efficient strategy of building a classification tree is to grow a "large" tree $T_{\max}$ and then reduce the size of $T_{\max}$ using the process called **pruning**. The basic idea is to define a **pruning measure** of any subtree T of $T_{\max}$ as follows:

$$R_\alpha(T) = R^{err}(T) + \alpha \times \#T,$$

where $R^{err}(T)$ is the re-substitution estimate of the misclassification rate of T, the symbol $\#T$ denotes the number of terminal nodes of T, which serves as a measure of the size of the tree, and $\alpha \geq 0$ is the so-called **complexity parameter**.

For each α, there is one or more "optimal" subtrees of $T_{\max}$ that minimize $R_\alpha(T)$. For $\alpha = 0$, the optimal tree is $T_0 := T_{\max}$ itself and for a very large α, the optimal tree is T_M containing only the root node. By continuously increasing the values of α from 0[120], we can construct a sequence of optimizing trees $T_0, T_1, \ldots, T_M$ such that T_i is a subtree of T_{i-1} for all $i = 1, \ldots, M$ (such a sequence of trees is called **nested**). From the set $T_0, T_1, \ldots, T_M$ we choose the final tree using, for instance, some form of cross-validation.

8.3 Bagging and classification forests

Ensemble methods[121] combine several individual classifiers into an aggregate classifier. The most important classes of such methods are **bagging**[122] and **boosting**. In this subsection we will briefly discuss the general method of bagging and a closely related technique of **random forests**, specialized to combine and improve the performance of tree classifiers.

The idea of bagging is as follows. We will construct B[123] classifiers $\mathcal{C}_1, \ldots, \mathcal{C}_B$, based on randomly selected subsets[124] $(\mathbf{X}_1, \mathbf{c}_1), \ldots, (\mathbf{X}_B, \mathbf{c}_B)$ of the full training dataset $(\mathbf{X}, \mathbf{c})$. If $\boldsymbol{x}$ is the feature vector of a new object to be classified, we classify $\boldsymbol{x}$ by each of the B classifiers, and base the resulting

[120] In a process that requires many technical details that we skip in this text.
[121] Sometimes a fancy term of "committee machines" is used.
[122] The word is an abbreviation of the expression "bootstrap aggregating".
[123] B is typically chosen to be a few thousands.
[124] We use sampling with replacement.

"ensemble" classification of $\boldsymbol{x}$ on the majority vote. That is, the classification of the object $\boldsymbol{x}$ is the class that occurs most frequently in the set of classes predicted by individual classifiers $\mathcal{C}_1, \ldots, \mathcal{C}_B$. Bagging improves the performance in particular if the method of construction of the underlying classifiers is "unstable", i.e., if small perturbations of the training set result in highly variable classifiers. Bagging is typically, but not necessarily, applied to classification trees.

It turns out that the performance of bagging – if applied to classification trees – can be improved by a surprising additional trick: Each split of each bootstrap classification trees $(\mathbf{X}_1, \mathbf{c}_1), \ldots, (\mathbf{X}_B, \mathbf{c}_B)$ is based only on a random subset of all explanatory variables (predictors, features, etc). Usually about $\sqrt{p}$ of explanatory variables are randomly selected as candidates for each split. It turns out that this trick makes the individual trees "less correlated", or more "diverse", and generally improves the overall performance of the classification. The resulting classifier is called a random forest.

By using bagging or random forests, we loose the simplicity and interpretability of classification trees, but often achieve a classifier with much better performance.

8.4 Boosting

Boosting refers to the methods which combine "weak" classifiers $\mathcal{C}_1, \ldots, \mathcal{C}_B$ into one "strong" classifier. These weak classifiers are often, albeit not always, classification trees. In contrast to bagging, the construction is sequential, adding one weak classifier at a time in an optimal way.[125]

Here we will explain the classical version of the method **Adaboost**.[126] Suppose that our underlying weak classifiers $\mathcal{C}_1, \ldots, \mathcal{C}_B$ are binary, and, for any $j = 1, \ldots, B$, the classifier $\mathcal{C}_j$ is determined by the discrimination function $k_j : \mathbb{R}^p \to \{-1, +1\}$.[127] We will also assume that every weak classifier

[125] Other differences between boosting and bagging include: 1) the same classifier can be added more than once; 2) the resulting classifier is not based on the majority vote of the weak classifiers, but rather on a special linear combination of the discrimination functions of the weak classifiers. This will be clarified next.

[126] From "adaptive boosting".

[127] That is, we will denote the classes by the labels $-1, +1$. This provides a more convenient notation in mathematical formulas, similarly to the support vector machines described in the next section. Note also that the classifiers $\mathcal{C}_1, \ldots, \mathcal{C}_B$ are "hard"; they do not provide nuanced degrees of belief about the class of a classified object.

classifies at least one object incorrectly and at least one object correctly.[128] We will sequentially construct a sequence $\mathcal{S}_1, \mathcal{S}_2, \ldots$ of binary classifiers with increasing strength,[129] which will be represented by discrimination functions $l_1, l_2, \ldots$.[130]

The initial classifier $\mathcal{S}_1$ is given by $l_s = k_s$ for some $s \in \{1, \ldots, B\}$, i.e., $\mathcal{S}_1$ is just one of the weak classifiers. Now, suppose that, for a fixed $t \geq 2$, we have already constructed $\mathcal{S}_{t-1}$ determined by its discrimination function l_{t-1}. The main idea of Adaboost is to construct the next classifier $\mathcal{S}_t$ such that its discrimination function is

$$l_t(\boldsymbol{x}) = l_{t-1}(\boldsymbol{x}) + \alpha k(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^p,$$

and both $\alpha \in \mathbb{R}$ and $k \in \{k_1, \ldots, k_B\}$ are selected to minimize some "error measure" of $\mathcal{S}_t$. A simple and practically efficient error measure of $\mathcal{S}_t$ is

$$E_t := \sum_{i=1}^{n} \exp(-y_i l_t(\boldsymbol{x}_i)). \tag{36}$$

In (36), y_i is the correct classification of the training object i. Note that if $\mathcal{S}_t$ correctly classifies i, then the contribution of $\exp(-y_i l_t(\boldsymbol{x}_i))$ to E_t is negligible with $|l_t(\boldsymbol{x}_i)| \to \infty$, and if $\mathcal{S}_t$ incorrectly classifies i, the contribution of $\exp(-y_i l_t(\boldsymbol{x}_i))$ to E_t is enormous if $|l_t(\boldsymbol{x}_i)| \to \infty$.

We will express E_t in a form which makes the dependence of the error on α and k more explicit. First, define the weights

$$w_i^{(t)} = \exp(-y_i l_{t-1}(\boldsymbol{x}_i))$$

for all objects i. The weights are strictly positive numbers, known at the stage of construction of $\mathcal{S}_t$.[131] Next, for any $k \in \{k_1, \ldots, k_B\}$ let $Q_{yes}^{(k)}$ be the set off all objects i for which $k(\boldsymbol{x}_i) = y_i$ and let $Q_{no}^{(k)}$ be the set off all objects

[128] Otherwise, the weak classifier would not be weak, or it would be completely useless. Also, these two assumptions lead to a smoother mathematical derivation of Adaboost.

[129] We will repeat this process a given number of iterations or until some other stopping criterion is satisfied.

[130] In the sense that the sign of $l_t(\boldsymbol{x})$ determines the class of the object characterized by $\boldsymbol{x} \in \mathbb{R}^p$, assigned to the object by $\mathcal{S}_t$.

[131] They are known because at this stage the discriminant function l_{t-1} is already known.

i for which $k(\boldsymbol{x}_i) \neq y_i$.[132] Using this notation we have

$$\begin{aligned} E_t &= \sum_{i=1}^{n} \exp(-y_i l_{t-1}(\boldsymbol{x}_i)) \exp(-y_i \alpha k(\boldsymbol{x}_i)) \\ &= \sum_{i=1}^{n} w_i^{(t)} \exp(-y_i \alpha k(\boldsymbol{x}_i)) \\ &= \sum_{i \in Q_{yes}^{(k)}} w_i^{(t)} \exp(-y_i \alpha k(\boldsymbol{x}_i)) + \sum_{i \in Q_{no}^{(k)}} w_i^{(t)} \exp(-y_i \alpha k(\boldsymbol{x}_i)). \end{aligned}$$

At this point, we notice that $y_i k(\boldsymbol{x}_i) = 1$ for all $i \in Q_{yes}^{(k)}$, and $y_i k(\boldsymbol{x}_i) = -1$ for all $i \in Q_{no}^{(k)}$, hence

$$E_t = \left\{ \sum_{i \in Q_{yes}^{(k)}} w_i^{(t)} \right\} e^{-\alpha} + \left\{ \sum_{i \in Q_{no}^{(k)}} w_i^{(t)} \right\} e^{\alpha}. \tag{37}$$

Now, we need to minimize (37) with respect to the choice of $\alpha \in \mathbb{R}$ and $k \in \{k_1, \ldots, k_B\}$. For simplicity, denote the sums of the weights in (37) as

$$w_{yes}^{(k)} := \sum_{i \in Q_{yes}^{(k)}} w_i^{(t)} \text{ and } w_{no}^{(k)} := \sum_{i \in Q_{no}^{(k)}} w_i^{(t)}.$$

An important observation is that for any fixed k the function $\alpha \to w_{yes}^{(k)} e^{-\alpha} + w_{no}^{(k)} e^{\alpha}$ on the right-hand side of (37) is smooth and strictly convex on $\mathbb{R}$, converging to ∞ for $\alpha \to \infty$ as well as for $\alpha \to -\infty$;[133] such a function has a unique minimum on $\mathbb{R}$ given by[134]

$$\alpha^{(k)} = \frac{1}{2}\left(\ln\left(w_{yes}^{(k)}\right) - \ln\left(w_{no}^{(k)}\right) \right).$$

We can therefore compute all values $E_t = E_t(k, \alpha^{(k)})$, $k = k_1, \ldots, k_B$,[135] and select the smallest one. The arguments of the smallest $E_t(k, \alpha^{(k)})$ provide the optimal weak classifier k^* and the corresponding optimal coefficient α^*.

[132] Note that for any k both $Q_{yes}^{(k)}$ and $Q_{no}^{(k)}$ are non-empty, which is a consequence of the technical assumption from the introduction of this subsection: each weak classifier classifies at least one object correctly and at least one object incorrectly.

[133] Strict convexity follows from the positivity of the coefficients $w_{yes}^{(k)}$, $w_{no}^{(k)}$ and from the strict convexity of the functions $\alpha \to e^{-\alpha}$, $\alpha \to e^{\alpha}$. The limit properties also follow from the positivity of both $w_{yes}^{(k)}$ and $w_{no}^{(k)}$.

[134] It is a simple exercise in calculus.

[135] Let us emphasize that this is only B simple to compute values.

We remark that typically we have $\alpha^{(k^*)} > 0$, which is equivalent to $w_{yes}^{(k^*)} > w_{no}^{(k^*)}$; that is because the classifiers k, albeit weak, still usually classify the majority of objects correctly. In the case $\alpha^{(k^*)} > 0$ we can characterize k^* more explicitly. To this end, note that (37) with $k = k^*$ and $\alpha = \alpha^{(k^*)}$ can be rewritten to the form

$$E_t = \sum_{i=1}^{n} w_i^{(t)} e^{-\alpha^{(k^*)}} + w_{no}^{(k^*)} \left(e^{\alpha^{(k^*)}} - e^{-\alpha^{(k^*)}} \right). \tag{38}$$

Now, $\alpha^{(k^*)} > 0$ implies that $e^{\alpha^{(k^*)}} - e^{-\alpha^{(k^*)}} > 0$, therefore k^* must be a weak classifier which provides the minimum value of $w_{no}^{(k)}$.[136] In words, the optimal weak classifier k^* to be incorporated into $\mathcal{S}_{t-1}$ is typically the one which minimizes the sum of the weights $w_i^{(t)}$ of all points i which it classifies incorrectly. Note that the weights $w_1^{(t)}, \ldots, w_n^{(t)}$ represent a measure of how "difficult" the objects $i = 1, \ldots, n$ are for the classifier $\mathcal{S}_{t-1}$. Hence, the weak classifier k^* to be added is, roughly put, the one which classifies the most problematic objects correctly, and the better k^* is with respect to this measure of difficulty, the larger is the coefficient $\alpha^{(k^*)}$.

The classifier based on AdaBoost usually performs better than any of the individual weak classifiers, especially when the weak classifiers are only slightly better than the toss of a coin, or when the weak classifiers "complement" each other.

9 Support vector machines

Support vector machines (SVMs) are binary[137] based on a series of ingenious mathematical ideas as explained in the following subsections.

We will consider a training sample with feature vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$ and corresponding classifications $c_1, \ldots, c_n \in \{-1, +1\}$,[138] such none of the

[136] We can see this by contradiction. Let there be some k^x, such that $w_{no}^{(k^x)} < w_{no}^{(k^*)}$. Then the pair $k = k^x$, $\alpha = \alpha^{(k^*)}$ would provide a smaller value of E_t than the pair $k = k^*$, $\alpha = \alpha^{(k^*)}$, i.e., the pair $k = k^*$, $\alpha = \alpha^{(k^*)}$ is not the minimizer of E_t.

[137] Multi-class SVMs *do* exist, but the fundamental principle of the SVMs pertains to the binary classification. One general approach to applying a binary classification method to a problem with $K > 2$ classes is called **one-on-one**: we construct a collection of $\binom{K}{2}$ binary classifiers between all pairs of classes. A new observation $\boldsymbol{x}$ is classified by all of the binary classifiers from the collection. The classification of $\boldsymbol{x}$ is then chosen to be the one that occurs most frequently in the results of these binary classifications.

[138] In the theory of SVMs, the two classes are usually labelled -1 and $+1$ because it permits a more compact statement of mathematical formulas and results compared to the standard class notations $\{0, 1\}$ or $\{1, 2\}$.

feature sets $C_- := \{\boldsymbol{x}_i : c_i = -1\}$ and $C_+ := \{\boldsymbol{x}_i : c_i = +1\}$ is empty.

9.1 Linear support vector machines

Assume first that the objects included in the training set are **linearly separable**,[139] which means that the convex hulls of the feature sets C_- and C_+ are disjoint. An equivalent definition of linear separability is

$$\begin{aligned} c_i = -1 \quad &\Rightarrow \quad \beta_1^T \boldsymbol{x}_i + \beta_0 < 0, \text{ and} \\ c_i = +1 \quad &\Rightarrow \quad \beta_1^T \boldsymbol{x}_i + \beta_0 > 0 \end{aligned}$$

for all $i = 1, \ldots, n$ and some $\beta = (\beta_0, \beta_1^T)^T \in \mathbb{R}^{p+1}, \beta_1 \neq \mathbf{0}_p$, i.e., there exists a hyperplane

$$H_\beta^0 := \{\boldsymbol{x} \in \mathbb{R}^p : \beta_1^T \boldsymbol{x} + \beta_0 = 0\}$$

that "separates" C_- and C_+. Such a **separating hyperplane** is a natural boundary between the classification regions R_{-1} and R_{+1} of the linear classifier that we wish to construct.

Note, however, that in the linearly separable case there is an entire continuum of separating hyperplanes. The fundamental principle of the SVMs is to base the classification on the so-called **maximum-margin** separating hyperplane $H_{\beta^*}^0$. More precisely, define the **margin** of a hyperplane H_β^0 as

$$d_\beta := \min_{i=1,\ldots,n} \rho(H_\beta^0, \boldsymbol{x}_i),$$

where $\rho(H_\beta^0, \boldsymbol{x}_i)$ is the Euclidean distance between H_β^0 and $\boldsymbol{x}_i$.[140] The optimal β^* is then chosen to maximize d_β subject to the condition that H_β^0 is a separating hyperplane.[141] This fully defines the optimization problem that we need to solve, but the key question is whether we can solve it efficiently enough to be applied to practical problems. We will show that it is indeed the case.

[139] In applications, the objects of the training set are usually *not* linearly separable, but linearly separable case serves as a springboard to more practical (linearly non-separable and non-linear) versions of the SVMs.

[140] Margin can be defined in many ways to serve the purpose of introducing the SVMs. Here we define it simply as the distance of H_β^0 to the nearest feature vector.

[141] This "geometric" definition already provides some useful insights into the properties of the SVMs. For instance,we see that the SVM classifier can be robust with respect to even extreme outliers. Moreover, the separating hyperplane is unique and well defined, even if $n < p$. These favourable properties are also inherited by the more complex versions of the SVMs that we will study next.

The points $\boldsymbol{x}_i$ that are closest to $H^0_{\beta^*}$ are called **support vectors** and they play a key role in our optimization problem. Formally, $\boldsymbol{x}_{i^*}$ is a support point if and only if

$$\rho(H^0_{\beta^*}, \boldsymbol{x}_{i^*}) = \min_{i=1,\dots,n} \rho(H^0_{\beta^*}, \boldsymbol{x}_i) \ (= d_{\beta^*}).$$

For a fixed β^*, the distance between $\boldsymbol{x}$ and $H^0_{\beta^*}$ depends only on $|(\beta^*_1)^T\boldsymbol{x}+\beta^*_0|$, since[142]

$$\rho(H^0_{\beta^*}, \boldsymbol{x}) = \frac{|(\beta^*_1)^T\boldsymbol{x} + \beta^*_0|}{\|\beta^*_1\|}.$$

Because $H^0_{\beta^*}$ is a maximum-margin separating hyperplane, it resides exactly in the middle between the support vectors from the two opposite classes. Thus, there exists some common $\epsilon > 0$ such that

$$c_i = -1 \quad \Rightarrow \quad (\beta^*_1)^T\boldsymbol{x}_i + \beta^*_0 \leq -\epsilon \tag{39}$$
$$c_i = +1 \quad \Rightarrow \quad (\beta^*_1)^T\boldsymbol{x}_i + \beta^*_0 \geq \epsilon, \tag{40}$$

for all $i = 1, \dots, n$, and the equalities are attained for the support vectors. Consequently,

$$d_{\beta^*} = \min_{i=1,\dots,n} \frac{|(\beta^*)^T_1\boldsymbol{x}_i + \beta^*_0|}{\|\beta^*_1\|} = \frac{\epsilon}{\|\beta^*_1\|}. \tag{41}$$

Thus, to find the maximum-margin hyperplane, it is enough to find $\beta^* = (\beta^*_0, (\beta^*_1)^T)^T \in \mathbb{R}^{p+1}$ and $\epsilon > 0$ that solve the problem: maximize (41) subject to both (39) and (40).[143] However, if (β^*, ϵ) is a solution to this problem, then $(\beta^*/\epsilon, 1)$ is also a solution to the same problem,[144] therefore we can fix $\epsilon = 1$. Finally, note that maximizing $1/\|\beta^*_1\|$ is the same as minimizing $\|\beta^*_1\|^2$ and the pair (39) and (40) of inequalities for $\epsilon = 1$ can be compactly written as $c_i((\beta^*_1)^T\boldsymbol{x}_i + \beta^*_0) \geq 1$. Therefore, to find a maximum-margin separating hyperplane, it is enough to compute a solution β^* to the following problem of linearly constrained **quadratic programming**:

$$\begin{aligned} &\min_{\beta_0\in\mathbb{R},\beta_1\in\mathbb{R}^p} \|\beta_1\|^2 \\ &\text{s.t.} \ \ c_i(\beta_1^T\boldsymbol{x}_i + \beta_0) \geq 1, \ i = 1, \dots, n. \end{aligned} \tag{42}$$

[142] You may recall this formula from the class on analytic geometry in the high school.

[143] Note that the feasible set for β^* determined by (39) and (40) does not include the zero vector.

[144] This is the same as the observation that $H^0_{\beta^*}$, as a set of points, does not change if we divide β^* by the positive ϵ.

This problem is still not computationally trivial, but it is a convex optimization problem[145] which can be solved by robust and efficient solvers of mathematical programming. The linear classifier which we obtain by this method in the case of linearly separable classes is called the **maximum-margin classifier**.

If the class sets C_- and C_+ are ***not* linearly separable**, the constraints (42) do not admit any feasible solution.[146] In such a case, the idea of how to proceed is to accept violations of the constraints, which is sometimes called the **soft margin** solution. One specification of the soft margin solution is to quantify the constraints violation via new non-negative variables, the so-called **slack variables** $\xi_1, \dots, \xi_n$; that is, we replace (42) with $c_i(\beta_1^T \boldsymbol{x}_i + \beta_0) \geq 1 - \xi_i$, $i = 1, \dots, n$. However, the violation should be as small as possible, which can be forced by adding a **penalty term** to the objective function. The standard choice of the penalty term is $C \sum_{i=1}^n \xi_i$,[147] where $C > 0$ is a **tuning parameter** chosen by the user of the method. The resulting optimization problem is

$$\begin{aligned} &\min_{\beta_0 \in \mathbb{R}, \beta_1 \in \mathbb{R}^p, \xi_1 \geq 0, \dots, \xi_n \geq 0} \|\beta_1\|^2 + C \sum_{i=1}^n \xi_i \\ &\text{s.t. } c_i(\beta_1^T \boldsymbol{x}_i + \beta_0) \geq 1 - \xi_i,\ i = 1, \dots, n. \end{aligned} \tag{43}$$

This optimization problem has always a feasible solution and it is still a problem of linearly constrained quadratic programming, i.e., not that difficult to numerically solve even for relatively large p and n.[148] Now, the optimal β^* will not define a separating hyperplane (there is no such hyperplane in the linearly non-separable case), yet the hyperplane $H^0_{\beta^*}$ forms a reasonable boundary of the classification sets R_- and R_+.

9.2 Nonlinear support vector machines

The practical usability of the linear SVMs is limited to less complex data. Nevertheless, there exists a simple trick of utilizing a linear classification

[145] That is, we minimize a convex function over a non-empty convex set. We remark that the optimization of the parameters of a classifier is a highly non-convex problem for some alternative methods (such as the artificial neural networks).

[146] Note that the feasible set (42) is non-empty if and only if the objects of the training set are linearly separable.

[147] The penalty term should be as simple as possible; with an overcomplicated penalty term, we could render the resulting optimization problem hard to solve.

[148] However, what is usually *really* solved is a dual problem which we formulate in the next subsection.

method for constructing a non-linear classifier. The idea is to transform the features via a non-linear transformation[149] ϕ into a new feature space[150], and construct a linear classifier in that new feature space.[151]

For instance, suppose that we replace the original feature vectors $\boldsymbol{x}_i$ with new feature vectors $\phi(\boldsymbol{x}_i) := (\boldsymbol{x}_i^T, \|\boldsymbol{x}_i\|^2)^T$. Now, if $\tilde{R}_-, \tilde{R}_+$ is the classification-set representation of a linear classifier[152] in the new feature space $\mathbb{R}^{p+1}$ then the corresponding classifications sets in the original space, i.e., $R_- = \{\boldsymbol{x} \in \mathbb{R}^p : \phi(\boldsymbol{x}) \in \tilde{R}_-\}$ and $R_+ = \{\boldsymbol{x} \in \mathbb{R}^p : \phi(\boldsymbol{x}) \in \tilde{R}_+\}$, are separated by a non-linear manifold. That is, the corresponding classifier is non-linear.[153]

This idea can be used to transform the linear SVMs to non-linear SVMs. Using the **duality theory** of convex optimization[154] it is possible to characterize the solution of (43) via an alternative problem of quadratic optimization as follows.

Theorem 9.1. *Let $C > 0$, let $\beta^* = (\beta_0^*, (\beta_1^*)^T)^T$ be the solution of (43) and let $\mathbf{H}$ be an $n \times n$ matrix with elements $\mathbf{H}_{ij} = c_i c_j \boldsymbol{x}_i^T \boldsymbol{x}_j$, $i, j = 1, \ldots, n$. Let $\alpha^* \in \mathbf{R}^n$ be the solution to the following optimization problem*

$$\begin{aligned} &\max_{\alpha \in \mathbb{R}^n} \sum_{i=1}^n \alpha_i - \frac{1}{2}\alpha^T \mathbf{H} \alpha \\ &s.t. \sum_{i=1}^n c_i \alpha_i = 0,\ 0 \le \alpha_i \le C,\ i = 1, \ldots, n. \end{aligned} \tag{44}$$

Then

$$\begin{aligned} \beta_1^* &= \sum_{i=1}^n c_i \alpha_i^* \boldsymbol{x}_i, \text{ and} \\ \beta_0^* &= c_j - \boldsymbol{x}_j^T \beta_1^* \text{ for any } j \text{ such that } 0 < \alpha_j^* < C. \end{aligned}$$

[149] This transformation is sometimes called a **feature map** in the machine learning literature.

[150] For the applications to the SVMs, the new feature space must be a Hilbert space, i.e., a linear vector space with a scalar product $\langle \cdot, \cdot \rangle$. Usually, the dimension of the new feature space is larger than p, often infinite.

[151] This trick can be applied to many linear classifiers; however, the SVM classifier is constructed such that it admits an efficient computation, even if the new feature space is huge.

[152] That is, $\tilde{R}_-$ and $\tilde{R}_+$ are complementary half-spaces.

[153] Which geometric object corresponds to the boundary between R_- and R_+?

[154] The theory of duality is challenging to the extent that it is better suited for theoretical lectures on convex optimization. We will thus skip the proof of the following theorem.

Note that the matrix $\mathbf{H}$ from Theorem 9.1 is non-negative definite[155] therefore the problem (44) is again one of linearly constrained quadratic programming.[156] However, the really *key* observation is that the problem (44) is based only on the mutual scalar products $\boldsymbol{x}_i^T\boldsymbol{x}_j$ between the feature vectors, in the sense that even if we do not know the actual feature vectors, but we do know the mutual scalar products of these vectors, we can solve (44). Specifically, the problem itself does not depend on the dimension p of the features.

To compute the parameter β^* of the hyperplane forming the decision boundary between R_- and R_+ requires that we know the feature vectors $\boldsymbol{x}_i$ themselves. However, for all practical purposes we do not in fact need to know β^* itself; we only need to be able to classify new feature vectors $\boldsymbol{x} \in \mathbb{R}^p$ into R_- or R_+. That is, we need to be able to decide whether the value $f(\boldsymbol{x}) = (\beta_1^*)^T\boldsymbol{x} + \beta_0^*$ of the discrimination function corresponding to the separating hyperplane $H^0_{\beta^*}$ is negative or positive. But (for any j such that $0 < \alpha_j < C$) we have

$$f(\boldsymbol{x}) = (\beta_1^*)^T\boldsymbol{x} + \beta_0^* = \sum_{i=1}^n c_i\alpha_i^*\boldsymbol{x}_i^T\boldsymbol{x} + c_j - \sum_{i=1}^n c_i\alpha_i^*\boldsymbol{x}_j^T\boldsymbol{x}_i.$$

Hence, the value of $f(\boldsymbol{x})$ is *also* fully determined by the scalar products $\boldsymbol{x}_i^T\boldsymbol{x}$ and $\boldsymbol{x}_j^T\boldsymbol{x}_i$ on the feature space.

Now, let us recall that one method of performing non-linear classification is to first construct an auxiliary linear classifier based on $\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)$ for some non-linear feature map ϕ, and then base the classification of $\boldsymbol{x}$ on how $\phi(\boldsymbol{x})$ would be classified via this linear classifier. Based on Theorem 9.1 and the explanation above we see that in order to use the linear SVM classifier in the role of this auxiliary linear classifier, we only need to be able to compute the scalar products $\langle\phi(\boldsymbol{x}), \phi(\boldsymbol{y})\rangle$ for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$. We do not need to explicitly compute the vectors $\phi(\boldsymbol{x})$ at all. We do not even need to know explicitly the new feature space (the codomain of ϕ).

The usual method of non-linear SVM classification is thus to use just a **kernel function**[157] $K(\boldsymbol{x}, \boldsymbol{y})$ equal to the scalar product $\langle\phi(\boldsymbol{x}), \phi(\boldsymbol{y})\rangle$ of a

[155] $\mathbf{H}$ is a matrix of all scalar products of the set of vectors $c_1\boldsymbol{x}_1, \ldots, c_n\boldsymbol{x}_n$, that is, it is the so-called Gram matrix.

[156] Note also the strict inequalities in the formula for β_0^*. In fact, the vectors x_j such that $0 < \alpha_j^*$ are called "support vectors" for this "soft margin" solution, although the geometric meaning is less clear than in the linearly separable situation.

[157] Some literature uses the term "support vector machine" only if the classifier uses the kernels. The less advanced versions are then called just a "support vector classifier".

function ϕ.[158] The feature map ϕ is typically not explicitly stated (nor is the Hilbert space codomain of ϕ).[159] The standard choices of such kernel functions are the so-called **Gaussian kernel**[160]

$$K(\boldsymbol{x},\boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right)$$

and the **polynomial kernel**

$$K(\boldsymbol{x},\boldsymbol{y}) = \left(\boldsymbol{x}^T\boldsymbol{y}+c\right)^d,$$

but there are many other types of kernels. The values $\sigma > 0, c \geq 0, d \in \mathbb{N}$ represent tuning parameters and the software that computes the non-linear SVMs usually allows the user to choose them, similarly to the parameter C.[161]

Note that despite the fact that we implicitly work with possibly infinite-dimensional transformed features $\phi(\boldsymbol{x})$, the size of the quadratic problem from theorem 9.1 is only given by the number n of objects in the training set. However, to construct the problem we need to pre-compute the elements $\mathbf{H}_{ij} = c_i c_j K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ which can take much time if the dimension p is large; therefore, the computational complexity is not completely independent of p. For the evaluation of the discriminant function $f(\boldsymbol{x})$, the additional computation of the kernel values $K(\boldsymbol{x}_i,\boldsymbol{x})$ is required. Nevertheless, the number of non-zero α_i^*'s tends to be much smaller than n in practical problems, and, to classify $\boldsymbol{x}$, we need to compute the correspondingly smaller number of values $K(\boldsymbol{x}_i,\boldsymbol{x})$.

To summarize, SVMs are classification methods that allow for complex, non-linear separation of classes, yet they can be efficiently computed via methods of quadratic convex optimization.

[158] In fact the features $\boldsymbol{x}$ do not even need to be real-valued vectors (they can be for instance text strings) provided that we have a method (more specifically, a kernel) that produces scalar products between pairs of objects.

[159] Of course, ϕ and the Hilbert space do exist; we just do not need to explicitly know them for practical purposes.

[160] Gaussian kernel is sometimes called the **radial basis kernel**.

[161] The choice of these so-called **hyper-parameters** is usually based on the grid-search exploration of the empirical performance of the resulting classifiers. Observe also that if we choose $c = 0$ and $d = 1$ in the polynomial kernel, we have $K(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}^T\boldsymbol{y}$. In this case the mapping ϕ is in fact linear - the identity.