Durham University

High Dimensional Statistics IV

(MATH4287)

Lecture notes, 2024–25

Reza Drikvandi

Durham University

Contents

CHAPTER 1

Introduction to high dimensional statistics

1.1. High dimensional data

We are in the era of "big data" where the rapid development of technologies, data storage resources and computing resources give rise to the production, storage and processing of an exponentially growing volume of data. A major characteristic of such modern data is that they can record simultaneously thousands or millions of variables on each object or individual. Such data are said to be high dimensional. Having information on a large number of variables helps us better understand a given phenomenon of interest. For example, wide-scale data enable us to better understand the regulation mechanisms of living organisms, to create new therapies, to monitor climate and biodiversity changes, to optimise the resources in the industry and in administrations, and to personalise the marketing for each individual consumer, etc.

High dimensional data is an aspect of big data, which are rapidly growing in many different fields due to the data revolution. Technically, the high dimensional data refers to the data in which the number of variables, say p, is much larger than the number of observations, say n, (or simply data with $n \ll p$). A popular example is genetic data which can contain tens of thousands of genes but may only include few hundreds of patients with a particular disease (e.g., cancer). High dimensional data have many different applications, for example, in microarray gene expression studies, fMRI data analysis, large-scale healthcare analytics, text/image analysis, natural language processing, finance and astronomy, to name but a few.

In this chapter, we provide an introduction to high dimensional statistics and data analysis. Despite the blessing and informativeness of high dimensional data, analysing such data is challenging from both methodological and computational points of view. We will see that classical statistical and mathematical methods often fail to work in high dimensional settings. Separating the signal from the noise is difficult in high

dimensional situations. This phenomenon, which we discuss in detail in the next section, is known as the *curse of dimensionality*.

1.2. Curse of dimensionality

In this section, we show that there are multiple impacts of high dimensionality on statistics. First, the accumulation of small fluctuations in many different directions can produce a large global fluctuation. Second, the concept of locality is often lost in high dimensions. Third, as the dimension increases, the typical distances between observations massively increase and often diverge with the dimension. Fourth, unlike in low dimensions, the tails of high dimensional distributions would not be thin. Fifth, numerical computations and optimisations in high dimensional spaces can be overly intensive. There are other consequences, of course.

1.2.1. Fluctuations cumulate

Suppose that we want to evaluate some function $F(\theta_1)$ of some scalar value $\theta_1 \in \mathbb{R}$. Assume that we only have access to a noisy observation of θ_1, denoted by $X_1 = \theta_1 + \varepsilon_1$, with $\mathrm{E}(\varepsilon_1) = 0$ and $\mathrm{Var}(\varepsilon_1) = \sigma^2$. If the function F is 1-Lipschitz (see Appendix A), then the mean square error satisfies

$$\mathrm{E}\big(\big|F(X_1) - F(\theta_1)\big|^2\big) \leq \mathrm{E}\big(\big|X_1 - \theta_1\big|^2\big) \leq \mathrm{E}\big(\big|\varepsilon_1\big|^2\big) = \mathrm{E}\big(\varepsilon_1^2\big) = \sigma^2.$$

Hence, if the variance σ^2 of the noise is small, then this mean square error is small too.

Suppose now that we need to evaluate a multivariate function $F(\theta_1, \ldots, \theta_p)$ from noisy observations $X_j = \theta_j + \varepsilon_j$ of the θ_j. Assume that the noise variables $\varepsilon_1, \ldots, \varepsilon_p$ all have mean 0 and variance σ^2. If as before F is 1-Lipschitz, we have

$$\mathrm{E}\big(\big\|F(X_1, \ldots, X_p) - F(\theta_1, \ldots, \theta_p)\big\|_2^2\big) \leq \mathrm{E}\big(\big\|\varepsilon_1, \ldots, \varepsilon_p\big\|_2^2\big) = \sum_{j=1}^{p} \mathrm{E}(\varepsilon_j^2) = p\sigma^2,$$

which implies the error upper bound scales with p. Furthermore, if function F fulfils $\big\|F(\boldsymbol{\theta} + \boldsymbol{h}) - F(\boldsymbol{\theta})\big\|_2^2 \geq c\big\|\boldsymbol{h}\big\|_2^2$ for some $c > 0$ (just take $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ and $\boldsymbol{h} = (\varepsilon_1, \ldots, \varepsilon_p)$), then the mean square error $\mathrm{E}\big(\big\|F(X_1, \ldots, X_p) - F(\theta_1, \ldots, \theta_p)\big\|_2^2\big)$ scales like $p\sigma^2$. Thus, this error can be very large in high dimensional settings when p gets very large, even if

σ^2 is small. A central example where such a situation happens is the linear regression model with high dimensional covariates, which we will study in Chapter 2.

1.2.2. Loss of locality in high dimensions

Consider a situation where we want to explain a response variable $Y \in \mathbb{R}$ by p variables or covariates $X^{(1)},\dots,X^{(p)} \in [0,1]$. Assume, for example, that each variable $X^{(k)}$ follows a uniform distribution on $[0,1]$. If these variables are independent, then the p-dimensional vector $\boldsymbol{X} = (X^{(1)},\dots,X^{(p)}) \in [0,1]^p$ follows a uniform distribution on the hypercube $[0,1]^p$. Suppose our observed data consist of n independent and identically distributed (i.i.d.) observations $(Y_i, \boldsymbol{X}_i)_{i=1,\dots,n}$ of the variables Y_i and $\boldsymbol{X}_i = (X_i^{(1)},\dots,X_i^{(p)})$. Suppose that we model the data with the classical regression equation

$$Y_i = f(\boldsymbol{X}_i) + \varepsilon_i, \quad i = 1,\dots,n,$$

with $f : [0,1]^p \to \mathbb{R}$ and $\varepsilon_1,\dots,\varepsilon_n$ independent with zero mean.

Assuming that the function f is smooth, it is natural to estimate $f(\boldsymbol{x})$, for a point $\boldsymbol{x}$, by some average of the Y_i associated with the $\boldsymbol{X}_i$ in the vicinity of $\boldsymbol{x}$. The most simple version of this idea is the k-Nearest Neighbours estimator, where $f(\boldsymbol{x})$ is estimated by the mean of the Y_i associated with the k points $\boldsymbol{X}_i$, which are the nearest from $\boldsymbol{x}$. Some more sophisticated versions of this idea use a weighted average of Y_i with weights that are a decreasing function of the Euclidean distance $\|\boldsymbol{X}_i - \boldsymbol{x}\|_2^1$ (like kernel smoothing). The basic idea in all cases is to use a local average of the data. This idea makes perfect sense in standard low dimensional settings, as illustrated in Figure 1.1 taken from Giraud [10].

Unfortunately, when the dimension p increases, the notion of "nearest points" vanishes. This phenomenon is illustrated in Figure 1.2, where we have plotted the histograms of the distribution of the pairwise-distances $\{\|\boldsymbol{X}_i - \boldsymbol{X}_j\|_2^1 : 1 \le i < j \le n\}$ (based on the Euclidean distance) for $n = 100$ and dimensions $p = 2, 10, 100, 1000$. From Figure 1.2, we observe that when the dimension p increases

- the minimal distance between two points increases, and
- all the points are at a similar distance from the others, so the notion of nearest points vanishes.

FIGURE 1.1. Canadian high school graduate earnings from the 1971 Canadian Census Public Use Tapes (Giraud [10]). The black dots: records. The grey line: 25-Nearest Neighbours estimator. The dashed line: local averaging by kernel smoothing.

So if the dimension of observations p increases while the number of observations n remains fixed, the observations $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n$ get rapidly very isolated and local methods cannot work. In fact, any estimator based on local averaging would fail with such high dimensional data when n remains much smaller than p. This is explained with the maths details in the next subsection.

1.2.3. Distance divergence

We here investigate the distances between high dimensional observations containing p variables $X^{(1)},\ldots,X^{(p)} \in [0,1]$ when the dimension p grows. Similar to the previous situation, assume that each variable $X^{(k)}$ follows a uniform distribution on the interval $[0,1]$. Suppose our data consist of n independent observations on each of these p variables where we write the i-th observation as $\boldsymbol{X}_i = (X_i^{(1)},\ldots,X_i^{(p)}) \in [0,1]^p$, $i = 1,\ldots,n$. Again if the p variables are independent, then the p-dimensional vector X_i follows a uniform distribution on the hypercube $[0,1]^p$. Let us focus on the pairwise distances

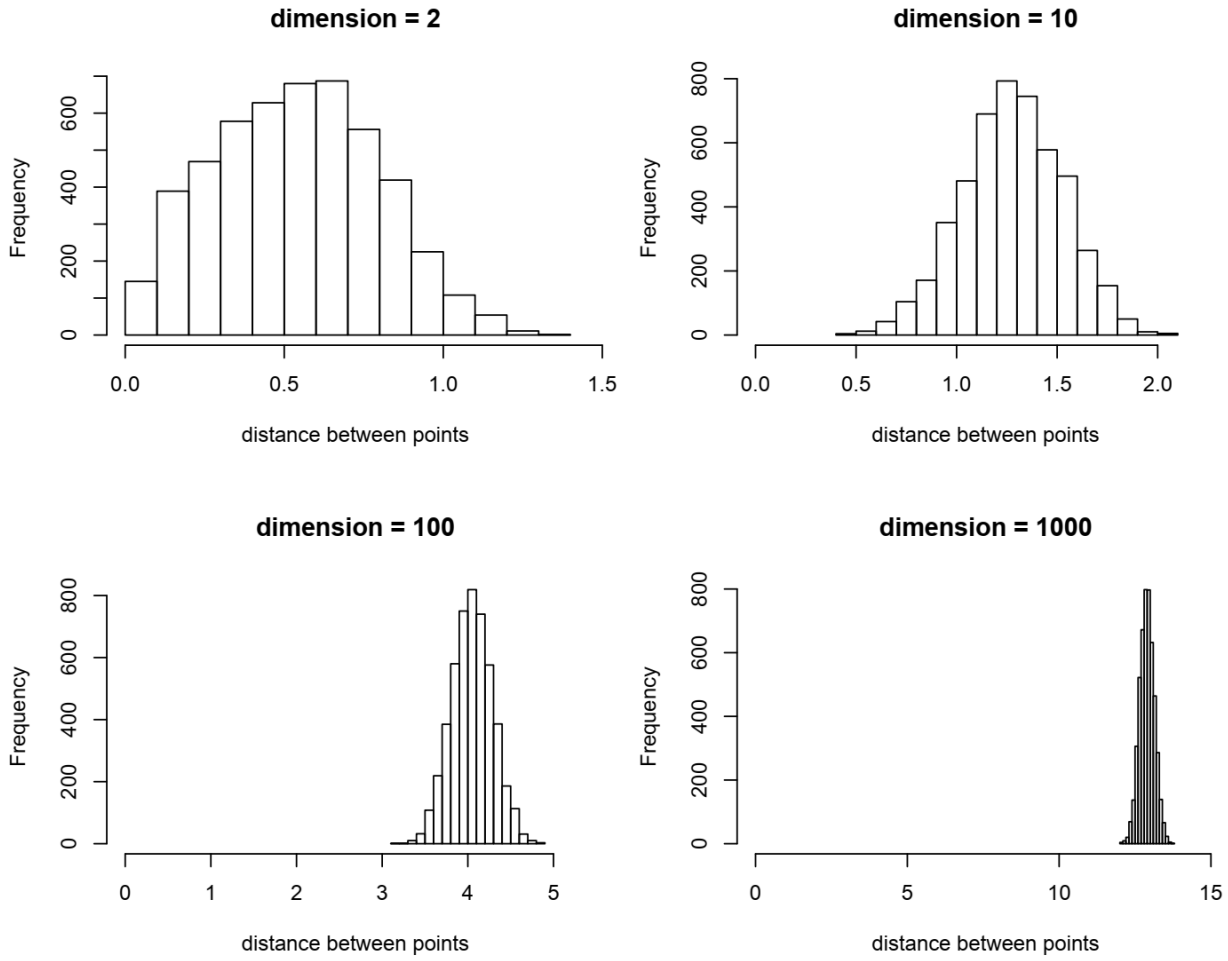


FIGURE 1.2. Histograms of the pairwise distances between $n = 100$ points sampled uniformly in the hypercube $[0,1]^p$ for $p = 2, 10, 100, 1000$.

$\{\|\boldsymbol{X}_i - \boldsymbol{X}_j\|_2^1 : 1 \le i < j \le n\}$ based on the Euclidean distance. Then, we obtain

$$\mathrm{E}\big(\|\boldsymbol{X}_i - \boldsymbol{X}_j\|_2^2\big) = \mathrm{E}\big(\sum_{k=1}^{p} (X_i^{(k)} - X_j^{(k)})^2\big) = \sum_{k=1}^{p} \mathrm{E}\big((X_i^{(k)} - X_j^{(k)})^2\big) = p\mathrm{E}\big((U - U')^2\big) = p/6,$$

where U and U' denote two independent random variables with uniform distribution on $[0,1]$. Thus, the typical square distance between two points sampled uniformly in $[0,1]^p$ diverges linearly with p.

Question 1. Show that the difference of two independent uniform random variables on the interval $[0,1]$ has the standard triangular distribution, that is, if $Z = U - U'$

then the probability density function of Z is given by (see `Problem Sheets`)

$$f_Z(z) = \begin{cases} z+1 & -1 < z < 0 \\ 1-z & 0 \le z < 1, \end{cases}$$

and hence $\mathrm{E}(Z^{2k}) = 2/(2k+1) - 2/(2k+2)$ for $k = 1, 2, \ldots$.

We similarly find that the standard deviation, denoted here by sd, satisfies

$$\begin{aligned} \mathrm{sd}\big(\|\boldsymbol{X}_i - \boldsymbol{X}_j\|_2^2\big) &= \sqrt{\mathrm{Var}\big(\|\boldsymbol{X}_i - \boldsymbol{X}_j\|_2^2\big)} \\ &= \sqrt{\mathrm{Var}\big(\sum_{k=1}^{p} (X_i^{(k)} - X_j^{(k)})^2\big)} \\ &= \sqrt{\sum_{k=1}^{p} \mathrm{Var}\big((X_i^{(k)} - X_j^{(k)})^2\big)} \\ &= \sqrt{p\mathrm{Var}\big((U-U')^2\big)} \approx 0.2\sqrt{p}. \end{aligned}$$

We can see that the standard deviation of the square distance between two points grows slower with a smaller rate of $\sqrt{p}$. The scaled deviation defined as

$$\frac{\mathrm{sd}\big(\|\boldsymbol{X}_i - \boldsymbol{X}_j\|_2^2\big)}{\mathrm{E}\big(\|\boldsymbol{X}_i - \boldsymbol{X}_j\|_2^2\big)}$$

then shrinks like $p^{-1/2}$.

To bypass this divergence issue in high dimensions, one solution is to define a modified Euclidean distance using the scaling factor $p^{-1/2}$ as $p^{-1/2}\|\boldsymbol{X}_i - \boldsymbol{X}_j\|_2^1$ which guaranties the convergence in high dimensions as $p \to \infty$ (see `Problem Sheets`). This modification was suggested by Hall et al. [11].

1.2.4. Tails of high dimensional distributions

Gaussian (normal) distributions are known to have very thin tails. Actually, the density of a standard multivariate Gaussian distribution $N(\boldsymbol{0}, \boldsymbol{I}_p)$ (here $\boldsymbol{I}_p$ is the $p \times p$ identity matrix) in $\mathbb{R}^p$ given by

$$f_p(\boldsymbol{x}) = (2\pi)^{-p/2} \exp\big(-\|\boldsymbol{x}\|_2^2/2\big)$$

decreases exponentially fast with the square norm of $\boldsymbol{x}$. However, when p is large, most of the mass of the standard Gaussian distribution lies in its tails! To see this, first note that the maximal value of $f_p(\boldsymbol{x})$ is $f_p(\mathbf{0}) = (2\pi)^{-p/2}$ which decreases exponentially fast toward 0 when p increases. Thus, the Gaussian distribution in high dimensions is much more flat than in dimension one or two. Let us compute the mass in its bell (central part of the distribution where density is the largest). Let $\delta > 0$ be a small positive real number and write

$$B_{p,\delta} = \{\boldsymbol{x} \in \mathbb{R}^p : f_p(\boldsymbol{x}) \geq \delta f_p(\mathbf{0})\} = \left\{\boldsymbol{x} \in \mathbb{R}^p : \|\boldsymbol{x}\|_2^2 \leq 2\log(\delta^{-1})\right\}$$

for the ball gathering all the points $\boldsymbol{x} \in \mathbb{R}^p$ such that the density $f_p(\boldsymbol{x})$ is larger or equal to δ times the density at $\mathbf{0}$. Using the Markov inequality (see Appendix A), we obtain that

$$\begin{aligned} P(\boldsymbol{X} \in B_{p,\delta}) = P\big(\|\boldsymbol{X}\|_2^2 \leq 2\log(\delta^{-1})\big) &= P\big(\exp\big(-\|\boldsymbol{X}\|_2^2/2\big) \geq \delta\big) \\ &\leq \frac{1}{\delta}\mathrm{E}\big(\exp\big(-\|\boldsymbol{X}\|_2^2/2\big)\big) \\ &= \frac{1}{\delta}\int_{\boldsymbol{x}\in\mathbb{R}^p} \exp\big(-\|\boldsymbol{x}\|_2^2\big)(2\pi)^{-p/2} d\boldsymbol{x} \\ &= \frac{1}{\delta 2^{p/2}}. \end{aligned} \tag{1.1}$$

This probability decreases as p increases, and in particular we have $P(\boldsymbol{X} \in B_{p,\delta}) \to 0$ as $p \to \infty$, so most of the mass of the standard Gaussian distribution is in the tail $\{\boldsymbol{x} \in \mathbb{R}^p : f_p(\boldsymbol{x}) < \delta f_p(\mathbf{0})\}$. Note that using (1.1), if we want to have $P(\boldsymbol{X} \in B_{p,\delta}) \geq 1/2$, we must choose $\delta \leq 2^{-p/2+1}$ which is exponentially small. Therefore, most of the mass of a Gaussian distribution is located in areas where the density is extremely small compared to its maximum value.

1.2.5. Computational complexity

Another difficulty with high dimensional data is that numerical computations can become very intensive and largely exceed the available computational and memory resources. For example, basic operations with $p \times p$ matrices (like multiplication, inversion etc) usually require at least p^α calculations with $\alpha \geq 2$. When p scales in thousands, iterating such operations can become quite intensive. Also, simultaneous regression methods such as forward selection and backward elimination are computationally very intensive when p is large as they require evaluating and computing

2^p submodels. Just imagine that with $p = 30$ we have $2^p = 1,073,741,824$ submodels to evaluate! Furthermore, iterative methods such as bootstrap and permutation tests can be very time-consuming in high dimensional settings.

1.3. High dimensional statistics

1.3.1. Circumventing the curse of dimensionality

As explained in the previous section, the high dimensionality of the data, which seems at first to be a blessing, is actually a major issue for statistical analysis. In light of the few examples described above, the situation may appear hopeless. Fortunately, high dimensional data are often much more low dimensional than they seem to be. Usually, they are not "uniformly" spread in $\mathbb{R}^p$, but rather concentrated around some low dimensional structures. These structures are due to the relatively small complexity of the systems producing the data. For example,

- biological data are the outcome of a "biological system" which is strongly regulated and whose regulation network has a relatively small complexity,
- marketing data strongly reflects some social structures in a population, and these structures are relatively simple,
- pixel intensities in an image are not purely random since there exist many geometrical structures in an image,
- technical data are the outcome of human technologies, whose complexity remains limited.

As another example of this in genomics, usually a small number of genes are truly associated with a heath condition such as a disease. So in many cases, the data have an intrinsic low complexity, and we can hope to extract useful information from them. When the low dimensional structures are known, we are back to some more classical "low dimensional statistics". However, the major issue with high dimensional data is that these structures are usually unknown. The main task will then be to identify at least approximately theses structures.

1.3.2. A paradigm shift

Classical statistics provides a very rich theory for analysing data with the following characteristics:

- a small number p of parameters,
- a large number n of observations.

Classical results carefully describe the asymptotic behaviour when n goes to infinity with p fixed, which makes sense in such a setting. However, as explained in the previous section, high dimensional data have very different characteristics:

- a huge number p of parameters,
- a sample size n, which is smaller than p or of the same size as p.

The asymptotic analysis with p being fixed and n goes to infinity does not make sense anymore in such high dimensional settings.

In order to provide a theory adapted to the 21st century data, two different points of view can be adopted. A first point of view is to investigate the properties of estimators in a setting where both n and p go to infinity, with $p \sim f(n)$ for some function f; for example, $f(n) = \alpha n$, or $f(n) = n^2$, or $f(n) = e^{\alpha n}$, etc. Such a point of view is definitely more suited to modern data than the classical point of view. Yet, it is sensitive to the choice of the function f. For example, asymptotic results for $f(n) = n^2$ and $f(n) = e^{\alpha n}$ can be very different. If $p = 1000$ and $n = 33$, are we in the setting $f(n) = n^2$ or $f(n) = e^{n/5}$?

An alternative point of view is to treat n and p as they are and provide a non-asymptotic analysis of estimators, which is valid for any value of n and p. Such an analysis avoids the above caveat of the asymptotic analysis. However, the main drawback is that non-asymptotic analyses are much more involved than asymptotic analyses. They usually require much more elaborate arguments in order to provide precise enough results.

CHAPTER 2

Dimension reduction and regularised regression

2.1. Introduction

Dimension reduction and variable selection are very useful in high dimensional data analysis. Their main objective often is to find variables that are the most relevant for statistical analysis including inference and prediction. In this chapter, we will see that this approach is particularly useful when the unknown underlying model has a sparse structure, that is, only a smaller number of variables are truly important and the other (many) variables are unimportant or less relevant. Recall the discussion on low dimensional structures in Section 1.3.1. The identification of those important variables will help reduce the noise and therefore improve the statistical analyses such as inference and prediction performance.

Regularisation is a popular technique in high dimensional statistics. It aims to regularise high dimensional problems by setting some conditions or constraints to tackle a high dimensional problem that cannot be solved using classical statistical methods. Some popular regularisation methods are the ridge regression with L_2-norm penalty and the lasso with L_1-norm penalty. In this chapter, we will see that, unlike ridge regression which cannot do variable selection with L_2-norm penalty as it keeps all variables in the fitted model, the lasso performs both variable selection and parameter estimation simultaneously using the L_1-norm penalty. This is very desirable in high dimensional situations as it is easier to interpret the results when the fitted model is sparse (i.e., has fewer variables). It also helps us to improve the prediction performance because having more variables often means higher uncertainty which could affect prediction results. We will study the high dimensional regularised regression both theoretically and computationally.

2.2. High dimensional linear regression

Suppose that we have a regression problem with a continuous response variable Y and a set of p covariates $X^{(1)},\dots,X^{(p)}$. Consider the setting where the observed data are realisations of

$$(Y_1,\boldsymbol{X}_1),\dots,(Y_n,\boldsymbol{X}_n),$$

with p-dimensional covariates $\boldsymbol{X}_i=(X_i^{(1)},\dots,X_i^{(p)})\in\mathscr{X}\subset\mathbb{R}^p$ and univariate responses $Y_i\in\mathscr{Y}\subset\mathbb{R}$. In high dimensional settings, we have $n\ll p$. For a continuous response variable Y, a simple but useful model is the linear regression model

$$Y_i=\beta_0+\sum_{j=1}^{p}\beta_j X_i^{(j)}+\varepsilon_i,\qquad i=1,\dots,n, \tag{2.1}$$

where $\beta_1,\dots,\beta_p$ are the regression coefficients associated with the p covariates, and $\varepsilon_1,\dots,\varepsilon_n$ are independent and identically distributed random errors with $\mathrm{E}(\varepsilon_i)=0$ and $\mathrm{Var}(\varepsilon_i)=\sigma^2>0$. For now, we do not make any "normality assumption" on the errors.

For simplicity and without loss of generality, we here assume that the intercept is zero, that is, $\beta_0=0$. In practice, this can be easily achieved if the response variable and all covariates are centred to have zero mean (by subtracting observations from their mean), because then $\beta_0=\frac{1}{n}\sum_{i=1}^{n}Y_i=0$.

In matrix notation, regression model (2.1), after adopting $\beta_0=0$, can be written as

$$\boldsymbol{Y}=\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{\varepsilon}, \tag{2.2}$$

in which

$$\boldsymbol{Y}=(Y_1,\dots,Y_n),\qquad \boldsymbol{X}=\begin{bmatrix}\boldsymbol{X}_1^T\\ \vdots\\ \boldsymbol{X}_n^T\end{bmatrix},$$

$$\boldsymbol{\beta}=(\beta_1,\dots,\beta_p),\qquad \boldsymbol{\varepsilon}=(\varepsilon_1,\dots,\varepsilon_n).$$

When $n\ll p$, the ordinary least squares (OLS) method does not provide a unique solution and will heavily overfit the data. To see this, consider the least square estimator

$$\hat{\boldsymbol{\beta}}_{OLS}=\underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\left\|\boldsymbol{\varepsilon}\right\|_2^2=\underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\left\|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right\|_2^2.$$

Simple algebra shows that the OLS estimator must satisfy the so-called normal equations

$$(\boldsymbol{X}^T\boldsymbol{X})\hat{\boldsymbol{\beta}}_{OLS} = \boldsymbol{X}^T\boldsymbol{Y}.$$

For standard linear regression when $n > p$ as in low dimensional data and if $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ exists, we obtain that $\hat{\boldsymbol{\beta}}_{OLS} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$. It is not difficult to show that the mean square error of the OLS estimator is (see `Problem Sheets`)

$$E\big(\|\hat{\boldsymbol{\beta}}_{OLS} - \boldsymbol{\beta}\|_2^2\big) = \mathrm{tr}\big((\boldsymbol{X}^T\boldsymbol{X})^{-1}\big)\sigma^2. \tag{2.3}$$

We can show that this estimation error gets larger as dimension p gets larger (the curse of dimensionality). To see this, consider a simple case when the columns of $\boldsymbol{X}$ are orthonormal (i.e., orthogonal with norm 1) which means $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}_p$. In this case, (2.3) simplifies to

$$E\big(\|\hat{\boldsymbol{\beta}}_{OLS} - \boldsymbol{\beta}\|_2^2\big) = p\sigma^2.$$

Hence, the larger p, the larger estimation error. In fact, if $p \to \infty$, $E\big(\|\hat{\boldsymbol{\beta}}_{OLS} - \boldsymbol{\beta}\|_2^2\big) \to \infty$, which is not desirable.

Furthermore, the prediction error of the least squares method is as follows (see `Problem Sheets`)

$$E\big(\frac{1}{n}\|\boldsymbol{X}\hat{\boldsymbol{\beta}}_{OLS} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2\big) = \frac{\sigma^2}{n}p, \tag{2.4}$$

which clearly diverges with the dimension p when $p > n$.

In high dimensional situations when $n \ll p$, $\boldsymbol{X}^T\boldsymbol{X}$ does not have full column rank because

$$\mathrm{rank}\big(\boldsymbol{X}^T\boldsymbol{X}\big) \le \min\big\{\mathrm{rank}(\boldsymbol{X}^T), \mathrm{rank}(\boldsymbol{X})\big\} \le \min\{n, p\} = n < p.$$

Therefore, $\boldsymbol{X}^T\boldsymbol{X}$ is singular (not invertible) in high dimensional settings. This implies that the OLS estimator is not unique when $n \ll p$. Therefore, the OLS method fails to work in high dimensional settings when $n \ll p$.

Due to the above issue, the ordinary least square regression is known as an ill-posed problem in high dimensional situations with $n \ll p$. This is where regularisation comes into play. The role of regularisation is to make this as a well-posed problem. This can be done by imposing some constraints (or conditions) on the regression coefficients in order to regularise the high dimensional regression problem. The constraint

is often in the form of a penalty term on the regression coefficients. We first explain the regularisation idea in the general form in the next section, which includes ridge and lasso regression as special cases.

2.3. Regularisation with L_q-norm penalty

A more general form of regularisation is based on the L_q-norm penalty. The main idea of regularisation is to restrict the parameter values through a penalty term such as

$$\sum_{j=1}^{p} |\beta_j|^q \leq t$$

where $t \geq 0$ is a threshold value. Obviously, if $t = \infty$ then there will be no restriction (same as OLS), and if $t = 0$ then all parameters have to be 0 (not practical). This more general form of regularisation, often known as "bridge regression", tries to solve the following problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \left\{ \left\| \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta} \right\|_2^2 \right\} \quad \text{w.r.t. constraint} \quad \sum_{j=1}^{p} |\beta_j|^q \leq t. \tag{2.5}$$

The penalty term $\sum_{j=1}^{p} |\beta_j|^q = \left\| \boldsymbol{\beta} \right\|_q^q \leq t$ puts a constraint on the magnitude of regression coefficients, which helps avoid the aforementioned issue with the OLS method.

The optimisation problem (2.5) is equivalent to solving

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \left\{ \left\| \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta} \right\|_2^2 + \lambda \left\| \boldsymbol{\beta} \right\|_q^q \right\}, \tag{2.6}$$

in which $\lambda \geq 0$ is a regularisation or shrinkage parameter. The equivalence of (2.5) and (2.6) holds because $\left\| \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta} \right\|_2^2$ is convex in $\boldsymbol{\beta}$ and the constraint $\sum_{j=1}^{p} |\beta_j|^q \leq t$ is convex with $q \geq 1$, so there is a one-to-one correspondence between t and λ, which depends on data. This is generally known as "Lagrangian duality". The choice of the shrinkage parameter λ is crucial and in practice it is often chosen based on cross validation, which is usually implemented in standard software including R.

It is important to note that the optimisation problem (2.6) may be convex or non-convex depending on the value of q:

- If $q \geq 1$, it is convex,
- If $0 < q < 1$, it is not convex.

The convexity helps solve the more general optimisation problem (2.6) for $q \geq 1$ using standard techniques for convex problems, such as gradient descent, which enable efficient computation of the estimator. For example, the R package `glmnet` uses a coordinate descent method which is very efficient for finding the estimates of parameters $\boldsymbol{\beta}$ for convex problems like $q = 1$ (lasso) or $q = 2$ (ridge). We will study this coordinate descent method later in Section 2.5.1. Note that solving the problem for $0 < q < 1$ requires special optimisation techniques for non-convex problems, which is beyond the level of this course.

Remark 2.1. If we set $q = 0$, the optimisation problem (2.6) becomes

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p}\left\{\|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_0^0\right\}, \tag{2.7}$$

where the L_0-norm penalty is $\|\boldsymbol{\beta}\|_0^0 = \sum_{j=1}^p I(\beta_j \neq 0)$, which concerns the number of non-zero parameters. This penalises the number of parameters instead of the magnitude of parameters. Many well known classical methods such as the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) fall into this framework.

2.4. Ridge regression in high dimensions

Ridge regression, first proposed by Hoerl and Kennard [15], uses the L_2-norm penalty and provides the following estimator with $\lambda \geq 0$

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p}\left\{\|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_2^2\right\}. \tag{2.8}$$

By solving the above problem using direct differentiation with respect to $\boldsymbol{\beta}$ (see `Problem Sheets`), we obtain the closed-form estimator

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \left(\boldsymbol{X}^T\boldsymbol{X}+\lambda\boldsymbol{I}_p\right)^{-1}\boldsymbol{X}^T\boldsymbol{Y}, \tag{2.9}$$

where we recall $\boldsymbol{I}_p$ is the $p \times p$ identity matrix.

If $\lambda > 0$, matrix $\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_p$ in (2.9) is always invertible even with $n \ll p$. To prove this, it suffices to show that this matrix is positive definite under the constraint on λ (remember a matrix $\boldsymbol{A}$ is positive definite if $\boldsymbol{z}^T\boldsymbol{A}\boldsymbol{z} > 0$ for all $\boldsymbol{z} \neq \boldsymbol{0}$). For this, just write,

for any $\boldsymbol{z} \neq \boldsymbol{0}$,

$$\boldsymbol{z}^T\left(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_p\right)\boldsymbol{z} = \boldsymbol{z}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{z} + \lambda \boldsymbol{z}^T\boldsymbol{I}_p\boldsymbol{z} = \underbrace{\left(\boldsymbol{X}\boldsymbol{z}\right)^T\boldsymbol{X}\boldsymbol{z}}_{\geq 0} + \underbrace{\lambda \boldsymbol{z}^T\boldsymbol{z}}_{>0} > 0.$$

PROPOSITION 2.2. *If the design matrix $\boldsymbol{X}$ has full column rank, then $\boldsymbol{X}^T\boldsymbol{X}$ is invertible.*

PROOF. Consider the scalar quantity $\boldsymbol{z}^T\left(\boldsymbol{X}^T\boldsymbol{X}\right)\boldsymbol{z}$. It can be written as $\left(\boldsymbol{X}\boldsymbol{z}\right)^T\boldsymbol{X}\boldsymbol{z}$. Written this way, it now represents the squared length of vector $\boldsymbol{X}\boldsymbol{z}$. This squared length is non-negative for sure. If matrix $\boldsymbol{X}$ has full column rank, then its Null Space only contains the zero vector. This implies that squared length of $\boldsymbol{X}\boldsymbol{z}$ is always greater than 0 for all $\boldsymbol{z} > 0$. Therefore, the symmetric matrix $\boldsymbol{X}^T\boldsymbol{X}$ is positive definite and hence invertible (also recall all eigenvalues of a positive definite matrix are positive). □

Obviously, if $\lambda = 0$ (i.e., no penalty) then $\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \hat{\boldsymbol{\beta}}_{\text{OLS}}$, provided that $\boldsymbol{X}$ has full column rank considering Proposition 2.2. With a $\lambda > 0$, the ridge penalty shrinks parameters towards zero, however no estimates will be exactly 0 because the L_2-norm penalty provides a smooth constraint (unlike the L_1-norm penalty as we see later in this chapter). Thus, ridge regression does not produce a sparse solution. This is not desirable in high dimensional situations because we wish to have a sparse solution so we can get:

- an easier interpretation of parameters, and
- a smaller prediction error (the more variables, the higher uncertainty).

The singular value decomposition (SVD) of the matrix $\boldsymbol{X}$ gives us some useful insight into the nature of ridge regression. The SVD is a simple and useful method in the analysis of many statistical methods (in Chapter 3 we use it for principal component analysis). The SVD of the $n \times p$ matrix $\boldsymbol{X}$ gives a factorisation of $\boldsymbol{X}$ in the form of

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T, \tag{2.10}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are, respectively, $n \times n$ and $p \times p$ orthogonal matrices (i.e., $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}_n$ and $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}_p$), with the columns of $\boldsymbol{U}$ spanning the column space of $\boldsymbol{X}$ and the columns of $\boldsymbol{V}$ spanning the row space of $\boldsymbol{X}$, and also $\boldsymbol{D}$ is an $n \times p$ matrix

of singular values as follow

$$\boldsymbol{D} = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & d_p \end{bmatrix} \quad \text{if } n = p$$

and

$$\boldsymbol{D} = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & d_p \\ \vdots & \vdots & \vdots \end{bmatrix} \quad \text{if } n > p$$

and

$$\boldsymbol{D} = \begin{bmatrix} d_1 & 0 & 0 & \cdots \\ 0 & \ddots & 0 & \cdots \\ 0 & 0 & d_n & \cdots \end{bmatrix} \quad \text{if } n < p$$

with $d_1 \geq \cdots \geq d_n \geq \cdots \geq d_p \geq 0$ being the singular values of $\boldsymbol{X}$. Note that the dots down and right to the main block of singular values d_j are all zeros. This formulation is known as the full SVD, which holds for both low and high dimensional data cases.

Using the full SVD formula (2.10), we get $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{V}^T$ and then

$$\begin{aligned} \boldsymbol{X}\hat{\boldsymbol{\beta}}_{\text{Ridge}} &= \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_p\right)^{-1}\boldsymbol{X}^T\boldsymbol{Y} \\ &= \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T\left(\boldsymbol{V}\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{V}^T + \lambda\boldsymbol{V}\boldsymbol{V}^T\right)^{-1}\boldsymbol{V}\boldsymbol{D}^T\boldsymbol{U}^T\boldsymbol{Y} \\ &= \boldsymbol{U}\boldsymbol{D}\left(\boldsymbol{D}^T\boldsymbol{D} + \lambda\boldsymbol{I}_p\right)^{-1}\boldsymbol{D}^T\boldsymbol{U}^T\boldsymbol{Y} \\ &= \sum_{j=1}^{n} \boldsymbol{u}_j \frac{d_j^2}{d_j^2 + \lambda}\boldsymbol{u}_j^T\boldsymbol{Y}, \end{aligned}$$

where the $\boldsymbol{u}_j$ are the columns of $\boldsymbol{U}$. Note that the sum is over n elements because $\boldsymbol{D}\left(\boldsymbol{D}^T\boldsymbol{D} + \lambda\boldsymbol{I}_p\right)^{-1}\boldsymbol{D}^T$ is an $n \times n$ diagonal matrix. Since $\lambda \geq 0$, we have $d_j^2/(d_j^2 + \lambda) \leq 1$. So ridge regression computes the coordinates of $\boldsymbol{Y}$ with respect to the orthonormal basis $\boldsymbol{U}$ and shrinks these coordinates by the factors $d_j^2/(d_j^2 + \lambda)$. A greater amount of shrinkage is applied to the coordinates of basis vectors with smaller d_j^2. Since $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{V}^T$ is an eigendecomposition, the smaller singular values d_j^2 are actually the smaller eigenvalues of $\boldsymbol{X}^T\boldsymbol{X}$ corresponding to the directions (i.e., eigenvectors) having small variance (we will see more details on eigenvalues and eigenvectors in Chapter 3).

Ridge regression shrinks these directions the most. According to the above formula, the ridge estimator shrinks $\boldsymbol{Y}$ in the directions $\boldsymbol{u}_j$ where $d_j \ll \lambda$, whereas it leaves $\boldsymbol{Y}$ almost unchanged in the directions $\boldsymbol{u}_j$ where $d_j \gg \lambda$. In practice, we use the R package `glmnet` to implement the ridge regression (we will have real data examples in our Computer Practical session).

The variance of the ridge estimator is also given by (see `Problem Sheets`)

$$\text{Var}(\hat{\boldsymbol{\beta}}_{\text{Ridge}}) = \sigma^2 \sum_{j=1}^{p} \boldsymbol{v}_j \Big(\frac{d_j}{d_j^2 + \lambda}\Big)^2 \boldsymbol{v}_j^T, \tag{2.11}$$

in which the $\boldsymbol{v}_j$ are the columns of $\boldsymbol{V}$. Note that the variance goes to 0 if $\lambda \to \infty$. However, $\lambda \to \infty$ implies the shrinkage penalty becomes very large and the coefficient estimates will all must be 0. This is not practical.

Compact SVD:
In the SVD process, depending on whether $n \geq p$ or $n < p$, we have

- If $n \geq p$ then all singular values $d_1, \ldots, d_p$ are non-zero, that is, $d_1 \geq \cdots \geq d_p > 0$.
- If $n < p$ then only the first n singular values $d_1, \ldots, d_p$ are non-zero, that is, $d_1 \geq \cdots \geq d_n > 0$ and $d_{n+1} = d_{n+2} = \cdots = d_p = 0$.

Note that if one or more singular values $d_1, \ldots, d_p$ are zero (i.e., $d_j = 0$ for some j) then $\boldsymbol{X}$ is singular. So in the high dimensional case $n \ll p$, $\boldsymbol{X}$ is always singular ($\boldsymbol{X}^T\boldsymbol{X}$ is not invertible). The number of non-zero singular values is the rank of matrix $\boldsymbol{X}$.

In a more concise and efficient way, we can say that if $\text{rank}(\boldsymbol{X}) = r \leq min(n, p)$ then $d_1 \geq \cdots \geq d_r > 0$ and $d_{r+j} = 0$ for any possible j. So, in high dimensional settings $n \ll p$, we can get a compact SVD as follows

$$\boldsymbol{X} = \boldsymbol{U}^* \boldsymbol{D}^* \boldsymbol{V}^{*T}, \tag{2.12}$$

where $\boldsymbol{U}^*$ and $\boldsymbol{V}^*$ are, respectively, $n \times r$ and $p \times r$ semi-orthogonal matrices (i.e., $\boldsymbol{U}^{*T}\boldsymbol{U}^* = \boldsymbol{I}_r$ and $\boldsymbol{V}^{*T}\boldsymbol{V}^* = \boldsymbol{I}_r$), and also $\boldsymbol{D}^*$ is an $r \times r$ diagonal matrix as

$$\boldsymbol{D}^* = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & d_r \end{bmatrix}$$

containing only the non-zero singular values. This compact SVD formulation is useful for high dimensional data (we will also use it for high dimensional PCA in chapter 3).

2.5. Lasso regression in high dimensions

Lasso regression, first proposed by Tibshirani [29], uses the L_1-norm penalty and provides the following estimator with $\lambda \geq 0$

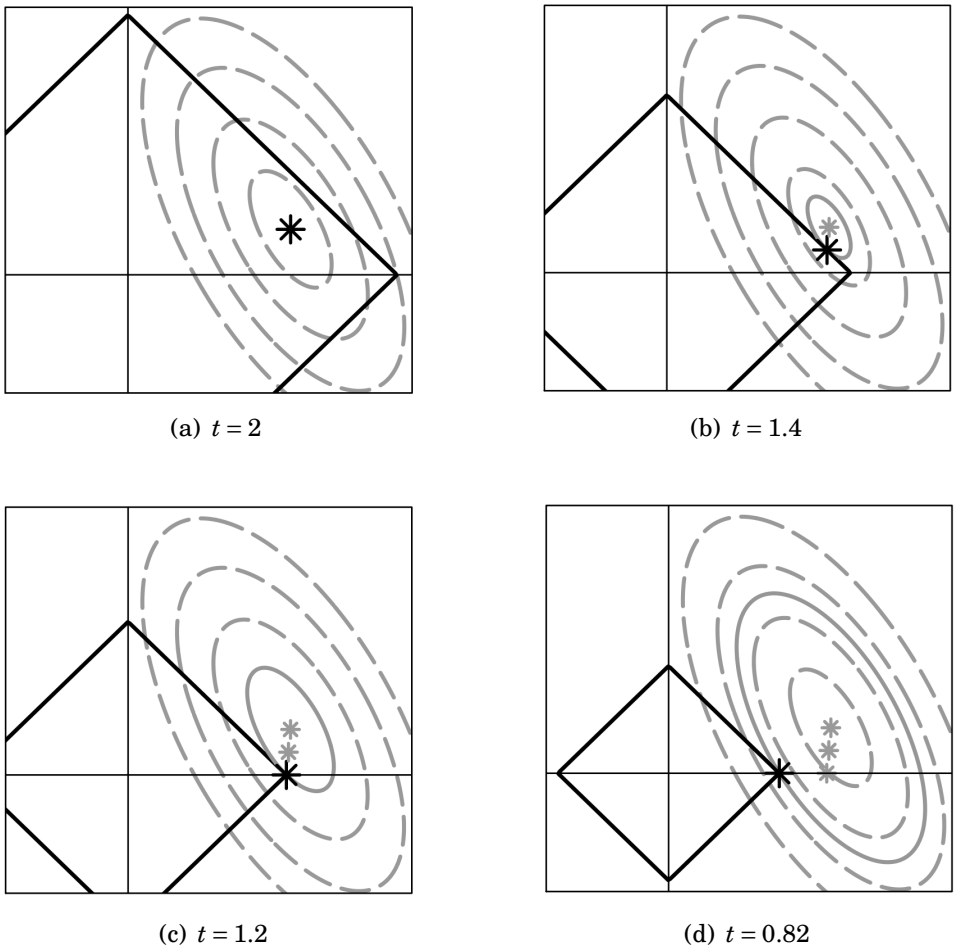
$$\hat{\boldsymbol{\beta}}_{\text{Lasso}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\left\{\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1^1\right\}. \tag{2.13}$$

Unlike the ridge estimator, the lasso estimator does not have a closed-form expression. This is due to the absolute value function in L_1-norm penalty (recall $\|\boldsymbol{\beta}\|_1^1 = \sum_{j=1}^p |\beta_j|$). Taking derivative of absolute value function is more complicated compared to squared value function as in ridge regression with L_2-norm penalty. However, the lasso optimisation problem is still convex, enabling efficient computation of lasso estimator using numerical methods such as gradient descent. Note that the R package `glmnet` is written based on such numerical computations.

Geometric insight:
Let us denote by $B_{L_1}(t)$ the L_1-ball of radius t defined as $B_{L_1}(t) = \{\boldsymbol{\beta} \in \mathbb{R}^p : \|\boldsymbol{\beta}\|_1^1 \leq t\}$, which represents the constraint with L_1 penalty with given t. In Figure 2.1, the level sets of function $\boldsymbol{\beta} \to \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$ are plotted (dashed lines), together with the constraint $\|\boldsymbol{\beta}\|_1^1 \leq t$ (plain lines) for decreasing values of t in an example with $p = 2$. Note that for t small enough which corresponds to λ large enough, some coordinate $\hat{\boldsymbol{\beta}}_j$ are set to 0. This illustrates the fact that the lasso estimator selects variables for λ large enough (i.e., t small enough).

Remark 2.3. The variable selection ability of lasso for t small enough (or λ large enough) comes from the non-smoothness of the $B_{L_1}(t)$. Replacing the L_1-ball by the L_q-ball, $B_{L_q}(t) = \{\boldsymbol{\beta} \in \mathbb{R}^p : \|\boldsymbol{\beta}\|_q^q \leq t\}$, for $1 < q < \infty$ would not lead to variable selection, which makes the lasso special in this regard.

Analytic insights:
In order to study analytical insights into the lasso regression for high dimensional data, let us first review some basic definitions and properties of convex multivariate



(a) $t = 2$ (b) $t = 1.4$

(c) $t = 1.2$ (d) $t = 0.82$

FIGURE 2.1. The level sets of the function $\boldsymbol{\beta} \to \left\| \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta} \right\|_2^2$ (dashed grey lines), together with the constraint $\|\boldsymbol{\beta}\|_1^1 \leq t$ (black plain lines) for decreasing values of t with $p = 2$. The dark stars represent the lasso estimator for the current value of t. When $t = 2$, the L_1-norm of lasso estimator is smaller than t, so the lasso estimator coincides with the OLS estimator. When $t = 1.2$ or smaller, the second coordinate of lasso estimator is equal to 0.

functions. In the following, let $f(\boldsymbol{x}) : D(f) \to \mathbb{R}$ be a real-valued multivariate function where $D(f)$ is the domain of f and $\boldsymbol{x}$ is a vector with $\boldsymbol{x} \in D(f)$.

- An affine function f is a linear function plus a translation, that is, $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$, where $\boldsymbol{A}$, $\boldsymbol{x}$, $\boldsymbol{b}$ are of appropriate dimensions. If f is real-valued, $f(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x} + b$.

- A region $\mathscr{R}$ is convex if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathscr{R}$ we have $t\boldsymbol{x} + (1-t)\boldsymbol{y} \in \mathscr{R} \ \ \forall t \in [0,1]$. For example, in one dimensional space, any line segment connecting two points in $\mathbb{R}$ lies entirely inside $\mathbb{R}$.

- A function f is a convex function if its domain $D(f)$ is a convex set and

$$f(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \leq tf(\boldsymbol{x}) + (1-t)f(\boldsymbol{y}) \quad \forall t \in [0,1] \text{ and } \boldsymbol{x}, \boldsymbol{y} \in D(f).$$

  We say f is strictly convex if strict inequality holds for all $t \in [0,1]$ when $\boldsymbol{x} \neq \boldsymbol{y}$.

- If function f is convex and differentiable, we have (see `Problem Sheets`)

$$f(\boldsymbol{y}) - f(\boldsymbol{x}) \geq (\boldsymbol{y} - \boldsymbol{x})^T \nabla f(\boldsymbol{x}) \quad \forall \boldsymbol{x}, \boldsymbol{y} \in D(f),$$

  in which $\nabla f(\boldsymbol{x})$ is the derivative of f at $\boldsymbol{x}$ (i.e., gradient of f).

- A multivariate function f that is twice differentiable at every point in its convex domain $D(f)$ is convex if and only if the hessian $\nabla^2 f$ is a positive semi-definite matrix.

- A subgradient of a convex function f at $\boldsymbol{x} \in D(f)$ is any vector $\boldsymbol{w}$ of suitable dimension such that $f(\boldsymbol{y}) - f(\boldsymbol{x}) \geq (\boldsymbol{y} - \boldsymbol{x})^T \boldsymbol{w} \ \ \forall \boldsymbol{y} \in D(f)$.

- The subdifferential of a convex function f at $\boldsymbol{x}$, denoted by $\partial f(\boldsymbol{x})$, is the set of all subgradients of f at $\boldsymbol{x}$, that is,

$$\partial f(\boldsymbol{x}) = \{\boldsymbol{w} : \boldsymbol{w} \text{ is a subgradient of } f \text{ at } \boldsymbol{x}\} = \{\boldsymbol{w} : f(\boldsymbol{y}) - f(\boldsymbol{x}) \geq (\boldsymbol{y} - \boldsymbol{x})^T \boldsymbol{w} \ \ \forall \boldsymbol{y} \in D(f)\}.$$

  It is important to note that when "f is convex and differentiable", we have that $\partial f(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$, so the derivative and subdifferential will be the same.

Example 2.1. Consider the absolute value function $f(x) = |x|$. The domain of f is the real numbers $D(f) = \mathbb{R}$. Clearly $\mathbb{R}$ is a convex set since adding any two real numbers is a real number. A subgradient of f at $x = 1$ would be $w = 1$ since for all $y \in R$ we have $|y| - 1 \geq (y-1)1$. Similarly, $w = -1$ is a subgradient of f at $x = -1$ since $|y| - 1 \geq (y+1)(-1)$. Generally, the subdifferential of $f(x) = |x|$ is

$$\partial f(x) = \partial |x| = \{w : w = \text{sign}(x) \text{ for } x \neq 0, w \in [-1,1] \text{ for } x = 0\},$$

where $\text{sign}(x) = 1(x > 0) - 1(x \leq 0)$.

PROPOSITION 2.4. *The minimisers of a convex multivariate function $f(\boldsymbol{x}) : D(f) \to \mathbb{R}$ are characterised by*

$$\boldsymbol{x}^* \in \underset{\boldsymbol{x} \in D(f)}{\arg\min} f(\boldsymbol{x}) \iff \mathbf{0} \in \partial f(\boldsymbol{x}^*).$$

This is an immediate result of the fact that $f(\boldsymbol{y}) - f(\boldsymbol{x}^*) \geq (\boldsymbol{y} - \boldsymbol{x}^*)^T \mathbf{0}$ for all $\boldsymbol{y} \in D(f)$, because $f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$ for all $\boldsymbol{y} \in D(f)$.

Using the above definitions and properties of convex functions, the subdifferential of the lasso optimisation function $L(\boldsymbol{\beta}) = \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1^1$ is

$$\partial L(\boldsymbol{\beta}) = \{-2\boldsymbol{X}^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \boldsymbol{z} : \boldsymbol{z} \in \partial \|\boldsymbol{\beta}\|_1^1\}. \tag{2.14}$$

To see this just note that the derivative of $\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$ with respect to $\boldsymbol{\beta}$ is $-2\boldsymbol{X}^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})$ (see `Problem Sheets`) and also $\boldsymbol{z}$ is a subgradient of $\|\boldsymbol{\beta}\|_1^1$. The first-order optimality condition in Proposition 2.4 ensures the existence of $\hat{\boldsymbol{z}} \in \partial \|\hat{\boldsymbol{\beta}}_\lambda\|_1^1$, fulfilling

$$-2\boldsymbol{X}^T(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}_\lambda) + \lambda \hat{\boldsymbol{z}} = 0.$$

Note that for simplicity of notation we here use $\hat{\boldsymbol{\beta}}_\lambda$ for the lasso solution, which depends on the regularisation parameter λ. Hence, the lasso estimator must satisfy

$$\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}}_\lambda + \frac{\lambda}{2}\hat{\boldsymbol{z}} = \boldsymbol{X}^T\boldsymbol{Y} \tag{2.15}$$

for some $\hat{\boldsymbol{z}} \in \mathbb{R}^p$, fulfilling $\hat{\boldsymbol{z}}_j = \text{sign}([\hat{\boldsymbol{\beta}}_\lambda]_j)$ if $[\hat{\boldsymbol{\beta}}_\lambda]_j \neq 0$ and $\hat{\boldsymbol{z}}_j \in [-1, 1]$ if $[\hat{\boldsymbol{\beta}}_\lambda]_j = 0$.

Equation (2.15) is known as the KKT conditions for lasso problem. We now investigate the variable selection induced by this formula. For this, we first consider a simple case of orthonormal design matrix $\boldsymbol{X}$.

Orthonormal setting:
Consider a simple case where the columns of $\boldsymbol{X}$ are orthonormal and thus $\boldsymbol{X}^T\boldsymbol{X} = I_p$. It means that the columns of $\boldsymbol{X}$ are orthogonal with norm 1. Notice that this necessarily implies $n \geq p$. In this simple case, the KKT conditions in equation (2.15) simplify to

$$[\hat{\boldsymbol{\beta}}_\lambda]_j + \frac{\lambda}{2}\text{sign}([\hat{\boldsymbol{\beta}}_\lambda]_j) = [\boldsymbol{X}^T\boldsymbol{Y}]_j \quad \text{for } [\hat{\boldsymbol{\beta}}_\lambda]_j \neq 0,$$

where the notation $[\boldsymbol{a}]_j$ denotes the j-th element of a vector $\boldsymbol{a}$. This result enforces both $\text{sign}([\hat{\boldsymbol{\beta}}_\lambda]_j) = \text{sign}([\boldsymbol{X}^T\boldsymbol{Y}]_j)$ and $[\hat{\boldsymbol{\beta}}_\lambda]_j = [\boldsymbol{X}^T\boldsymbol{Y}]_j - \frac{\lambda}{2}\text{sign}([\boldsymbol{X}^T\boldsymbol{Y}]_j)$ for $[\hat{\boldsymbol{\beta}}_\lambda]_j \neq 0$. It turns out that we cannot have $[\hat{\boldsymbol{\beta}}_\lambda]_j \neq 0$ when $\big|[\boldsymbol{X}^T\boldsymbol{Y}]_j\big| \leq \frac{\lambda}{2}$ (see `Problem Sheets`). Therefore, we have $[\hat{\boldsymbol{\beta}}_\lambda]_j = 0$ when $\big|[\boldsymbol{X}^T\boldsymbol{Y}]_j\big| \leq \frac{\lambda}{2}$ and $[\hat{\boldsymbol{\beta}}_\lambda]_j = [\boldsymbol{X}^T\boldsymbol{Y}]_j - \frac{\lambda}{2}\text{sign}([\boldsymbol{X}^T\boldsymbol{Y}]_j)$ when $\big|[\boldsymbol{X}^T\boldsymbol{Y}]_j\big| > \frac{\lambda}{2}$. To sum this up in the orthonormal setting, the lasso estimator is

$$[\hat{\boldsymbol{\beta}}_\lambda]_j = [\boldsymbol{X}^T\boldsymbol{Y}]_j\Big(1 - \frac{\lambda}{2\big|[\boldsymbol{X}^T\boldsymbol{Y}]_j\big|}\Big)_+, \quad j = 1,\dots,p, \quad \text{with } (x)_+ = \max(x,0),$$

which can be equivalently rewritten as

$$[\hat{\boldsymbol{\beta}}_\lambda]_j = \frac{[\boldsymbol{X}^T\boldsymbol{Y}]_j}{\big|[\boldsymbol{X}^T\boldsymbol{Y}]_j\big|}\Big(\big|[\boldsymbol{X}^T\boldsymbol{Y}]_j\big| - \frac{\lambda}{2}\Big)_+ = \text{sign}([\boldsymbol{X}^T\boldsymbol{Y}]_j)\Big(\big|[\boldsymbol{X}^T\boldsymbol{Y}]_j\big| - \frac{\lambda}{2}\Big)_+, \quad j = 1,\dots,p.$$

In this case, lasso selects the coordinates j such that $\big|[\boldsymbol{X}^T\boldsymbol{Y}]_j\big| > \frac{\lambda}{2}$.

Non-orthogonal setting:
In the general case when the columns of $\boldsymbol{X}$ are not orthogonal, there is no analytic formula for $\hat{\boldsymbol{\beta}}_\lambda$, and the lasso will not select the same variables in general. In the general case, equation (2.15) gives

$$\begin{aligned} 0 \leq \hat{\boldsymbol{\beta}}_\lambda^T\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}}_\lambda &= \langle\hat{\boldsymbol{\beta}}_\lambda, \boldsymbol{X}^T\boldsymbol{Y} - \frac{\lambda}{2}\hat{\boldsymbol{z}}\rangle \\ &= \sum_{j:[\hat{\boldsymbol{\beta}}_\lambda]_j\neq 0} [\hat{\boldsymbol{\beta}}_\lambda]_j\big([\boldsymbol{X}^T\boldsymbol{Y}]_j - \frac{\lambda}{2}\text{sign}([\hat{\boldsymbol{\beta}}_\lambda]_j)\big), \end{aligned}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product of two vectors (see Appendix A). From this result, we deduce that $\hat{\boldsymbol{\beta}}_\lambda = \boldsymbol{0}$ for $\lambda \geq 2\max_j\{[\boldsymbol{X}^T\boldsymbol{Y}]_j\} = 2\big\|\boldsymbol{X}^T\boldsymbol{Y}\big\|_\infty$. And when $\lambda < 2\big\|\boldsymbol{X}^T\boldsymbol{Y}\big\|_\infty$, the lasso estimator $\hat{\boldsymbol{\beta}}_\lambda$ is non-zero, however there is no simple formula describing its support.

2.5.1. Computing lasso solution via coordinate descent algorithm

The lasso estimator (2.13) is the minimiser of the convex but non-differentiable function $L(\boldsymbol{\beta}) = \big\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\big\|_2^2 + \lambda\big\|\boldsymbol{\beta}\big\|_1^1$. An efficient numerical method for computing the lasso estimator is the coordinate descent which is described below.

Repeatedly minimising $L(\boldsymbol{\beta}) = L(\boldsymbol{\beta}_1,\dots,\boldsymbol{\beta}_p)$ with respect to each coordinate $\boldsymbol{\beta}_j$ is a simple and efficient procedure for minimising $L(\boldsymbol{\beta})$. This algorithm converges to the

lasso estimator thanks to the convexity of $L(\boldsymbol{\beta})$. Suppose the covariates are standardised so that the columns $\boldsymbol{X}_j$ of $\boldsymbol{X}$ have norm 1. The derivative of the function $L(\boldsymbol{\beta}) = \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1^1$ with respect to $\boldsymbol{\beta}_j$ is

$$\partial_j L(\boldsymbol{\beta}) = -2\boldsymbol{X}_j^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda\frac{\boldsymbol{\beta}_j}{|\boldsymbol{\beta}_j|} = 2\boldsymbol{\beta}_j - 2R_j + \lambda\frac{\boldsymbol{\beta}_j}{|\boldsymbol{\beta}_j|} \quad \forall\, \boldsymbol{\beta}_j \neq 0,$$

where $R_j = \boldsymbol{X}_j^T\big(\boldsymbol{Y} - \sum_{k\neq j}\boldsymbol{\beta}_k\boldsymbol{X}_k\big)$.
Since $L(\boldsymbol{\beta})$ is convex, the minimiser of $L(\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_{j-1},\boldsymbol{\beta}_j,\boldsymbol{\beta}_{j+1},\ldots,\boldsymbol{\beta}_p)$ is the solution in $\boldsymbol{\beta}_j$ of $\partial_j L(\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_{j-1},\boldsymbol{\beta}_j,\boldsymbol{\beta}_{j+1},\ldots,\boldsymbol{\beta}_p) = 0$ when such a solution exists, and it is $\boldsymbol{\beta}_j = 0$ otherwise. Therefore, the function $L(\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_{j-1},\boldsymbol{\beta}_j,\boldsymbol{\beta}_{j+1},\ldots,\boldsymbol{\beta}_p)$ is minimum in

$$\boldsymbol{\beta}_j = R_j\Big(1 - \frac{\lambda}{2|R_j|}\Big)_+ \quad \text{with } R_j = \boldsymbol{X}_j^T\big(\boldsymbol{Y} - \sum_{k\neq j}\boldsymbol{\beta}_k\boldsymbol{X}_k\big). \tag{2.16}$$

Repeatedly computing $\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_p,\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_p,\ldots$ according to (2.16) gives the coordinate descent algorithm, which is summarised in Algorithm 1.

Algorithm 1: Coordinate descent algorithm for lasso

Initialisation: $\boldsymbol{\beta} = \boldsymbol{\beta}_{\text{init}}$ with $\boldsymbol{\beta}_{\text{init}} \in \mathbb{R}^p$ arbitrary.
Repeat, until convergence of $\boldsymbol{\beta}$, the loop:
for $j = 1,\ldots,p$

$$\boldsymbol{\beta}_j = R_j\Big(1 - \frac{\lambda}{2|R_j|}\Big)_+ \quad \text{with } R_j = \boldsymbol{X}_j^T\big(\boldsymbol{Y} - \sum_{k\neq j}\boldsymbol{\beta}_k\boldsymbol{X}_k\big).$$

Output: $\boldsymbol{\beta}$

The coordinate descent algorithm for lasso is implemented in the R package `glmnet`. For further illustration, we refer to our Computer Practical session where we analyse real data applications with high dimensional data.

2.5.2. Theory for lasso in high dimensions

In this section, we study some of the theoretical properties of the lasso regression, expanding the results in Bühlmann and Van De Geer [5]. We here assume that the true regression model is linear (linearity is quite a big assumption!). Let $\boldsymbol{\beta}^0 = (\beta_1^0,\ldots,\beta_p^0)$ denote the unknown "true parameter values". The philosophy that could generally rescue us in high dimensional regression (and beyond) is to "believe" that in fact only a few of the unknown true parameters are non-zero. This is known as sparsity, which

we briefly discussed earlier. Without sparsity assumptions, high dimensional analysis is usually ill-posed. In fact, it is quite rare that all (many) covariates contribute to the response variable (for example, not all genes contribute to a disease; often a few do.). We thus define the number of true non-zero parameters as

$$S_0 := \{j : \beta_j^0 \neq 0\},$$

which is known as the "active set". The cardinality of the active set denoted by

$$s_0 := |S_0|$$

is known as the "sparsity index". Obviously, S_0 is unknown, so is s_0.

The basis of all our derivations in this subsection is the following result known as the "basic inequality".

LEMMA 2.5. *(Basic inequality) We have*

$$\frac{1}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\|_2^2 + \lambda\|\hat{\boldsymbol{\beta}}\|_1^1 \leq \frac{2}{n}\boldsymbol{\varepsilon}^T\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0) + \lambda\|\boldsymbol{\beta}^0\|_1^1,$$

in which $\hat{\boldsymbol{\beta}}$ is the lasso estimator (2.13).

PROOF. Since the lasso estimator $\hat{\boldsymbol{\beta}}$ in (2.13) is the minimiser of the lasso problem, we must have (note that dividing by a constant n does not have any impact on the minimisation problem)

$$\frac{1}{n}\|\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\|_2^2 + \lambda\|\hat{\boldsymbol{\beta}}\|_1^1 \leq \frac{1}{n}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}^0\|_2^2 + \lambda\|\boldsymbol{\beta}^0\|_1^1.$$

This can be rewritten as

$$\frac{1}{n}(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^T(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) + \lambda\|\hat{\boldsymbol{\beta}}\|_1^1 \leq \frac{1}{n}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}^0)^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}^0) + \lambda\|\boldsymbol{\beta}^0\|_1^1.$$

Simple calculations give

$$\frac{1}{n}(\boldsymbol{Y}^T\boldsymbol{Y} + \hat{\boldsymbol{\beta}}^T\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} - 2\boldsymbol{Y}^T\boldsymbol{X}\hat{\boldsymbol{\beta}}) + \lambda\|\hat{\boldsymbol{\beta}}\|_1^1 \leq \frac{1}{n}(\boldsymbol{Y}^T\boldsymbol{Y} + (\boldsymbol{\beta}^0)^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}^0 - 2\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta}^0) + \lambda\|\boldsymbol{\beta}^0\|_1^1,$$

which is further simplified to

$$\frac{1}{n}\hat{\boldsymbol{\beta}}^T\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} - \frac{2}{n}\boldsymbol{Y}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} + \lambda\|\hat{\boldsymbol{\beta}}\|_1^1 \leq \frac{1}{n}(\boldsymbol{\beta}^0)^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}^0 - \frac{2}{n}\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta}^0 + \lambda\|\boldsymbol{\beta}^0\|_1^1.$$

Now, replacing $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta}^0 + \boldsymbol{\varepsilon}$, we get

$$\frac{1}{n}\hat{\boldsymbol{\beta}}^T\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} - \frac{2}{n}(\boldsymbol{X}\boldsymbol{\beta}^0 + \boldsymbol{\varepsilon})^T\boldsymbol{X}\hat{\boldsymbol{\beta}} + \lambda\|\hat{\boldsymbol{\beta}}\|_1^1 \leq \frac{1}{n}(\boldsymbol{\beta}^0)^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}^0 - \frac{2}{n}(\boldsymbol{X}\boldsymbol{\beta}^0 + \boldsymbol{\varepsilon})^T\boldsymbol{X}\boldsymbol{\beta}^0 + \lambda\|\boldsymbol{\beta}^0\|_1^1,$$

and consequently

$$\frac{1}{n}\hat{\boldsymbol{\beta}}^T\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} - \frac{2}{n}(\boldsymbol{\beta}^0)^T\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} - \frac{2}{n}\boldsymbol{\varepsilon}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} + \lambda\|\hat{\boldsymbol{\beta}}\|_1^1 \leq -\frac{1}{n}(\boldsymbol{\beta}^0)^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}^0 - \frac{2}{n}\boldsymbol{\varepsilon}^T\boldsymbol{X}\boldsymbol{\beta}^0 + \lambda\|\boldsymbol{\beta}^0\|_1^1.$$

Rearranging this we obtain

$$\frac{1}{n}\hat{\boldsymbol{\beta}}^T\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} + \frac{1}{n}(\boldsymbol{\beta}^0)^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}^0 - \frac{2}{n}(\boldsymbol{\beta}^0)^T\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} + \lambda\|\hat{\boldsymbol{\beta}}\|_1^1 \leq \frac{2}{n}\boldsymbol{\varepsilon}^T\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0) + \lambda\|\boldsymbol{\beta}^0\|_1^1.$$

This actually rewrites as

$$\frac{1}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\|_2^2 + \lambda\|\hat{\boldsymbol{\beta}}\|_1^1 \leq \frac{2}{n}\boldsymbol{\varepsilon}^T\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0) + \lambda\|\boldsymbol{\beta}^0\|_1^1,$$

which completes the proof. □

From the basic inequality, the only random part in the upper bound for lasso solution is the term $2\boldsymbol{\varepsilon}^T\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)$, which is also known as the "empirical process" part of the problem. Using the Holder's inequality, the empirical process part can further be bounded in terms of L_1-norm of the parameters involved as follows

$$\frac{2}{n}\left|\boldsymbol{\varepsilon}^T\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\right| \leq \frac{2}{n}\|\boldsymbol{\varepsilon}^T\boldsymbol{X}\|_\infty\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_1^1. \tag{2.17}$$

The idea of the lasso penalty is that it should typically "overrule" the empirical process part. For this, let us introduce the set

$$\mathscr{F} := \left\{\frac{2}{n}\|\boldsymbol{\varepsilon}^T\boldsymbol{X}\|_\infty \leq \lambda_0\right\},$$

where we assume (quite arbitrarily) that $\lambda \geq 2\lambda_0$ to make sure that on $\mathscr{F}$ we can get rid of the random part of the problem.

It is not difficult to show that for a suitable value of λ_0, the set $\mathscr{F}$ has large probability. If the errors are assumed to be Gaussian, the argument goes as follows. Let us denote the diagonal elements of the sample covariance matrix (scaled by $1/n$) $\boldsymbol{S}_n = \boldsymbol{X}^T\boldsymbol{X}/n$ by

$$\hat{\sigma}_j^2 := [\boldsymbol{S}_n]_{jj}, \qquad j = 1, \ldots, p.$$

LEMMA 2.6. *Assume the errors* $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)$ *are Gaussian and all independent with mean* 0 *and variance* σ^2. *Suppose that* $\hat{\sigma}_j^2 = 1$ *for all* $j = 1, \ldots, p$ *(this can be achieved by scaling variables). Then, we have for all* $t > 0$ *and* $\lambda_0 = 2\sigma\sqrt{\frac{t^2+2\log(p)}{n}}$ *that*

$$P(\mathscr{F}) \geq 1 - 2\exp(-t^2/2).$$

PROOF. Since the errors are Gaussian and that $\hat{\sigma}_j^2 = 1$ (i.e., $\boldsymbol{X}_j^T \boldsymbol{X}_j = n$), it is easy to see that $V_j := \boldsymbol{\varepsilon}^T \boldsymbol{X}_j / (\sqrt{n\sigma^2}) \sim N(0,1)$. So

$$\begin{aligned}
P\Big(\frac{2}{n}\big\|\boldsymbol{\varepsilon}^T\boldsymbol{X}\big\|_\infty > \lambda_0\Big) &= P\Big(\frac{2}{n}\big\|\boldsymbol{\varepsilon}^T\boldsymbol{X}\big\|_\infty > 2\sigma\sqrt{\frac{t^2+2\log(p)}{n}}\Big) \\
&= P\Big(\max_{1\leq j\leq p}|V_j| > \sqrt{t^2+2\log(p)}\Big) \\
&\leq \sum_{j=1}^{p} P\big(|V_j| > \sqrt{t^2+2\log(p)}\big) \\
&\leq 2p\exp\Big(-\frac{t^2+2\log(p)}{2}\Big) = 2\exp(-t^2/2),
\end{aligned}$$

where the last inequality is obtained using the classical result $P(|V_j| > c) \leq 2\exp(-c^2/2)$ for any constant $c > 0$ (see `Problem Sheets`). Hence, $P(\mathscr{F}) \geq 1 - 2\exp(-t^2/2)$. □

We now apply Lemmas 2.5 and 2.6 to establish the consistency of the lasso estimator.

THEOREM 2.7. *(Consistency of the lasso solution) Assume the errors are Gaussian as in Lemma 2.6 and furthermore* $\hat{\sigma}_j^2 = 1$ *for all* $j = 1, \ldots, p$. *For some* $t > 0$, *let the regularisation parameter be* $\lambda = 4\hat{\sigma}\sqrt{\frac{t^2+2\log(p)}{n}}$ *where* $\hat{\sigma}$ *is some estimator of* σ. *Then, with probability at least* $1 - \alpha$ *where* $\alpha = 2\exp(-t^2/2) + P(\hat{\sigma} \leq \sigma)$, *we have*

$$\frac{2}{n}\big\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\big\|_2^2 \leq 3\lambda\big\|\boldsymbol{\beta}^0\big\|_1^1.$$

PROOF. The proof of this theorem would follow from Lemmas 2.5 and 2.6 and Equation (2.17). It is similar to the proof of Lemma 2.9. See `Problem Sheets` □

Remark 2.8. We draw an important conclusion here that taking the regularisation parameter λ of order $\sqrt{\log(p)/n}$, and assuming that the L_1-norm for the true $\boldsymbol{\beta}^0$ is of

smaller order than $\sqrt{n/\log(p)}$, that is,

$$\|\boldsymbol{\beta}^0\|_1^1 = o\Big(\sqrt{\frac{n}{\log(p)}}\Big),$$

will result in the consistency of the lasso estimator. In other words, under these conditions, for the lasso estimator we have

$$\frac{1}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\|_2^2 \xrightarrow{P} 0 \quad \text{as } p > n \to \infty.$$

Note that for the probability of convergence we need a well-chosen estimator $\hat{\sigma}$ of σ, i.e., not too small but also not too large. As a simple choice, one can consider the estimator $\hat{\sigma}^2 = \boldsymbol{Y}^T\boldsymbol{Y}/n$ (after centring $\boldsymbol{Y}$).

We now try to obtain a more refined oracle inequality for the consistency of the lasso estimator under some further conditions. To exploit the sparsity of $\boldsymbol{\beta}^0$, we here introduce some more notation. Let us write $\boldsymbol{\beta}_S$ for any index set $S \subset \{1,\ldots,p\}$, with its j-th element defined as

$$[\boldsymbol{\beta}_S]_j := \boldsymbol{\beta}_j 1(j \in S) = \begin{cases} \boldsymbol{\beta}_j & \text{if } j \in S \\ 0 & \text{if } j \notin S, \end{cases}$$

and similarly for the complement S^c of S

$$[\boldsymbol{\beta}_{S^c}]_j := \boldsymbol{\beta}_j 1(j \notin S) = \begin{cases} 0 & \text{if } j \in S \\ \boldsymbol{\beta}_j & \text{if } j \notin S. \end{cases}$$

Thus, $\boldsymbol{\beta}_S$ has zeroes outside the set S, and the elements of $\boldsymbol{\beta}_{S^c}$ can only be non-zero in S^c. Clearly, $\boldsymbol{\beta} = \boldsymbol{\beta}_S + \boldsymbol{\beta}_{S^c}$. The next lemma is a starting point for the deterministic part of the problem.

LEMMA 2.9. *We have on $\mathcal{F}$, with $\lambda \geq 2\lambda_0$,*

$$\frac{2}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\|_2^2 + \lambda\|\hat{\boldsymbol{\beta}}_{S_0^c}\|_1^1 \leq 3\lambda\|\hat{\boldsymbol{\beta}}_{S_0} - \boldsymbol{\beta}^0_{S_0}\|_1^1,$$

where we recall S_0 is the active set.

PROOF. First, using the basic inequality in Lemma 2.5, we can write

$$\frac{2}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\|_2^2 + 2\lambda\|\hat{\boldsymbol{\beta}}\|_1^1 \leq \frac{4}{n}\boldsymbol{\varepsilon}^T\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0) + 2\lambda\|\boldsymbol{\beta}^0\|_1^1,$$

which, using inequality (2.17), can be written as

$$\frac{2}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2+2\lambda\|\hat{\boldsymbol{\beta}}\|_1^1\leq\frac{4}{n}\|\boldsymbol{\varepsilon}^T\boldsymbol{X}\|_\infty\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_1^1+2\lambda\|\boldsymbol{\beta}^0\|_1^1.$$

On $\mathscr{F}$, this becomes

$$\frac{2}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2+2\lambda\|\hat{\boldsymbol{\beta}}\|_1^1\leq 2\lambda_0\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_1^1+2\lambda\|\boldsymbol{\beta}^0\|_1^1.$$

Then, using $2\lambda_0\leq\lambda$ we get

$$\frac{2}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2+2\lambda\|\hat{\boldsymbol{\beta}}\|_1^1\leq\lambda\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_1^1+2\lambda\|\boldsymbol{\beta}^0\|_1^1. \tag{2.18}$$

Now on the left-hand side of (2.18), using the triangle inequality, we find

$$\|\hat{\boldsymbol{\beta}}\|_1^1=\|\hat{\boldsymbol{\beta}}_{S_0}\|_1^1+\|\hat{\boldsymbol{\beta}}_{S_0^c}\|_1^1\geq\|\boldsymbol{\beta}^0_{S_0}\|_1^1-\|\hat{\boldsymbol{\beta}}_{S_0}-\boldsymbol{\beta}^0_{S_0}\|_1^1+\|\hat{\boldsymbol{\beta}}_{S_0^c}\|_1^1, \tag{2.19}$$

whereas on the right-hand side, we find (note that $\boldsymbol{\beta}^0_{S_0^c}=\mathbf{0}$)

$$\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_1^1=\|\hat{\boldsymbol{\beta}}_{S_0}-\boldsymbol{\beta}^0_{S_0}\|_1^1+\|\hat{\boldsymbol{\beta}}_{S_0^c}\|_1^1. \tag{2.20}$$

Plugging in (2.19) and (2.20) into (2.18) and considering the fact $\|\boldsymbol{\beta}^0\|_1^1=\|\boldsymbol{\beta}^0_{S_0}\|_1^1$, we obtain that

$$\frac{2}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2+\lambda\|\hat{\boldsymbol{\beta}}_{S_0^c}\|_1^1\leq 3\lambda\|\hat{\boldsymbol{\beta}}_{S_0}-\boldsymbol{\beta}^0_{S_0}\|_1^1,$$

which completes the proof. □

As we are working with high dimensional settings, we need certain conditions on the design matrix $\boldsymbol{X}$ to make the theory work. These conditions will be referred to as "compatibility conditions", as they require a certain compatibility of L_1-norms with L_2-norms. In fact, a compatibility condition is just simply an assumption that makes the proof go through.

In Lemma 2.9, a term involving the L_1-norm $\|\hat{\boldsymbol{\beta}}_{S_0}-\boldsymbol{\beta}^0_{S_0}\|_1^1$ occurs on the right-hand side. To get rid of it, we want to somehow incorporate it into the term $\frac{2}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2$ on the left-hand side. By using the properties of norms (see Appendix A), we can replace the L_1-norm by the L_2-norm, paying a price $\sqrt{s_0}$ (recall s_0 is the sparsity index):

$$\|\hat{\boldsymbol{\beta}}_{S_0}-\boldsymbol{\beta}^0_{S_0}\|_1^1\leq\sqrt{s_0}\|\hat{\boldsymbol{\beta}}_{S_0}-\boldsymbol{\beta}^0_{S_0}\|_2^1 \tag{2.21}$$

We are now in the L_2-world, where we wish to relate $\|\hat{\boldsymbol{\beta}}_{S_0} - \boldsymbol{\beta}^0_{S_0}\|_2^1$ to $\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\|_2^1$. Recall the (scaled by $1/n$) sample covariance matrix $\boldsymbol{S}_n = \boldsymbol{X}^T\boldsymbol{X}/n$ so that

$$\frac{1}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\|_2^2 = (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)^T\boldsymbol{S}_n(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0).$$

If for some constant $\phi_0 > 0$

$$\|\hat{\boldsymbol{\beta}}_{S_0} - \boldsymbol{\beta}^0_{S_0}\|_2^2 \le (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)^T\boldsymbol{S}_n(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)/\phi_0^2 = \|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\|_2^2/(n\phi_0^2), \tag{2.22}$$

we can continue our chain of inequalities in a desirable way. However, as $\hat{\boldsymbol{\beta}}$ is random, we need inequalities for a whole class of $\boldsymbol{\beta}$'s. It may be too rough to require (2.22) for all $\boldsymbol{\beta}$, as this needs $\boldsymbol{S}_n$ to be non-singular (which is of course troublesome in high dimensional settings with $p \gg n$, where $\boldsymbol{S}_n$ is always singular). But, we know from Lemma 2.9, that on $\mathscr{F}$,

$$\|\hat{\boldsymbol{\beta}}_{S_0^c}\|_1^1 \le 3\|\hat{\boldsymbol{\beta}}_{S_0} - \boldsymbol{\beta}^0_{S_0}\|_1^1. \tag{2.23}$$

Thus we may restrict ourselves to such $\boldsymbol{\beta}$ (provided that $\mathscr{F}$ has large probability). The compatibility condition is exactly this.

Definition 2.10. (Compatibility condition) We say the compatibility condition is met for the active set S_0, if for some $\phi_0 > 0$, and for all $\boldsymbol{\beta}$ satisfying $\|\boldsymbol{\beta}_{S_0^c}\|_1^1 \le 3\|\boldsymbol{\beta}_{S_0}\|_1^1$, it holds that

$$\|\boldsymbol{\beta}_{S_0}\|_1^2 \le (\boldsymbol{\beta}^T\boldsymbol{S}_n\boldsymbol{\beta})s_0/\phi_0^2. \tag{2.24}$$

We remark that the constant 3 appearing in this definition is quite arbitrary. It can be replaced by any constant bigger than 1, when we adjust some other constants (in particular in the lower bound for λ).

A natural question is: when does this compatibility condition hold? We first recall that if in (2.24) we replace $\|\boldsymbol{\beta}_{S_0}\|_1^2$ by its upper bound $s_0\|\boldsymbol{\beta}_{S_0}\|_2^2$, the condition is similar to a condition on the smallest eigenvalue of $\boldsymbol{S}_n$. But the restriction $\|\boldsymbol{\beta}_{S_0^c}\|_1^1 \le 3\|\boldsymbol{\beta}_{S_0}\|_1^1$ puts a limitation on the set of $\boldsymbol{\beta}$'s for which (2.24) is required, so that it is in fact weaker than imposing non-zero eigenvalues. Note further that this cannot be checked in practice since S_0 is unknown. Appreciating this limitation, we present an oracle inequality in the following theorem.

THEOREM 2.11. *Suppose that the compatibility condition holds for S_0. Then on $\mathscr{F}$, we have for $\lambda \geq 2\lambda_0$,*

$$\frac{1}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2+\lambda\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_1^1 \leq 4\lambda^2 s_0/\phi_0^2.$$

PROOF. First, using (2.20) in the proof of Lemma 2.9, we can write

$$\begin{aligned}\frac{2}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2+\lambda\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_1^1 &= \frac{2}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2+\lambda\|\hat{\boldsymbol{\beta}}_{S_0}-\boldsymbol{\beta}^0_{S_0}\|_1^1+\lambda\|\hat{\boldsymbol{\beta}}_{S_0^c}\|_1^1\\ &\leq 4\lambda\|\hat{\boldsymbol{\beta}}_{S_0}-\boldsymbol{\beta}^0_{S_0}\|_1^1,\end{aligned}$$

where the last inequality is obtained from the result of Lemma 2.9, that is, $\frac{2}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2+\lambda\|\hat{\boldsymbol{\beta}}_{S_0^c}\|_1^1 \leq 3\lambda\|\hat{\boldsymbol{\beta}}_{S_0}-\boldsymbol{\beta}^0_{S_0}\|_1^1$.
Using (2.21) and (2.22) we get

$$4\lambda\|\hat{\boldsymbol{\beta}}_{S_0}-\boldsymbol{\beta}^0_{S_0}\|_1^1 \leq 4\lambda\sqrt{s_0}\|\hat{\boldsymbol{\beta}}_{S_0}-\boldsymbol{\beta}^0_{S_0}\|_2^1 \leq 4\lambda\sqrt{s_0}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^1/(\sqrt{n}\phi_0).$$

Hence,

$$\begin{aligned}\frac{2}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2+\lambda\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_1^1 &\leq 4\lambda\sqrt{s_0}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^1/(\sqrt{n}\phi_0)\\ &\leq \frac{1}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2+4\lambda^2 s_0/\phi_0^2.\end{aligned}$$

where the last inequality is obtained from the inequality $4ab \leq a^2+4b^2$ where we used $a=\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^1/\sqrt{n}$ and $b=\lambda\sqrt{s_0}/\phi_0$. This completes the proof. □

Remark 2.12. Theorem 2.11 combines two results. Firstly, it shows the following upper bound for the "prediction error"

$$\frac{1}{n}\|\boldsymbol{X}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)\|_2^2 \leq 4\lambda^2 s_0/\phi_0^2.$$

And secondly, it gives the following upper bound for the "estimation error"

$$\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_1^1 \leq 4\lambda s_0/\phi_0^2.$$

Combining Theorem 2.11 with the probabilistic statement of Lemma 2.6 to handle the set $\mathscr{F}$, we get the following corollary.

COROLLARY 2.13. *(Consistency of lasso solution with compatibility condition) Assume the errors are Gaussian and that $\hat{\sigma}_j^2=1$ for all $j=1,\ldots,p$, and further the compatibility condition holds for S_0, with the scaled sample covariance matrix $\boldsymbol{S}_n$. For some $t>0$, let*

the regularisation parameter be $\lambda = 4\hat{\sigma}\sqrt{\frac{t^2+2\log(p)}{n}}$ *where* $\hat{\sigma}^2$ *is an estimator of the error variance* σ^2*. Then, with probability at least* $1-\alpha$ *where* $\alpha = 2\exp(-t^2/2) + P(\hat{\sigma} \le \sigma)$*, we have*

$$\frac{1}{n}\left\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\right\|_2^2 + \lambda\left\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\right\|_1^1 \le 4\lambda^2 s_0/\phi_0^2.$$

Remark 2.14. The conclusion here is that by taking the regularisation parameter λ of order $\sqrt{\log(p)/n}$ and assuming the compatibility condition, we have

$$\frac{1}{n}\left\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\right\|_2^2 = O_P\Big(\frac{s_0\log(p)}{n\phi_0^2}\Big).$$

So if $s_0 < \frac{n}{\log(p)}$ as $p > n \to \infty$, we have

$$\frac{1}{n}\left\|\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)\right\|_2^2 \xrightarrow{P} 0 \quad \text{as } p > n \to \infty.$$

Note that, unlike Theorem 2.7, the above consistency of lasso with the compatibility condition provides upper bound for both the estimation error and the prediction error, as discussed in Remark 2.12.

2.6. Regularisation with elastic net

We have seen in Section 2.4 that the ridge regression does not provide variable selection (i.e., no sparse solution). This because the L_2-ball is smooth. We have also seen in Section 2.5 that the variable selection property of the lasso estimator is induced by the non-smoothness of the L_1-ball.

The elastic net estimator, which was proposed by Zou and Hastie [32], involves both L_1 and L_2 penalties. It is suggested to improve the lasso estimator in situations when the covariates, or the columns of design matrix $\boldsymbol{X}$, are strongly correlated. The elastic net estimator is defined as follows

$$\hat{\boldsymbol{\beta}}_{\text{EN}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\Big\{\left\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\right\|_2^2 + \lambda_1\left\|\boldsymbol{\beta}\right\|_1^1 + \lambda_2\left\|\boldsymbol{\beta}\right\|_2^2\Big\}, \tag{2.25}$$

where $\lambda_1, \lambda_2 \ge 0$ are two separate regularisation parameters. It can be seen that the elastic net regularisation method linearly combines the L1 and L2 penalties of the lasso and ridge methods.

The actual reason for adding an additional L_2 penalty to the lasso is motivated by Zou and Hastie [32] as follows. For strongly correlated covariates, the lasso may select one but typically not both of them (and the non-selected variable can then be approximated as a linear function of the selected one). From the point of view of sparsity, this is what we would like to do. However, in terms of interpretation, we may want to have two even strongly correlated variables among the selected variables: this is motivated by the idea that we do not want to miss a "true" variable due to selection of a "non-true" which is highly correlated with the true one. Also, from the prediction point of view, there can be gains as well by using the elastic net in comparison to the lasso (e.g., Hebiri and van de Geer [14]).

The elastic net problem is also convex (though non-differentiable), so the computation of elastic net estimator can be done similarly to the lasso by using the gradient descend algorithm. We here skip the technical details of the computation, but we note that this is also implemented in the R package `glmnet`.

CHAPTER 3

Principal component analysis for high dimensional data

3.1. Introduction

Principal component analysis (PCA) is a useful tool for dimensionality reduction in high dimensional data which can be used to reduce a large set of variables to a smaller set that still contains the most of information (variation) in the large set. PCA is also useful for visualisation purposes as a result of dimensionality reduction. In certain applications, PCA is used to transform a set of correlated variables into uncorrelated variables. In fact, principal components are linearly uncorrelated by construction, which is a desirable property. Remarkably, PCA dates back to a paper by Karl Pearson in 1901 (Pearson [25]). In this chapter, we will see that PCA is a projection method that projects a high dimensional data set to a lower dimensional data set while most of the information or variation in data is preserved.

We first present the concept of PCA for low dimensional data. Let $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n$ be n random observations each p-dimensional (i.e., n data points on p variables). These observations could be potentially correlated, so the PCA does not require independency assumption of data. Let $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1^T \\ \vdots \\ \boldsymbol{X}_n^T \end{bmatrix}$ denote the $n \times p$ matrix of the entire data, with each column being one of the p variables. In PCA, it is often needed to centre the data so that each column of the data matrix, which represents one of the p variables, has zero mean. This is easily done by subtracting each column from its mean, as also discussed in the sequel.

For simplicity of presentation, we first show the derivation of the first principal component, which represents the direction of largest variance in data. The following theorem addresses this.

THEOREM 3.1. *For the $n \times p$ data matrix $\boldsymbol{X}$ with all its columns being centred at zero, the first principal component is $\boldsymbol{X}\boldsymbol{v}_1$ in which $\boldsymbol{v}_1$ is the eigenvector corresponding to the largest eigenvalue of matrix $\boldsymbol{X}^T\boldsymbol{X}$.*

PROOF. We need to find the direction $\boldsymbol{v}_1$ that maximises the variance of $\boldsymbol{X}\boldsymbol{v}_1$. For this, we write

$$\max_{\boldsymbol{v}_1}\Big\{\text{Var}(\boldsymbol{X}\boldsymbol{v}_1)\Big\} = \max_{\boldsymbol{v}_1}\Big\{\boldsymbol{v}_1^T\text{Var}(\boldsymbol{X})\boldsymbol{v}_1\Big\} = \max_{\boldsymbol{v}_1}\Big\{\boldsymbol{v}_1^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v}_1\Big\},$$

where we also need to account for the constraint $\boldsymbol{v}_1^T\boldsymbol{v}_1 = 1$ that enforces the length of $\boldsymbol{v}_1$ to be 1. For this, using the Lagrange method we maximise

$$L = \boldsymbol{v}_1^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v}_1 - \lambda_1(\boldsymbol{v}_1^T\boldsymbol{v}_1 - 1).$$

Since $\frac{\partial L}{\partial \boldsymbol{v}_1} = 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v}_1 - 2\lambda_1\boldsymbol{v}_1 = 0$, we get

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v}_1 = \lambda_1\boldsymbol{v}_1,$$

which shows that $\boldsymbol{v}_1$ and λ_1 are eigenvector and eigenvalue of $\boldsymbol{X}^T\boldsymbol{X}$. Now, since $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v}_1 = \lambda_1\boldsymbol{v}_1$, we get

$$\boldsymbol{v}_1^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v}_1 = \boldsymbol{v}_1^T\lambda_1\boldsymbol{v}_1 = \lambda_1\boldsymbol{v}_1^T\boldsymbol{v}_1 = \lambda_1.$$

Hence, to maximise $\text{Var}(\boldsymbol{X}\boldsymbol{v}_1)$ we must choose the eigenvector corresponding to the largest eigenvalue of $\boldsymbol{X}^T\boldsymbol{X}$. □

Similar to the above theorem, one can show that the p principal components of $\boldsymbol{X}$ are obtained using the p eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$, which we here denote by $\boldsymbol{v}_1,\ldots,\boldsymbol{v}_p$. Note that because $\boldsymbol{X}^T\boldsymbol{X}$ is a $p \times p$ covariance matrix, it has at most p eigenvalues and p eigenvectors, which can be written as

$$\begin{array}{cccc} \lambda_1 > \lambda_2 > & \ldots & > \lambda_p \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 & \ldots & \boldsymbol{v}_p \\ \downarrow & \downarrow & \ldots & \downarrow \\ PC_1 & PC_2 & \ldots & PC_p \end{array}$$

The proportion of variation of the first d principal components to the total variation is

$$\mathcal{V}_d = \frac{\sum_{j=1}^{d}\lambda_j}{\sum_{j=1}^{p}\lambda_j}.$$

For dimensionality reduction purposes, if there is a small d, $d < p$, such that $\mathcal{V}_d \approx 1$ or $\mathcal{V}_d$ is large enough then the dimension of the data matrix $\boldsymbol{X}$ can be successfully reduced from p to d.

Example:

We consider a real data set, called `fat`, containing the dimensions (10 variables) of the human body as measured in a study on 252 men (Faraway [8]).

The R code, including loading the data set, and the results on PCA are given below.

```
> data(fat,package="faraway")
> cfat <- fat[,9:18]
> prfat <- prcomp(cfat)
> summary(prfat)
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6
Standard deviation     15.990 4.06584 2.96596 2.00044 1.69408 1.49881
Proportion of Variance  0.867 0.05605 0.02983 0.01357 0.00973 0.00762
Cumulative Proportion   0.867 0.92304 0.95287 0.96644 0.97617 0.98378
                           PC7     PC8     PC9    PC10
Standard deviation     1.30322 1.25478 1.10955 0.52737
Proportion of Variance 0.00576 0.00534 0.00417 0.00094
Cumulative Proportion  0.98954 0.99488 0.99906 1.00000
```

It can be seen that about 92% of the variation in the data can be explained by the first two principal components, which is a remarkable result for dimensionality reduction.

3.2. Dimension reduction and high dimensional PCA

When faced with a high dimensional data set, it is often needed to reduce its dimension, either by projecting it to a lower dimension space or by finding a better representation for the data. Indeed, PCA continues to be one of the useful tools for dimensionality reduction. Recall we wish to (linearly) project the data to d dimensions ($d < p$). This is particularly useful if one wants to visualise the data in two or three dimensions. We here present two different ways we can try to choose this projection:

(1) Finding a d-dimensional projection of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ that preserves as much variance of the data as possible (statistical formulation).

(2) Finding a d-dimensional affine subspace for which the projections of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ on it best approximate the original points $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ (geometric formulation).

Interestingly, we will see that these two approaches are actually "equivalent" corresponding to PCA. We start with the first interpretation of PCA and then show that it is equivalent to the second interpretation.

3.2.1. PCA as d-dimensional projection that preserves the most variance

As before, the aim is to find an orthonormal basis $\boldsymbol{v}_1,\dots,\boldsymbol{v}_d$ of a d-dimensional space such that the projection of $\boldsymbol{X}_1,\dots,\boldsymbol{X}_n$ projected on this subspace has the most variance. Let $\boldsymbol{V}=[\boldsymbol{v}_1,\dots,\boldsymbol{v}_d]\in\mathbb{R}^{p\times d}$ with $\boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}_d$. Equivalently we here look for the projected points $\boldsymbol{X}_1^T\boldsymbol{V},\dots,\boldsymbol{X}_n^T\boldsymbol{V}$ to have as much variance as possible. Note that the $n\times d$ matrix $\boldsymbol{XV}=\begin{bmatrix}\boldsymbol{X}_1^T\boldsymbol{V}\\ \vdots\\ \boldsymbol{X}_n^T\boldsymbol{V}\end{bmatrix}$ is then the lower dimensional projection of the $n\times p$ data matrix $\boldsymbol{X}$, which we are trying to obtain with PCA. Here, we are interested in solving

$$\max_{\substack{\boldsymbol{V}\\ \boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}_d}} \sum_{i=1}^n \left\|\boldsymbol{X}_i^T\boldsymbol{V}-\frac{1}{n}\sum_{r=1}^n \boldsymbol{X}_r^T\boldsymbol{V}\right\|_2^2. \tag{3.1}$$

Note that we here study the general case without having to centre variables. Let us define the sample mean and the sample covariance of observations as

$$\boldsymbol{\mu}_n=\frac{1}{n}\sum_{r=1}^n \boldsymbol{X}_r,$$

and

$$\boldsymbol{\Sigma}_n=\frac{1}{n-1}\sum_{r=1}^n(\boldsymbol{X}_r-\boldsymbol{\mu}_n)(\boldsymbol{X}_r-\boldsymbol{\mu}_n)^T.$$

We then find

$$\sum_{i=1}^n \left\|\boldsymbol{X}_i^T\boldsymbol{V}-\frac{1}{n}\sum_{r=1}^n \boldsymbol{X}_r^T\boldsymbol{V}\right\|_2^2=\sum_{i=1}^n \left\|(\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T\boldsymbol{V}\right\|_2^2=\sum_{i=1}^n(\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T\boldsymbol{V}\boldsymbol{V}^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n).$$

Applying a few simple algebraic calculations using properties of the trace operator gives us

$$\begin{aligned}\sum_{i=1}^n(\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T\boldsymbol{V}\boldsymbol{V}^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n)&=\sum_{i=1}^n \mathrm{tr}\Big((\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T\boldsymbol{V}\boldsymbol{V}^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n)\Big)\\ &=\sum_{i=1}^n \mathrm{tr}\Big(\boldsymbol{V}^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n)\boldsymbol{V}\Big)\\ &=\mathrm{tr}\Big(\boldsymbol{V}^T\sum_{i=1}^n(\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n)\boldsymbol{V}\Big)\\ &=(n-1)\mathrm{tr}\big(\boldsymbol{V}^T\boldsymbol{\Sigma}_n\boldsymbol{V}\big).\end{aligned}$$

Hence, by ignoring the constant $n-1$, solving (3.1) leads to

$$\begin{aligned}\max_{\substack{\boldsymbol{V}\\ \boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}_d}} \sum_{i=1}^{n} \left\|\boldsymbol{X}_i^T\boldsymbol{V} - \frac{1}{n}\sum_{r=1}^{n}\boldsymbol{X}_r^T\boldsymbol{V}\right\|_2^2 &= \max_{\substack{\boldsymbol{V}\\ \boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}_d}} \operatorname{tr}\left(\boldsymbol{V}^T\boldsymbol{\Sigma}_n\boldsymbol{V}\right)\\ &= \max_{\substack{\boldsymbol{v}_1,\dots,\boldsymbol{v}_d\in\mathbb{R}^p\\ \boldsymbol{v}_i^T\boldsymbol{v}_j=\delta_{ij}}} \sum_{j=1}^{d}\boldsymbol{v}_j^T\boldsymbol{\Sigma}_n\boldsymbol{v}_j,\end{aligned} \tag{3.2}$$

where δ_{ij} is the Kronecker delta which is 1 if $i=j$ and 0 if $i\neq j$.

When $d=1$, as in the case for the first principal component, equation (3.2) reduces to

$$\max_{\substack{\boldsymbol{v}_1\in\mathbb{R}^p\\ \boldsymbol{v}_1^T\boldsymbol{v}_1=1}} \boldsymbol{v}_1^T\boldsymbol{\Sigma}_n\boldsymbol{v}_1,$$

which is a more familiar problem. We know, from Theorem 3.1, that this is maximised by the eigenvector corresponding to the largest eigenvalue of $\boldsymbol{\Sigma}_n$.

In general, it is not very difficult to show (it follows from a Theorem of Fan, see for example page 3 of Moslehian [21]) that equation (3.2) is maximised by taking $\boldsymbol{v}_1,\dots,\boldsymbol{v}_d$ to be the d leading eigenvectors (i.e. corresponding to the d largest eigenvalues) of $\boldsymbol{\Sigma}_n$, and that its value is simply the sum of the d largest eigenvalues of $\boldsymbol{\Sigma}_n$. The nice consequence of this is that the solution to (3.2) can be computed sequentially: we first solve for $d=1$, computing $\boldsymbol{v}_1$, then $\boldsymbol{v}_2$, and so on.

3.2.2. PCA as best d-dimensional affine fit

A second approach is to try to approximate each $\boldsymbol{X}_i$ using an affine function as follows

$$\boldsymbol{X}_i \approx \boldsymbol{\mu} + \sum_{j=1}^{d}[\boldsymbol{\beta}_i]_j\boldsymbol{v}_j,$$

where $\boldsymbol{v}_1,\dots,\boldsymbol{v}_d$ is an orthonormal basis for the d-dimensional subspace, $\boldsymbol{\mu}\in\mathbb{R}^p$ represents the translation, and $\boldsymbol{\beta}_i\in\mathbb{R}^d$ denotes the coefficients for approximating $\boldsymbol{X}_i$. Recalling $\boldsymbol{V}=[\boldsymbol{v}_1,\dots,\boldsymbol{v}_d]\in\mathbb{R}^{p\times d}$, the above approximation can be rewritten in the following matrix form

$$\boldsymbol{X}_i \approx \boldsymbol{\mu} + \boldsymbol{V}\boldsymbol{\beta}_i,$$

where we remind that $\boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}_d$ as vectors $\boldsymbol{v}_1,\dots,\boldsymbol{v}_d$ are orthonormal. Note that we need the constraint $\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\beta}_i=0$ to ensure a unique solution to the problem of finding an affine subspace for the data.

We measure the goodness-of-fit in terms of least squares error and try to solve

$$\min_{\substack{\boldsymbol{\mu},\boldsymbol{V},\boldsymbol{\beta}_i \\ \boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}_d}} \sum_{i=1}^{n} \left\|\boldsymbol{X}_i - (\boldsymbol{\mu}+\boldsymbol{V}\boldsymbol{\beta}_i)\right\|_2^2. \tag{3.3}$$

We start by optimising for $\boldsymbol{\mu}$. It is easy to see that the first order conditions for $\boldsymbol{\mu}$ correspond to

$$\frac{\partial}{\partial \boldsymbol{\mu}} \sum_{i=1}^{n} \left(\left\|\boldsymbol{X}_i - (\boldsymbol{\mu}+\boldsymbol{V}\boldsymbol{\beta}_i)\right\|_2^2\right) = 0 \iff \sum_{i=1}^{n} \left(\boldsymbol{X}_i - (\boldsymbol{\mu}+\boldsymbol{V}\boldsymbol{\beta}_i)\right) = 0.$$

Thus, the optimal value $\boldsymbol{\mu}^*$ of $\boldsymbol{\mu}$ satisfies

$$\left(\sum_{i=1}^{n} \boldsymbol{X}_i\right) - n\boldsymbol{\mu}^* - \boldsymbol{V}\left(\sum_{i=1}^{n} \boldsymbol{\beta}_i\right) = 0,$$

and hence, since $\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\beta}_i = 0$, we get

$$\boldsymbol{\mu}^* = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{X}_i = \boldsymbol{\mu}_n.$$

So the optimal value of $\boldsymbol{\mu}$ is the sample mean $\boldsymbol{\mu}_n$.

We then proceed on finding the solution for (3.3) by solving

$$\min_{\substack{\boldsymbol{V},\boldsymbol{\beta}_i \\ \boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}_d}} \sum_{i=1}^{n} \left\|\boldsymbol{X}_i - \boldsymbol{\mu}_n - \boldsymbol{V}\boldsymbol{\beta}_i\right\|_2^2. \tag{3.4}$$

Let us proceed by optimising for $\boldsymbol{\beta}_i$. Since the problem decouples for each i, we can focus on, for each i,

$$\min_{\boldsymbol{\beta}_i} \left\|\boldsymbol{X}_i - \boldsymbol{\mu}_n - \boldsymbol{V}\boldsymbol{\beta}_i\right\|_2^2 = \min_{\boldsymbol{\beta}_i} \left\|\boldsymbol{X}_i - \boldsymbol{\mu}_n - \sum_{j=1}^{d} [\boldsymbol{\beta}_i]_j \boldsymbol{v}_j\right\|_2^2.$$

Since $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d$ are orthonormal, it is easy to see that the solution of this is given by

$$[\boldsymbol{\beta}_i^*]_j = \boldsymbol{v}_j^T(\boldsymbol{X}_i - \boldsymbol{\mu}_n),$$

or equivalently we have (see `Problem Sheets`)

$$\boldsymbol{\beta}_i^* = \boldsymbol{V}^T(\boldsymbol{X}_i - \boldsymbol{\mu}_n).$$

Hence, (3.4) is equivalent to

$$\min_{\substack{\boldsymbol{V} \\ \boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}_d}} \sum_{i=1}^{n} \left\|(\boldsymbol{X}_i - \boldsymbol{\mu}_n) - \boldsymbol{V}\boldsymbol{V}^T(\boldsymbol{X}_i - \boldsymbol{\mu}_n)\right\|_2^2. \tag{3.5}$$

Direct calculations give

$$\begin{aligned}\left\|(\boldsymbol{X}_i-\boldsymbol{\mu}_n)-\boldsymbol{V}\boldsymbol{V}^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n)\right\|_2^2 &= (\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n)\\ &\quad -2(\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T\boldsymbol{V}\boldsymbol{V}^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n)\\ &\quad +(\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{V}^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n)\\ &= (\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n)\\ &\quad -(\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T\boldsymbol{V}\boldsymbol{V}^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n).\end{aligned}$$

Since $(\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n)$ does not depend on $\boldsymbol{V}$, minimising (3.5) is equivalent to

$$\max_{\substack{\boldsymbol{V}\\ \boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}_d}} \sum_{i=1}^{n}(\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T\boldsymbol{V}\boldsymbol{V}^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n). \tag{3.6}$$

As shown in the previous case, we have

$$\sum_{i=1}^{n}(\boldsymbol{X}_i-\boldsymbol{\mu}_n)^T\boldsymbol{V}\boldsymbol{V}^T(\boldsymbol{X}_i-\boldsymbol{\mu}_n) = (n-1)\mathrm{tr}\big(\boldsymbol{V}^T\boldsymbol{\Sigma}_n\boldsymbol{V}\big).$$

This then means that the solution to (3.6) is given by

$$\max_{\substack{\boldsymbol{V}\\ \boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}_d}} \mathrm{tr}\big(\boldsymbol{V}^T\boldsymbol{\Sigma}_n\boldsymbol{V}\big), \tag{3.7}$$

which is the same optimisation problem as (3.2). Therefore, the second interpretation of PCA is equivalent to the first one presented in the previous subsection.

Remark 3.2. To summarise our dimensionality reduction with PCA in high dimensions, when we have an $n\times p$ data matrix $\boldsymbol{X}$ with p variables as columns, the reduced data using PCA is the $n\times d$ matrix $\boldsymbol{X}\boldsymbol{V}$ with $\boldsymbol{V}=[\boldsymbol{v}_1,\dots,\boldsymbol{v}_d]\in\mathbb{R}^{p\times d}$ being the d leading eigenvectors of the sample covariance $\boldsymbol{\Sigma}_n$. Note that the columns of $\boldsymbol{X}\boldsymbol{V}$ are the principal components. In the following, we explain how to calculate eigenvectors $\boldsymbol{V}$.

Calculating the principal components:
As we have seen, given observations $\boldsymbol{X}_1,\dots,\boldsymbol{X}_n\in\mathbb{R}^p$, in order to compute the principal components we need to calculate the leading eigenvectors of

$$\boldsymbol{\Sigma}_n=\frac{1}{n-1}\sum_{r=1}^{n}(\boldsymbol{X}_r-\boldsymbol{\mu}_n)(\boldsymbol{X}_r-\boldsymbol{\mu}_n)^T.$$

A "naive" way of doing this would be to first calculate the $p\times p$ matrix $\boldsymbol{\Sigma}_n$ whose computational complexity is $O(np^2)$, and then find its spectral decomposition (i.e.,

eigenvalues and eigenvectors) with computational complexity of $O(p^3)$. This means that the computational complexity of this procedure is $O(\max\{np^2, p^3\})$. This would be $O(p^3)$ in high dimensional settings when $n \ll p$. So this can be computationally very expensive when p is large.

A more efficient and practical way is to use the singular value decomposition (SVD), which we saw in the previous chapter. Recall that $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1^T \\ \vdots \\ \boldsymbol{X}_n^T \end{bmatrix}$ denotes the $n \times p$ matrix of the entire data, with $\boldsymbol{X}^T = \left[\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\right]$ being its transpose. The sample covariance $\boldsymbol{\Sigma}_n$ can be rewritten as

$$\boldsymbol{\Sigma}_n = \frac{1}{n-1}\left(\boldsymbol{X} - \mathbf{1}_n \boldsymbol{\mu}_n^T\right)^T \left(\boldsymbol{X} - \mathbf{1}_n \boldsymbol{\mu}_n^T\right),$$

where $\mathbf{1}_n$ is an n-dimensional vector of ones. Let us write the compact SVD of the $n \times p$ matrix $\boldsymbol{X} - \mathbf{1}_n \boldsymbol{\mu}_n^T$, which is $\boldsymbol{X} - \mathbf{1}_n \boldsymbol{\mu}_n^T = \boldsymbol{U}_L \boldsymbol{D} \boldsymbol{U}_R^T$ where $\boldsymbol{U}_L^T \boldsymbol{U}_L = \boldsymbol{I}_r$ with r being the rank of $\boldsymbol{X} - \mathbf{1}_n \boldsymbol{\mu}_n^T$ (typically $r = n$ when $n \ll p$), $\boldsymbol{D} \in \mathbb{R}^{r \times r}$ diagonal, and $\boldsymbol{U}_R \in \mathbb{R}^{p \times r}$. Then,

$$\boldsymbol{\Sigma}_n = \frac{1}{n-1}\left(\boldsymbol{X} - \mathbf{1}_n \boldsymbol{\mu}_n^T\right)^T \left(\boldsymbol{X} - \mathbf{1}_n \boldsymbol{\mu}_n^T\right) = \frac{1}{n-1} \boldsymbol{U}_R \boldsymbol{D} \boldsymbol{U}_L^T \boldsymbol{U}_L \boldsymbol{D} \boldsymbol{U}_R^T = \frac{1}{n-1} \boldsymbol{U}_R \boldsymbol{D}^2 \boldsymbol{U}_R^T,$$

which implies that the columns of $\boldsymbol{U}_R$ are the eigenvectors of $\boldsymbol{\Sigma}_n$. Computing the SVD of $\boldsymbol{X} - \mathbf{1}_n \boldsymbol{\mu}_n^T$ takes $O(nrp)$. And if one is interested in simply computing the d leading eigenvectors ($d < r$), the computational cost reduces to $O(ndp)$. So the computational cost will be reduced drastically compared to the naive approach of $O(p^3)$. This approach is much more efficient in high dimensional settings and is easily implemented in standard software. In particular, the R function `prcomp` uses the SVD for conducting the PCA. We will see examples in our Computer Practical session.

Remark 3.3. In practice, one should centre all the columns of data matrix $\boldsymbol{X}$ to have zero mean (sometimes standardisation may be required particularly when the scales of variables are much different - we will have examples in our Computer Practical session). Once the columns of data matrix $\boldsymbol{X}$ are centred, we can simply apply the SVD on the $n \times p$ centred matrix $\boldsymbol{X}$, say $\boldsymbol{X} = \boldsymbol{U}_L \boldsymbol{D} \boldsymbol{U}_R^T$, and the columns of $\boldsymbol{U}_R$ will be the required eigenvectors for PCA, similar to above.

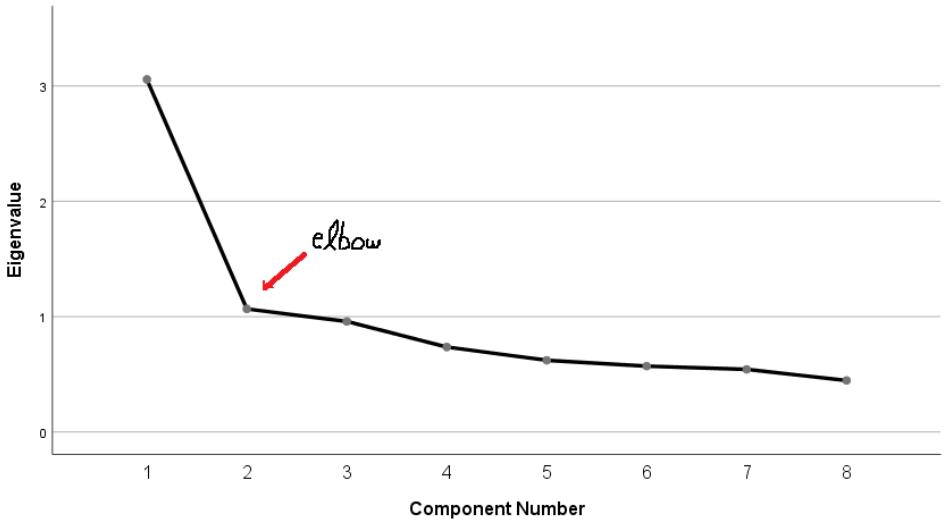


FIGURE 3.1. A scree plot with an "elbow" suggesting a cut-off value.

Which d should we choose?

Given a data set, if the objective is just to visualise the data then picking $d = 2$ or $d = 3$ might make the most sense. However, PCA is useful for many other purposes, in particular: (1) often times the data belongs to a lower dimensional space but is corrupted by high dimensional noise or unimportant variables. When using PCA it is often possible to reduce the noise or the dimension while keeping the signal, and (2) one may be interested in running an algorithm that would be computationally very expensive to run in high dimensions, while the dimension reduction could help there. In these applications (and others) it is not clear how to choose d.

If we denote the k-th largest eigenvalue of $\mathbf{\Sigma}_n$ by λ_k , then the k-th principal component has a $\frac{\lambda_k}{\text{tr}(\mathbf{\Sigma}_n)}$ proportion of the variance. This is what we saw in Section 3.1 (remember the trace of a matrix is the sum of its eigenvalues). A fairly common approach is to try choosing the cut-off at a component that has significantly more variance than the one immediately after. This is usually visualised by a scree plot, which is a plot of the values of the ordered eigenvalues. An example is presented is Figure 3.1.

It is common to identify an "elbow" on the scree plot to choose an appropriate cut-off. In the next section we will look into random matrix theory to try to better understand the behaviour of the eigenvalues of $\mathbf{\Sigma}_n$, which helps us understand when to cut-off.

3.3. PCA in high dimensions and Marchenko-Pastur distribution

Let us here assume that the data points $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n \in \mathbb{R}^p$ are independent draws of a Gaussian distribution $N(\boldsymbol{\mu},\boldsymbol{\Sigma})$ for some covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$. Without loss of generality, we can here assume $\boldsymbol{\mu} = \mathbf{0}$; remember we can simply centre the variables to have zero mean. In this case when we use PCA we hope to find a low dimensional structure in the distribution that should correspond to large eigenvalues of $\boldsymbol{\Sigma}$ (and their corresponding eigenvectors). For this reason and since PCA depends on the spectral properties of sample covariance $\boldsymbol{\Sigma}_n$, we would like to understand whether the spectral properties (i.e., eigenvalues and eigenvectors) of $\boldsymbol{\Sigma}_n$ are close to the ones of $\boldsymbol{\Sigma}$ (i.e., the assumed covariance matrix).

Since $E(\boldsymbol{\Sigma}_n) = \boldsymbol{\Sigma}$ (see `Problem Sheets`), if $n \to \infty$ and p is fixed then the law of large numbers guarantees that indeed $\boldsymbol{\Sigma}_n \xrightarrow{P} \boldsymbol{\Sigma}$. However, in high dimensional settings when $n \ll p$, it is also the case that $p \to \infty$ as well. Unfortunately, in that case or even when p is in the order of n, it is no longer clear that $\boldsymbol{\Sigma}_n \xrightarrow{P} \boldsymbol{\Sigma}$. As before, dealing with this type of difficulties is the realm of high dimensional statistics.

When the data are centred, it is convenient to work with the sample covariance scaled by $1/n$

$$\boldsymbol{S}_n = \frac{1}{n}\sum_{r=1}^{n} \boldsymbol{X}_r \boldsymbol{X}_r^T = \frac{1}{n}\boldsymbol{X}^T\boldsymbol{X},$$

instead of $\boldsymbol{\Sigma}_n$ scaled by $1/(n-1)$. Since $\frac{n}{n-1} \to 1$ as $n \to \infty$ and that $\boldsymbol{\mu}_n \to \mathbf{0}$ as $\boldsymbol{\mu} = \mathbf{0}$, the spectral properties of $\boldsymbol{S}_n$ is essentially the same as $\boldsymbol{\Sigma}_n$ asymptotically. Note that in the multivariate normal distribution $N(\boldsymbol{\mu},\boldsymbol{\Sigma})$ (here $\boldsymbol{\mu} = \mathbf{0}$), it is easy to show that $\boldsymbol{S}_n$ is the maximum likelihood estimator of $\boldsymbol{\Sigma}$. Unlike $\boldsymbol{\Sigma}_n$ which is unbiased for $\boldsymbol{\Sigma}$, $\boldsymbol{S}_n$ is biased in small samples, however they are asymptotically the same as $n \to \infty$.

Using the above assumption that the centred data follow $N(\mathbf{0},\boldsymbol{\Sigma})$, we try to understand the spectral properties of the sample covariance $\boldsymbol{S}_n$. We start by looking into a simple case when $\boldsymbol{\Sigma} = \boldsymbol{I}_p$. In this case, the distribution has no low dimensional structure because the distribution is rotation invariant. Figure 3.2 shows a histogram (left) and a scree plot (right) of the eigenvalues of a simulated sample of $\boldsymbol{S}_n$ (when $\boldsymbol{\Sigma} = \boldsymbol{I}_p$) for $p = 500$ and $n = 1000$. We use a large sample size $n = 1000$ just because of asymptotic

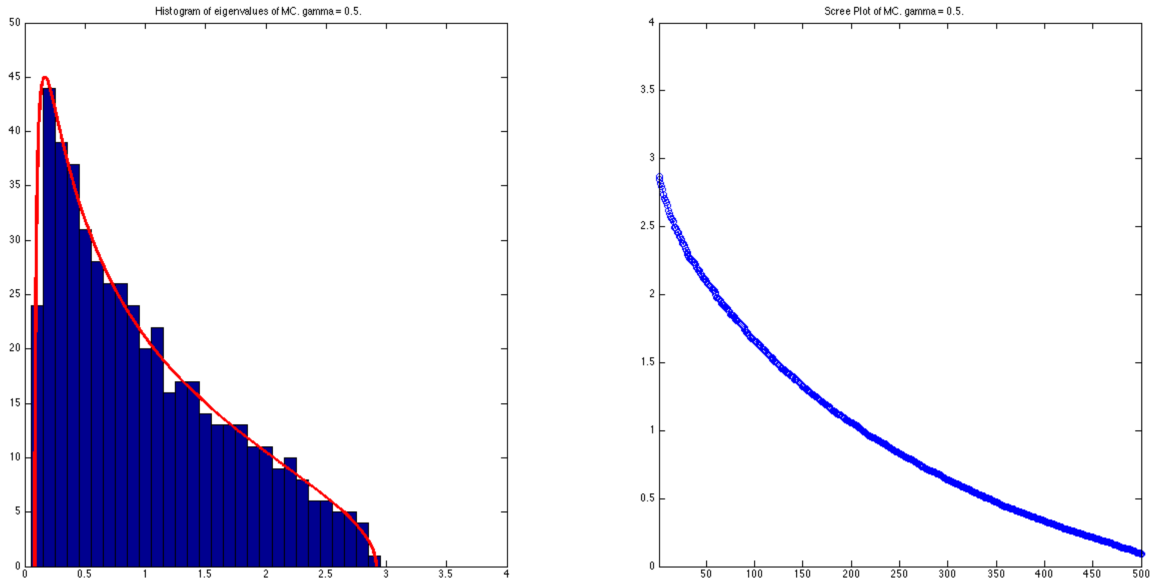


FIGURE 3.2. Histogram (left) and scree plot (right) of the eigenvalues of a simulated sample of $\boldsymbol{S}_n$ (when $\boldsymbol{\Sigma} = \boldsymbol{I}_p$) for $p = 500$ and $n = 1000$.

considerations and comparisons. For instance, the red line is the eigenvalue distribution predicted by the Marchenko-Pastur distribution (3.8), which is asymptotically accurate as we will discuss shortly. From Figure 3.2, it can be seen that there are many eigenvalues considerably larger than 1 (and some considerably larger than others). Notice that given this profile of eigenvalues of $\boldsymbol{S}_n$ one could potentially be led to believe that the data has low dimensional structure, while in truth the distribution it was drawn from is isotropic.

Marchenko-Pastur distribution of eigenvalues:
Understanding the distribution of eigenvalues of random matrices is the core of Random Matrix Theory (there are many useful books on Random Matrix Theory, e.g. Tao [28] and Anderson et al. [1]). A particular limiting distribution was first established in 1967 by Marchenko and Pastur (Marchenko and Pastur [20]) and is now referred to as the Marchenko-Pastur distribution. They showed that, if p and n both go to ∞ with their ratio being fixed as $p/n = \gamma$, the sample distribution of the eigenvalues of $\boldsymbol{S}_n$ (like the histogram above), in the limit, will be

$$f_\gamma(\lambda) = \frac{1}{2\pi} \frac{\sqrt{(\gamma_+ - \lambda)(\lambda - \gamma_-)}}{\gamma\lambda} 1_{[\gamma_-, \gamma_+]}(\lambda) \tag{3.8}$$

with the support $[\gamma_-, \gamma_+]$, where $\gamma_- = (1-\sqrt{\gamma})^2$ and $\gamma_+ = (1+\sqrt{\gamma})^2$, for $0 < \gamma \leq 1$. This was plotted by the red line in Figure 3.2. Note that a very similar result also holds for the case when $\gamma > 1$.

Due to time limitation, we here do not show a full derivation of the Marchenko-Pastur distribution, however we provide a sketch of its proof (Bai [2] presents some different proofs). An approach to deriving this distribution is to use the so-called moment method. The key idea is to note that one can compute moments of the eigenvalue distribution in two ways and note that (in the limit) for any k (see also `Problem Sheets`)

$$\frac{1}{p}E\Big(\mathrm{tr}\big[\big(\boldsymbol{X}^T\boldsymbol{X}/n\big)^k\big]\Big) = \frac{1}{p}E\Big(\mathrm{tr}\big[\big(\boldsymbol{S}_n\big)^k\big]\Big) = E\Big(\frac{1}{p}\sum_{j=1}^{p}\lambda_j^k\Big) = \int_{\gamma_-}^{\gamma_+}\lambda^k f_\gamma(\lambda)d\lambda,$$

and that the quantities $\frac{1}{p}E\Big(\mathrm{tr}\big[\big(\boldsymbol{X}^T\boldsymbol{X}/n\big)^k\big]\Big)$ can be estimated (these estimates rely essentially in combinatorics). The distribution $f_\gamma(\lambda)$ can then be computed from its moments.

3.4. Spike model and BPP transition

What if there actually is some (linear) low dimensional structure in the data? Can we expect to capture it using PCA? A particularly simple, yet relevant, example to analyse is when the covariance matrix $\boldsymbol{\Sigma}$ is an identity with a rank 1 perturbation, which we refer to as a spike model $\boldsymbol{\Sigma} = \boldsymbol{I} + \beta\boldsymbol{v}\boldsymbol{v}^T$, for a unit norm vector $\boldsymbol{v}$ and $\beta \geq 0$.

One way of thinking about this instance is like when each data point $\boldsymbol{x}$ consists of a signal part $\sqrt{\beta}g_0\boldsymbol{v}$ with g_0 being a one-dimensional standard Gaussian (a Gaussian multiple of a fixed vector $\sqrt{\beta}\boldsymbol{v}$) and a noise part $\boldsymbol{g} \sim N(\boldsymbol{0}, \boldsymbol{I})$ (independent of g_0). Then $\boldsymbol{x} = \sqrt{\beta}g_0\boldsymbol{v} + \boldsymbol{g}$ is a Gaussian random vector as

$$\boldsymbol{x} \sim N(\boldsymbol{0}, \boldsymbol{I} + \beta\boldsymbol{v}\boldsymbol{v}^T).$$

A natural question is whether this rank 1 perturbation can be seen in $\boldsymbol{S}_n$. Let us build some intuition with an example. Figure 3.3 shows the histogram of the eigenvalues of a sample of $\boldsymbol{S}_n$ for $p = 500$, $n = 1000$, where $\boldsymbol{v}$ is the first element of the canonical basis $\boldsymbol{v} = \boldsymbol{e}_1$, and $\beta = 1.5$. The figure suggests that there is an eigenvalue of $\boldsymbol{S}_n$ that "pops out" of the support of the Marchenko-Pastur distribution (below we will estimate the location of this eigenvalue, and that estimate corresponds to the red "$\boldsymbol{x}$"). It is

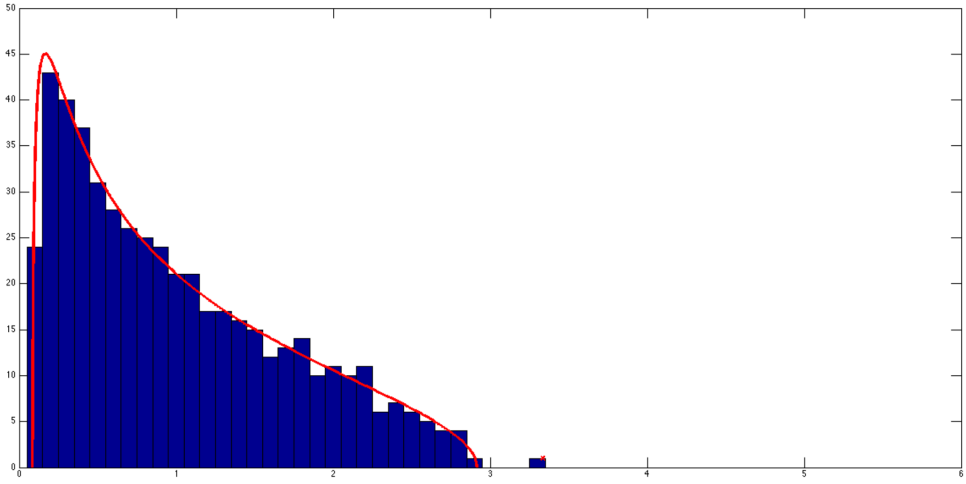


FIGURE 3.3. Histogram of the eigenvalues of $\boldsymbol{S}_n$ from Monte-Carlo simulations with spike when $\gamma = 0.5$ and $\beta = 1.5$. The red line is the Marchenko-Pastur distribution.

worth noticing that the largest eigenvalue of $\boldsymbol{\Sigma}$ is simply $1+\beta = 2.5$ while the largest eigenvalue of $\boldsymbol{S}_n$ appears considerably larger than that. Let us then try the same experiment but with $\beta = 0.5$. The result is shown in Figure 3.4. It seems that, for $\beta = 0.5$, the distribution of the eigenvalues appears to be undistinguishable from when $\boldsymbol{\Sigma} = \boldsymbol{I}$. This motivates the following question:

Question 2. For which values of γ and β can we expect to see an eigenvalue of $\boldsymbol{S}_n$ popping out of the support of the Marchenko-Pastur distribution, and what is the limit value that we expect it to take?

As we will show below, there is a critical value of β below which we do not expect to see a change in the distribution of eigenvalues and above which we expect one of the eigenvalues to pop out of the support. This is known as BPP transition (named after Baik et al. [4]). There are many papers about this phenomenon such as Paul [23], Baik et al. [4], Paul [24], Baik and Silverstein [3]. In what follows we will find the critical value of β and estimate the location of the largest eigenvalue of $\boldsymbol{S}_n$.

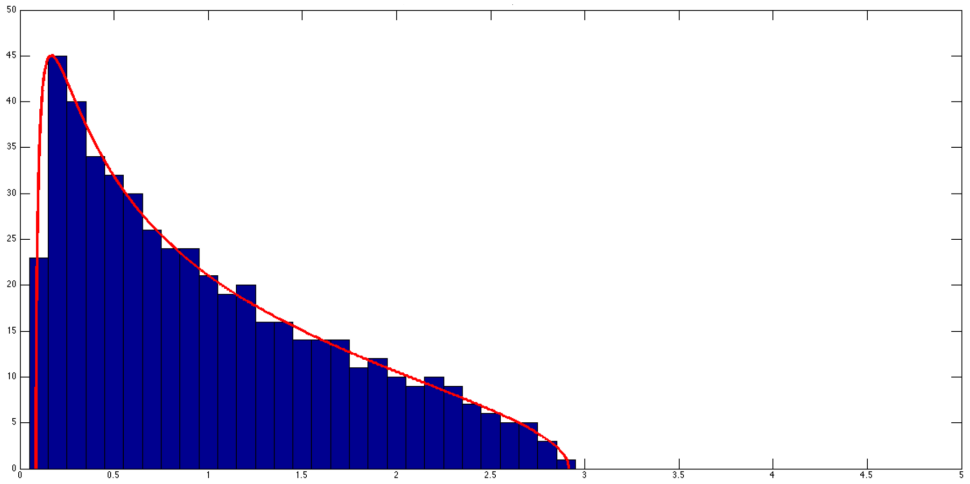


FIGURE 3.4. Histogram of the eigenvalues of $\boldsymbol{S}_n$ from Monte-Carlo simulations with spike when $\gamma = 0.5$ and $\beta = 0.5$. The red line is the Marchenko-Pastur distribution.

First of all, it is not difficult to see that we can assume that $\boldsymbol{v} = \boldsymbol{e}_1$ (since everything else is rotation invariant). We want to understand the behaviour of the leading eigenvalue of

$$\boldsymbol{S}_n = \frac{1}{n}\sum_{r=1}^{n} \boldsymbol{X}_r \boldsymbol{X}_r^T = \frac{1}{n}\boldsymbol{X}^T\boldsymbol{X}.$$

Using the Gaussian assumption and noting that the largest eigenvalue of $\boldsymbol{\Sigma} = \boldsymbol{I} + \beta \boldsymbol{v}\boldsymbol{v}^T$ with $\boldsymbol{v} = \boldsymbol{e}_1$ is $1+\beta$ (see `Problem Sheets`), we can represent the $n \times p$ data matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ as

$$\boldsymbol{X} = \begin{bmatrix} \sqrt{1+\beta}\boldsymbol{Z}_1 & \boldsymbol{Z}_2 \end{bmatrix},$$

where $\boldsymbol{Z}_1 \in \mathbb{R}^{n\times 1}$ and $\boldsymbol{Z}_2 \in \mathbb{R}^{n\times(p-1)}$ are both populated with i.i.d. standard Gaussian entries ($N(0,1)$). Then,

$$\boldsymbol{S}_n = \frac{1}{n}\boldsymbol{X}^T\boldsymbol{X} = \frac{1}{n}\begin{bmatrix} \sqrt{1+\beta}\boldsymbol{Z}_1^T \\ \boldsymbol{Z}_2^T \end{bmatrix}\begin{bmatrix} \sqrt{1+\beta}\boldsymbol{Z}_1 & \boldsymbol{Z}_2 \end{bmatrix} = \frac{1}{n}\begin{bmatrix} (1+\beta)\boldsymbol{Z}_1^T\boldsymbol{Z}_1 & \sqrt{1+\beta}\boldsymbol{Z}_1^T\boldsymbol{Z}_2 \\ \sqrt{1+\beta}\boldsymbol{Z}_2^T\boldsymbol{Z}_1 & \boldsymbol{Z}_2^T\boldsymbol{Z}_2 \end{bmatrix}.$$

Now, let $\hat{\lambda}$ denote an eigenvalue of $\boldsymbol{S}_n$, and further $\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{bmatrix}$ with $\boldsymbol{v}_1 \in \mathbb{R}$ and $\boldsymbol{v}_2 \in \mathbb{R}^{p-1}$ be the associated eigenvector. By the definition of eigenvalue and eigenvector we have

$$\frac{1}{n}\begin{bmatrix} (1+\beta)\boldsymbol{Z}_1^T\boldsymbol{Z}_1 & \sqrt{1+\beta}\boldsymbol{Z}_1^T\boldsymbol{Z}_2 \\ \sqrt{1+\beta}\boldsymbol{Z}_2^T\boldsymbol{Z}_1 & \boldsymbol{Z}_2^T\boldsymbol{Z}_2 \end{bmatrix}\begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{bmatrix} = \hat{\lambda}\begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{bmatrix},$$

which can be rewritten as

$$\frac{1}{n}(1+\beta)\boldsymbol{Z}_1^T\boldsymbol{Z}_1\boldsymbol{v}_1+\frac{1}{n}\sqrt{1+\beta}\boldsymbol{Z}_1^T\boldsymbol{Z}_2\boldsymbol{v}_2=\hat{\lambda}\boldsymbol{v}_1 \tag{3.9}$$

$$\frac{1}{n}\sqrt{1+\beta}\boldsymbol{Z}_2^T\boldsymbol{Z}_1\boldsymbol{v}_1+\frac{1}{n}\boldsymbol{Z}_2^T\boldsymbol{Z}_2\boldsymbol{v}_2=\hat{\lambda}\boldsymbol{v}_2. \tag{3.10}$$

Equation (3.10) is equivalent to

$$\frac{1}{n}\sqrt{1+\beta}\boldsymbol{Z}_2^T\boldsymbol{Z}_1\boldsymbol{v}_1=\big(\hat{\lambda}\boldsymbol{I}-\frac{1}{n}\boldsymbol{Z}_2^T\boldsymbol{Z}_2\big)\boldsymbol{v}_2.$$

When $\hat{\lambda}\boldsymbol{I}-\frac{1}{n}\boldsymbol{Z}_2^T\boldsymbol{Z}_2$ is invertible (Paul [23] provides a justification), we can rewrite the above equation as

$$\boldsymbol{v}_2=\big(\hat{\lambda}\boldsymbol{I}-\frac{1}{n}\boldsymbol{Z}_2^T\boldsymbol{Z}_2\big)^{-1}\frac{1}{n}\sqrt{1+\beta}\boldsymbol{Z}_2^T\boldsymbol{Z}_1\boldsymbol{v}_1.$$

Plugging in this into (3.9), we get

$$\frac{1}{n}(1+\beta)\boldsymbol{Z}_1^T\boldsymbol{Z}_1\boldsymbol{v}_1+\frac{1}{n}\sqrt{1+\beta}\boldsymbol{Z}_1^T\boldsymbol{Z}_2\big(\hat{\lambda}\boldsymbol{I}-\frac{1}{n}\boldsymbol{Z}_2^T\boldsymbol{Z}_2\big)^{-1}\frac{1}{n}\sqrt{1+\beta}\boldsymbol{Z}_2^T\boldsymbol{Z}_1\boldsymbol{v}_1=\hat{\lambda}\boldsymbol{v}_1.$$

If $\boldsymbol{v}_1\neq\boldsymbol{0}$, we obtain

$$\hat{\lambda}=\frac{1}{n}(1+\beta)\boldsymbol{Z}_1^T\boldsymbol{Z}_1+\frac{1}{n}\sqrt{1+\beta}\boldsymbol{Z}_1^T\boldsymbol{Z}_2\big(\hat{\lambda}\boldsymbol{I}-\frac{1}{n}\boldsymbol{Z}_2^T\boldsymbol{Z}_2\big)^{-1}\frac{1}{n}\sqrt{1+\beta}\boldsymbol{Z}_2^T\boldsymbol{Z}_1.$$

Now, because $\boldsymbol{Z}_1\in\mathbb{R}^n$ has standard Gaussian entries, $\frac{1}{n}\boldsymbol{Z}_1^T\boldsymbol{Z}_1\to 1$, meaning that

$$\hat{\lambda}=(1+\beta)\Big[1+\frac{1}{n}\boldsymbol{Z}_1^T\boldsymbol{Z}_2\big(\hat{\lambda}\boldsymbol{I}-\frac{1}{n}\boldsymbol{Z}_2^T\boldsymbol{Z}_2\big)^{-1}\frac{1}{n}\boldsymbol{Z}_2^T\boldsymbol{Z}_1\Big]. \tag{3.11}$$

Consider the full SVD of $\boldsymbol{Z}_2=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ where $\boldsymbol{U}\in\mathbb{R}^{n\times n}$ and $\boldsymbol{V}\in\mathbb{R}^{(p-1)\times(p-1)}$ are two orthogonal matrices (meaning that $\boldsymbol{U}^T\boldsymbol{U}=\boldsymbol{U}\boldsymbol{U}^T=\boldsymbol{I}_n$ and $\boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{V}\boldsymbol{V}^T=\boldsymbol{I}_{p-1}$), and $\boldsymbol{\Sigma}\in\mathbb{R}^{n\times(p-1)}$ is a matrix of singular values. Take $\boldsymbol{D}=\frac{1}{n}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}$ (note that $\boldsymbol{D}^T=\boldsymbol{D}$), then

$$\frac{1}{n}\boldsymbol{Z}_2^T\boldsymbol{Z}_2=\frac{1}{n}\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\boldsymbol{V}^T=\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T,$$

indicating that the diagonal entries of $\boldsymbol{D}$ correspond to the eigenvalues of $\frac{1}{n}\boldsymbol{Z}_2^T\boldsymbol{Z}_2$ which we expect to be distributed (in the limit) according to the Marchenko-Pastur distribution for $\frac{p-1}{n}\approx\gamma$. Replacing this in (3.11) gives (see `Problem Sheets` for the calculations)

$$\hat{\lambda}=(1+\beta)\Big[1+\frac{1}{n^2}\big(\boldsymbol{U}^T\boldsymbol{Z}_1\big)^T\boldsymbol{\Sigma}\big(\hat{\lambda}\boldsymbol{I}-\boldsymbol{D}\big)^{-1}\boldsymbol{\Sigma}^T\big(\boldsymbol{U}^T\boldsymbol{Z}_1\big)\Big]. \tag{3.12}$$

Define $\boldsymbol{g}:=\boldsymbol{U}^T\boldsymbol{Z}_1\in\mathbb{R}^n$. Because the columns of $\boldsymbol{U}$ are orthonormal, $\boldsymbol{g}$ is an isotropic Gaussian ($\boldsymbol{g}\sim N(\boldsymbol{0},\boldsymbol{I})$), and also

$$E(\boldsymbol{g}\boldsymbol{g}^T)=E(\boldsymbol{U}^T\boldsymbol{Z}_1\boldsymbol{Z}_1^T\boldsymbol{U})=\boldsymbol{U}^TE(\boldsymbol{Z}_1\boldsymbol{Z}_1^T)\boldsymbol{U}=\boldsymbol{U}^T\boldsymbol{U}=\boldsymbol{I}_n.$$

We then have

$$\begin{aligned}\hat{\lambda} &= (1+\beta)\Big[1+\frac{1}{n^2}\boldsymbol{g}^T\boldsymbol{\Sigma}\big(\hat{\lambda}\boldsymbol{I}-\boldsymbol{D}\big)^{-1}\boldsymbol{\Sigma}^T\boldsymbol{g}\Big]\\ &= (1+\beta)\Big[1+\frac{1}{n}\boldsymbol{g}^T\big(\frac{1}{\sqrt{n}}\boldsymbol{\Sigma}\big)\big(\hat{\lambda}\boldsymbol{I}-\boldsymbol{D}\big)^{-1}\big(\frac{1}{\sqrt{n}}\boldsymbol{\Sigma}^T\big)\boldsymbol{g}\Big]\\ &= (1+\beta)\Big[1+\frac{1}{n}\sum_{j=1}^{p-1}\boldsymbol{g}_j^2\frac{\boldsymbol{D}_{jj}}{\hat{\lambda}-\boldsymbol{D}_{jj}}\Big]\\ &= (1+\beta)\Big[1+\frac{\gamma}{p-1}\sum_{j=1}^{p-1}\boldsymbol{g}_j^2\frac{\boldsymbol{D}_{jj}}{\hat{\lambda}-\boldsymbol{D}_{jj}}\Big].\end{aligned}$$

Note that $\boldsymbol{D}$ has $p-1$ diagonal elements $\boldsymbol{D}_{jj}$, but $\boldsymbol{D}_{jj}=0$ for $j>n$ when $n\ll p$. Since we expect the diagonal entries of $\boldsymbol{D}$ to be distributed according to the Marchenko-Pastur distribution and $\boldsymbol{g}$ to be independent to it, we expect to have (recall $E(\boldsymbol{g}_j^2)=1$)

$$\frac{1}{p-1}\sum_{j=1}^{p-1}\boldsymbol{g}_j^2\frac{\boldsymbol{D}_{jj}}{\hat{\lambda}-\boldsymbol{D}_{jj}}\to\int_{\gamma_-}^{\gamma_+}\frac{x}{\hat{\lambda}-x}f_\gamma(x)dx.$$

This leads to the following equation for $\hat{\lambda}$

$$\hat{\lambda}=(1+\beta)\Big[1+\gamma\int_{\gamma_-}^{\gamma_+}\frac{x}{\hat{\lambda}-x}f_\gamma(x)dx\Big].$$

This equation can be solved with the help of a computer programme that computes integrals symbolically (such as Mathematica) to give (see Paul [23] for a derivation)

$$\hat{\lambda}=(1+\beta)\big(1+\frac{\gamma}{\beta}\big).$$

An important point to notice here is that for $\beta=\sqrt{\gamma}$ we have

$$\hat{\lambda}=(1+\sqrt{\gamma})\big(1+\frac{\gamma}{\sqrt{\gamma}}\big)=(1+\sqrt{\gamma})^2=\gamma_+,$$

suggesting that $\beta=\sqrt{\gamma}$ is the critical point (see also `Problem Sheets`).

Indeed this is the case and it is possible to show that this generally holds for the leading eigenvalue (see, e.g., Paul [23]) as

- If $\beta\le\sqrt{\gamma}$ then
$$\lambda_{\max}\to\gamma_+.$$
- If $\beta>\sqrt{\gamma}$ then
$$\lambda_{\max}\to(1+\beta)\big(1+\frac{\gamma}{\beta}\big)>\gamma_+.$$

3.5. Sparse principal components analysis

In this section, we study sparse principal components analysis, abbreviated here as sparse PCA. We interpret principal components by examining the direction vectors $\boldsymbol{v}_j$ (the loadings) to see which variables play a role. With high dimensional data often this interpretation can be made easier if the loadings are sparse. In this section, we study two methods for deriving principal components with sparse loadings. They are based on the lasso L_1 penalty.

As before, consider an $n \times p$ data matrix $\boldsymbol{X}$ with centred columns. Those two methods focus on either the maximum-variance property of principal components, or the minimum reconstruction error. As the first approach, Jolliffe et al. [17] proposed the so-called SCoTLASS procedure, by solving

$$\max_{\boldsymbol{v}} \boldsymbol{v}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{v} \quad \text{subject to } \boldsymbol{v}^T \boldsymbol{v} = 1, \sum_{j=1}^{p} |\boldsymbol{v}_j| \leq t.$$

The absolute value constraint encourages some of the loadings to be zero and hence $\boldsymbol{v}$ to be sparse. Further sparse principal components are found in the same way, by forcing the k-th component to be orthogonal to the first $k-1$ components. However, this problem is not convex, so the computations are difficult.

As the second method, Zou et al. [33] instead started with the regression/reconstruction property of the PCA. As before, let $\boldsymbol{X}_i^T$ be the i-th row of $\boldsymbol{X}$. For a single component, e.g. with the leading eigenvector, their sparse PCA technique solves

$$\min_{\boldsymbol{v},\boldsymbol{\theta}} \sum_{i=1}^{n} \left\| \boldsymbol{X}_i - \boldsymbol{\theta} \boldsymbol{v}^T \boldsymbol{X}_i \right\|_2^2 + \lambda \left\| \boldsymbol{v} \right\|_2^2 + \lambda_1 \left\| \boldsymbol{v} \right\|_1^1, \quad \text{subject to } \boldsymbol{\theta}^T \boldsymbol{\theta} = 1.$$

We investigate this approach in more detail below to clarify the motivation behind it.

- If both λ and λ_1 are zero and $n > p$ as in low dimensional data, it is easy to show that $\boldsymbol{v} = \boldsymbol{\theta}$ and is the leading principal component direction (see `Problem Sheets`).

- When $n \ll p$ as in high dimensional data, the solution is not necessarily unique unless $\lambda > 0$. For $\lambda > 0$ and $\lambda_1 = 0$ (ridge penalty), the solution for $\boldsymbol{v}$ is proportional to the leading principal component direction (see Theorem 2 in Zou et al. [33]).

- The second penalty on $\boldsymbol{v}$ (i.e., L_1-norm) encourages sparseness of the loadings.

For the general case of d components, this sparse PCA procedure minimises

$$\min_{\boldsymbol{V},\boldsymbol{\Theta}} \sum_{i=1}^{n} \left\| \boldsymbol{X}_i - \boldsymbol{\Theta}\boldsymbol{V}^T\boldsymbol{X}_i \right\|_2^2 + \lambda \sum_{j=1}^{d} \left\| \boldsymbol{v}_j \right\|_2^2 + \sum_{j=1}^{d} \lambda_{1j} \left\| \boldsymbol{v}_j \right\|_1^1, \quad \text{subject to } \boldsymbol{\Theta}^T\boldsymbol{\Theta} = \boldsymbol{I}_d, \tag{3.13}$$

where $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d] \in \mathbb{R}^{p \times d}$ as in Section 3.2, and also $\boldsymbol{\Theta}$ is $p \times d$.

Computation of the sparse PCA (3.13):
The criterion (3.13) is not jointly convex in $\boldsymbol{V}$ and $\boldsymbol{\Theta}$, but it is convex in each parameter with the other parameter being fixed. Minimisation over $\boldsymbol{V}$ with fixed $\boldsymbol{\Theta}$ is equivalent to d elastic net problems, so this can be done efficiently (for example using the R package `glmnet`). On the other hand, minimisation over $\boldsymbol{\Theta}$ with fixed $\boldsymbol{V}$ is a version of the Procrustes problem which can be solved by an SVD calculation. These steps are alternated until convergence. The computation algorithm is easily implemented in standard software.

Example 3.1. Figure 3.5 shows an example of sparse PCA using (3.13), taken from Sjostrand et al. [27]. Here the shape of the mid-sagittal cross-section of the corpus callosum (CC) is related to various clinical parameters in a study involving 569 elderly persons. In this example PCA is applied to shape data, and is a popular tool in morphometrics. For such applications, a number of landmarks are identified along the circumference of the shape; an example is given in Figure 3.6. These are aligned by Procrustes analysis to allow for rotations, as well as scaling in this case. The features used for the PCA are the sequence of coordinate pairs for each landmark, unpacked into a single vector. In this analysis, both standard principal components and sparse principal components were computed, and components that were significantly associated with various clinical parameters were identified. In Figure 3.5, the shape variations corresponding to significant principal components (red curves) are overlaid on the mean CC shape (black curves). Low walking speed relates to CCs that are thinner (displaying atrophy) in regions connecting the motor control and cognitive centres of the brain. Low verbal fluency relates to CCs that are thinner in regions connecting auditory/visual/cognitive centres. For more explanations on these, see Section D of Sjostrand et al. [27]. The sparse PCA procedure seems to give a more parsimonious and potentially more informative picture of the important differences.

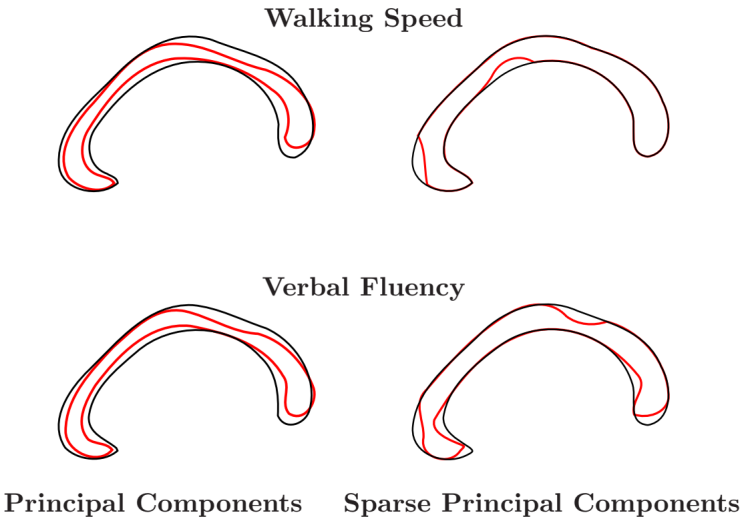


FIGURE 3.5. Standard and sparse principal components from a study of the corpus callosum variation. The shape variations corresponding to significant principal components (red curves) are overlaid on the mean CC shape (black curves).

FIGURE 3.6. An example of a mid-saggital brain slice, with the corpus callosum (CC) annotated with landmarks.

CHAPTER 4

Cluster analysis for high dimensional data

As we are in the era of "big data or modern data", data come from multiple sources and are often inhomogeneous meaning that the observations come from different distributions. A recipe for dealing with such inhomogeneous data is to consider them as an assemblage of several homogeneous data sets, corresponding to homogeneous "subpopulations". Then each subpopulation can be treated either independently or jointly. The main hurdle in this approach is to recover the unknown subpopulations, which is the main goal of cluster analysis.

Clustering can also be used for other purposes. Two important motivations are scientific understanding and data quantisation. In many fields, finding groups of items with similar behaviour is of primary interest, as finding structures is a first step towards the scientific understanding of complex systems. For example, finding groups of genes with similar expression level profiles is important in biology, as these genes are likely to be involved in a common regulatory mechanism. Summarising a cloud of n data points by a smaller cloud of k points is another important motivation for clustering. For example, we may wish to summarise the expression level profiles of thousands of genes by a small number of representative profiles (templates) and then only work with these templates which are lighter to handle. Another goal is sometimes to arrange the clusters into a natural hierarchy. This involves successively grouping the clusters themselves so that at each level of the hierarchy, clusters within the same group are more similar to each other than those in different groups.

Central to all of the goals of cluster analysis is the notion of degree of similarity (or proximity) between the individual objects being clustered. A clustering method attempts to group objects based on the definition of similarity supplied to it. The choice

of similarity (or dissimilarity) measures is often crucial as it can influence the outcome of clustering. As we saw in Chapter 1, the distances between observations do not behave naturally in high dimensions when the dimension p grows.

In this chapter, we first study the common clustering methods for multivariate data, especially the K-means clustering method and the hierarchical clustering method as in Hastie et al. [13]. We then study cluster analysis of high dimensional data using dimensionality reduction.

4.1. K-means clustering

The K-means clustering is one of the most popular iterative descent clustering methods. It is applicable to situations where all variables are of the quantitative type. Suppose that we have n observations $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n$, each p-dimensional as before. The K-means algorithm directly assigns each observation to a group or cluster without regard to a probability model describing the data. A prespecified number of clusters K, $K < n$, is postulated, and each cluster is labelled by an integer $k \in \{1,\ldots,K\}$. Each observation $\boldsymbol{X}_i$ is assigned to one and only one cluster. These assignments can be characterised by a many-to-one mapping, or encoder $C(i) = k$, that assigns the i-th observation to the k-th cluster. We seek an optimal encoder $C^*(i)$ that achieves our required goal (e.g., minimise a mathematical loss function) based on the dissimilarities $d(\boldsymbol{X}_i,\boldsymbol{X}_j)$ between every pair of observations $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, where $d(\cdot,\cdot)$ is a dissimilarity measure specified by the user (such as the Euclidean distance). Generally, the encoder $C(i)$ is explicitly delineated by giving its value (cluster assignment) for each observation i. Thus, the parameters of the procedure are the individual cluster assignments for each of the n observations. These are adjusted so as to minimise a loss function that characterises the degree to which the clustering goal is not met.

In K-means clustering algorithm, the squared Euclidean distance

$$d(\boldsymbol{X}_i,\boldsymbol{X}_j) = \left\|\boldsymbol{X}_i - \boldsymbol{X}_j\right\|_2^2 = \sum_{l=1}^{p} \left(\boldsymbol{X}_{il} - \boldsymbol{X}_{jl}\right)^2$$

is used as the dissimilarity measure. As explained above, one needs to specify a mathematical loss function and attempt to minimise it through some combinatorial optimisation algorithm. Since the goal of K-means clustering is to assign close points to the

same cluster, a natural loss function for this purpose is

$$L(C) = \sum_{k=1}^{K} N_k \sum_{C(i)=k} \|\boldsymbol{X}_i - \bar{\boldsymbol{X}}_k\|_2^2,$$

where $N_k = \sum_{i=1}^{n} 1(C(i) = k)$ is the number of observations assigned to the k-th cluster and $\bar{\boldsymbol{X}}_k = \frac{1}{N_k} \sum_{C(i)=k} \boldsymbol{X}_i$ is the mean of observations in the k-th cluster. This criterion is then minimised by assigning the n observations to the K clusters in such a way that within each cluster the average dissimilarity of the observations from the cluster mean, as defined by observations in that cluster, is minimised. That is, we try to find an optimal clustering assignment C^* such that

$$C^* = \arg\min_{C} L(C) = \arg\min_{C} \sum_{k=1}^{K} N_k \sum_{C(i)=k} \|\boldsymbol{X}_i - \bar{\boldsymbol{X}}_k\|_2^2.$$

An iterative descent algorithm for solving this optimisation problem can be obtained by noting that for any set of observations S (see `Problem Sheets`)

$$\bar{\boldsymbol{X}}_S = \arg\min_{\boldsymbol{m}} \sum_{i \in S} \|\boldsymbol{X}_i - \boldsymbol{m}\|_2^2. \tag{4.1}$$

Hence we can obtain C^* by solving the enlarged optimisation problem

$$\min_{C, \{\boldsymbol{m}_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k} \|\boldsymbol{X}_i - \boldsymbol{m}_k\|_2^2. \tag{4.2}$$

The reason is that this can be feasibly minimised by an alternating optimisation procedure as given in Algorithm 2.

Algorithm 2: K-means clustering

1. For a given cluster assignment C, the total cluster variance (4.2) is minimised with respect to $\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_K\}$ yielding the means of the currently assigned clusters (4.1).
2. Given a current set of means $\{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_K\}$, the total cluster variance (4.2) is minimised by assigning each observation to the closest (current) cluster mean. That is,
$$C(i) = \arg\min_{1 \le k \le K} \|\boldsymbol{X}_i - \boldsymbol{m}_k\|_2^2.$$
3. Steps 1 and 2 are iterated until the assignments do not change.

Note that each of steps 1 and 2 in Algorithm 2 reduces the value of the criterion (4.2), so that convergence is assured. However, the result may represent a suboptimal local

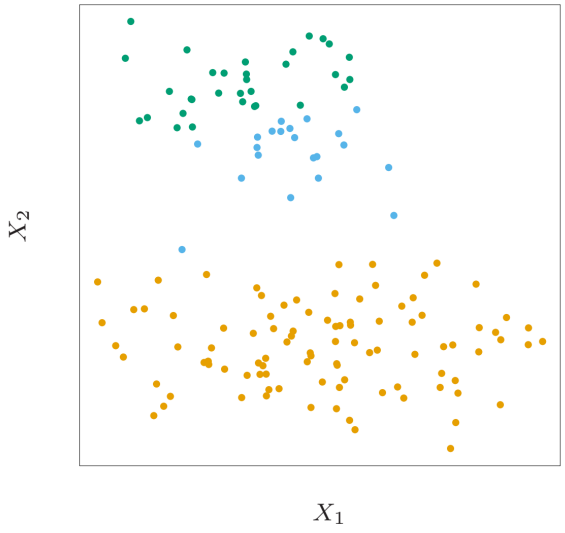


FIGURE 4.1. Simulated data in a plane, clustered into three classes (represented by orange, blue and green) by the K-means clustering algorithm.

minimum. The algorithm of Hartigan and Wong [12] goes further, and ensures that there is no single switch of an observation from one group to another group that will decrease the objective. In practice, one should start the algorithm with many different random choices for the starting means, and choose the solution having smallest value of the objective function. The K-means clustering algorithm is implemented in the R package `kmeans`.

Figure 4.1 shows some simulated data clustered into three groups using the K-means algorithm. In this simulated example, two of the clusters are not well separated. Also, Figure 4.2 shows some of the K-means iterations for these simulated data. The centroids are depicted by "O"s. The straight lines show the partitioning of points, each sector being the set of points closest to each centroid. This partitioning is called the `Voronoi tessellation`. After 20 iterations the procedure has converged.

Remark 4.1. The K-means clustering method requires a prespecified number of clusters, however in practice we can find an optimal number of clusters using a scree plot

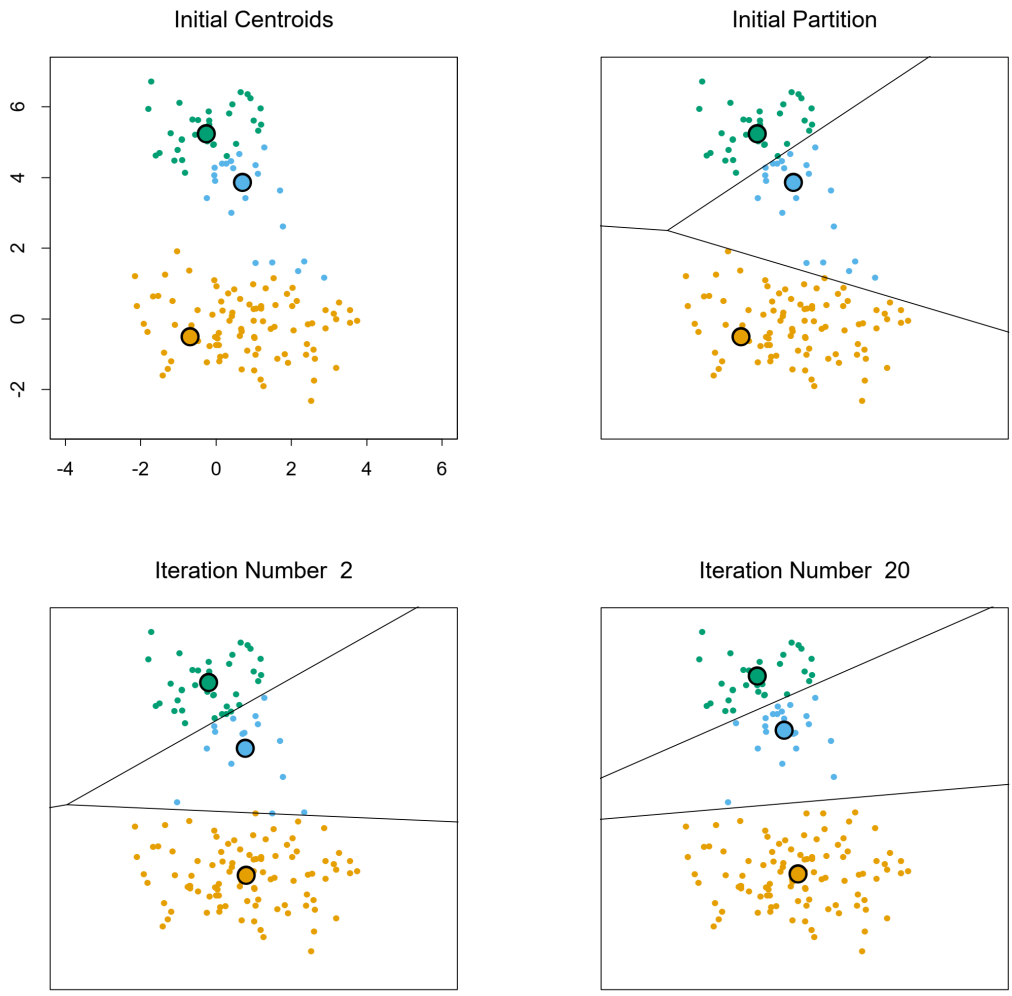


FIGURE 4.2. Successive iterations of the K-means clustering algorithm for the simulated data of Figure 4.1.

(i.e., elbow method) based on the total within sum of squares of distances from cluster mean. We will have examples on this in our Computer Practical session. A more sophisticated approach is to use the gap statistic [30].

Remark 4.2. The K-means clustering is based on the L_2-norm distances between observations which turns into clustering based on cluster mean as motivated by equation (4.1). One can also use the L_1-norm distances between observations which turns into

using cluster median for clustering instead of cluster mean. This is because the minimiser of the L_1-norm function (i.e., absolute value function) is the median of observations. This method is known as K-medians clustering and there are R packages for implementing this such as `flexclust`. Which method is better depends on the application, however if data contains extreme values (e.g., outliers), the K-medians clustering may be preferred because the median does not get much affected by outliers.

4.2. Hierarchical clustering

The results of applying K-means clustering algorithm depend on the choice for the number of clusters to be searched and a starting configuration assignment. In contrast, hierarchical clustering methods do not require such specifications. Instead, they require the user to specify a measure of dissimilarity between (disjoint) groups of observations, based on the pairwise dissimilarities among the observations in the two groups. As the name suggests, they produce hierarchical representations in which the clusters at each level of the hierarchy are created by merging clusters at the next lower level. At the lowest level, each cluster contains a single observation. At the highest level there is only one cluster containing all of the observations.

Strategies for hierarchical clustering divide into two basic paradigms: agglomerative (bottom-up) and divisive (top-down). The agglomerative strategy starts at the bottom and at each level recursively merges a selected pair of clusters into a single cluster. This produces a grouping at the next higher level with one less cluster. The pair chosen for merging consist of the two groups with the smallest intergroup dissimilarity. The divisive method starts at the top and at each level recursively splits one of the existing clusters at that level into two new clusters. The split is chosen to produce two new groups with the largest between-group dissimilarity. With both paradigms there are $n-1$ levels in the hierarchy.

Each level of the hierarchy represents a particular grouping of the data into disjoint clusters of observations. The entire hierarchy represents an ordered sequence of such groupings. It is up to the user to decide which level represents a “natural” clustering in the sense that observations within each of its groups are sufficiently more similar to each other than to observations assigned to different groups at that level.

A dendrogram provides a highly interpretable complete description of the hierarchical clustering in a graphical format. This is one of the main reasons for the popularity of hierarchical clustering methods.

Example 4.1. Figure 4.3 shows a dendrogram resulting from agglomerative clustering with average linkage to a high dimensional data set called human tumor microarray data (we will see more details of this application in our Computer Practical session). The agglomerative clustering will be discussed in detail in the following subsection. Cutting the dendrogram horizontally at a particular height partitions the data into disjoint clusters represented by the vertical lines that intersect it. These are the clusters that would be produced by terminating the procedure when the optimal intergroup dissimilarity exceeds that threshold cut value. Groups that merge at high values, relative to the merger values of the subgroups contained within them lower in the tree, are candidates for natural clusters. Note that this may occur at several different levels, indicating a clustering hierarchy: that is, clusters nested within clusters.

Such a dendrogram is often viewed as a graphical summary of the data itself, rather than a description of the results of the algorithm. However, such interpretations should be treated with caution. First, different hierarchical methods (see below), as well as small changes in the data, can lead to quite different dendrograms. Also, such a summary will be valid only to the extent that the pairwise observation dissimilarities possess the hierarchical structure produced by the algorithm. Hierarchical methods impose hierarchical structure whether or not such structure actually exists in the data.

4.2.1. Agglomerative clustering

Agglomerative clustering algorithms begin with every observation representing a single cluster. At each of the $n-1$ steps the closest two (least dissimilar) clusters are then merged into a single cluster, producing one less cluster at the next higher level. Therefore, a measure of dissimilarity between two clusters (groups of observations) must be defined.

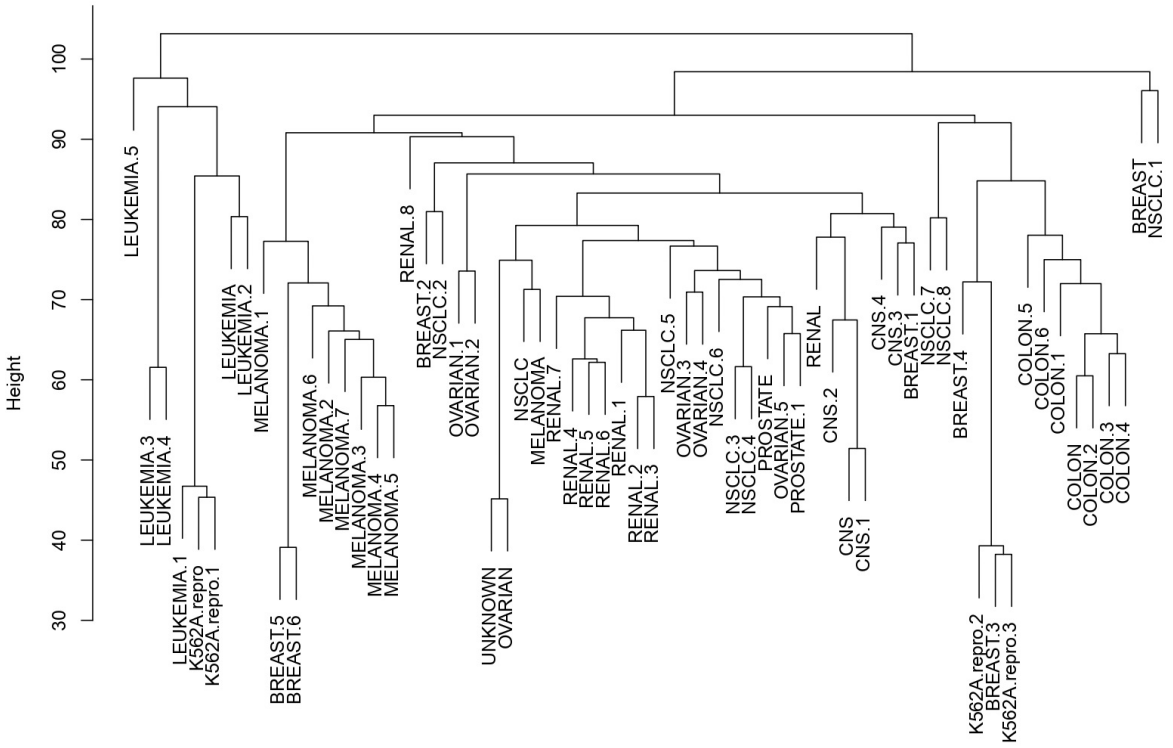


FIGURE 4.3. The dendrogram resulting from agglomerative clustering with average linkage to the high dimensional human tumor microarray data.

Let G and H represent two such groups. The dissimilarity $d(G,H)$ between G and H is computed from the set of pairwise observation dissimilarities $d_{ii'}$ where one member of the pair i is in G and the other i' is in H. Single linkage (SL) agglomerative clustering takes the intergroup dissimilarity to be that of the closest (least dissimilar) pair

$$d_{SL}(G,H) = \min_{\substack{i\in G \\ i'\in H}} d_{ii'}.$$

This is a kind of nearest-neighbour technique. Complete linkage (CL) agglomerative clustering (furthest-neighbour technique) takes the intergroup dissimilarity to be that of the furthest (most dissimilar) pair

$$d_{CL}(G,H) = \max_{\substack{i\in G \\ i'\in H}} d_{ii'}.$$

Group average (GA), or average linkage, clustering uses the average dissimilarity between the groups

$$d_{GA}(G,H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{i' \in H} d_{ii'},$$

where N_G and N_H are the respective number of observations in each group. Although there have been many other proposals for defining intergroup dissimilarity in the context of agglomerative clustering, the above three are the ones most commonly used. In our Computer Practical session we show examples of all these three approaches.

If the data dissimilarities $\{d_{ii'}\}$ exhibit a strong clustering tendency, with each of the clusters being compact and well separated from others, then all the three agglomerative methods produce similar results. Clusters are compact if all of the observations within them are relatively close together (small dissimilarities) as compared with observations in different clusters. To the extent this is not the case, results will differ.

Remark 4.3. One can argue that group average clustering has a statistical consistency property violated by single and complete linkage approaches. Assume we have attribute-value data $\boldsymbol{X} = (X_1, \ldots, X_p)$ and that each cluster k is a random sample from some population joint density $p_k(\boldsymbol{x})$. The complete data set is then a random sample from a mixture of K such densities. The group average dissimilarity $d_{GA}(G,H)$ is an estimate of

$$\int \int d(\boldsymbol{x},\boldsymbol{x}') p_G(\boldsymbol{x}) p_H(\boldsymbol{x}') d\boldsymbol{x} d\boldsymbol{x}', \tag{4.3}$$

where $d(\boldsymbol{X},\boldsymbol{X}')$ is the dissimilarity between points $\boldsymbol{X}$ and $\boldsymbol{X}'$ in the space of attribute values. As the sample size n approaches infinity $d_{GA}(G,H)$ approaches (4.3), which is a characteristic of the relationship between the two densities $p_G(\boldsymbol{x})$ and $p_H(\boldsymbol{x})$. For single linkage, $d_{SL}(G,H)$ approaches zero as $n \to \infty$ independent of $p_G(\boldsymbol{x})$ and $p_H(\boldsymbol{x})$. For complete linkage, $d_{CL}(G,H)$ becomes infinite as $n \to \infty$, again independent of the two densities. Thus, it is not clear what specific aspects of the population distribution are estimated by $d_{SL}(G,H)$ and $d_{CL}(G,H)$.

4.2.2. Divisive clustering

Divisive clustering algorithms begin with the entire data set as a single cluster, and recursively divide one of the existing clusters into two daughter clusters at each iteration in a top-down fashion. This approach has not been studied nearly as extensively as agglomerative methods in the clustering literature. It has been explored somewhat in the engineering literature (Gersho and Gray [9]) in the context of compression. In the clustering setting, a potential advantage of divisive over agglomerative methods can occur when interest is focused on partitioning the data into a relatively small number of clusters.

The divisive paradigm can be employed by recursively applying K-means method with $K = 2$ to perform the splits at each iteration. However, such an approach would depend on the starting configuration specified at each step. Moreover, it would not necessarily produce a splitting sequence that possesses the monotonicity property (e.g., decreasing similarities here) required for hierarchical representation.

A divisive algorithm that avoids these problems was proposed by Macnaughton-Smith et al. [19]. It begins by placing all observations in a single cluster G. It then chooses that observation whose average dissimilarity from all the other observations is largest. This observation forms the first member of a second cluster H. At each successive step that observation in G whose average distance from all other observations in G, minus that for those in H is largest, is transferred to H. This continues until the corresponding difference in averages becomes negative. That is, there are no longer any observations in G that are, on average, closer to those in H. The result is a split of the original cluster into two daughter clusters, the observations transferred to H and those remaining in G. These two clusters represent the second level of the hierarchy. Each successive level is produced by applying this splitting procedure to one of the clusters at the previous level. A question that arises is: how should we choose the cluster G to split? Kaufman and Rousseeuw [18] suggested choosing the cluster at each level with the largest diameter $d_{CL}(G,G) = \max_{\substack{i\in G \\ i'\in G}} d_{ii'}$ for splitting. An alternative would be to choose the one with the largest average dissimilarity among its members

$$\bar{d}_G = \frac{1}{N_G} \sum_{i\in G} \sum_{i'\in G} d_{ii'}.$$

The recursive splitting then continues until all clusters either become singletons or all members of each one have zero dissimilarity from one another.

4.3. High dimensional clustering using dimensionality reduction

In this section, we study how dimensionality reduction techniques can be used for clustering of high dimensional data. High dimensional data are often transformed into lower dimensional data, for example using variable selection techniques or via PCA where coherent patterns can be detected clearly. Such dimensionality reduction is used in different areas such as meteorology, image processing, genomics and information retrieval. One can use PCA to project high dimensional data to a lower dimensional subspace as we saw in Chapter 3, and then apply a standard clustering method such as K-means or hierarchical clustering to the projected data.

For this method, we first apply the PCA on high dimensional observations $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n$ through applying the SVD on the $n \times p$ data matrix $\boldsymbol{X}$ (assuming all the columns are centred at zero). This gives $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$. We know that $\boldsymbol{X}\boldsymbol{V} = \boldsymbol{U}\boldsymbol{D}$ represents the principal components. Let $\boldsymbol{V}_r$ denote the first r columns of $\boldsymbol{V}$, that is $\boldsymbol{V}_r$ is a $p \times r$ submatrix of $\boldsymbol{V}$. Then $\boldsymbol{X}\boldsymbol{V}_r$ represents the r leading principal components. We ideally hope there is $r \ll p$ so a high proportion of variance is preserved. Let $\boldsymbol{Z} := \boldsymbol{X}\boldsymbol{V}_r$ denote the $n \times r$ data matrix with reduced dimension. This means $\boldsymbol{Z}_1,\ldots,\boldsymbol{Z}_n$ are the lower dimensional transformation of $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n$ by PCA, with each $\boldsymbol{Z}_i$ being r dimensional.

Algorithm 3: High dimensional K-means clustering using PCA

1. Apply PCA to the high dimensional observations $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n$ using SVD (with $n \ll p$).
2. For $r \ll p$, choose the first r leading principal components that preserve a high proportion of data variance. Denote them by $\boldsymbol{Z}_1,\ldots,\boldsymbol{Z}_n$.
3. With a prespecified number of clusters K, apply the standard K-means clustering algorithm 2 to $\boldsymbol{Z}_1,\ldots,\boldsymbol{Z}_n$ to assign the observations to K clusters.

Remark 4.4. Algorithm 3 can be easily implemented in standard software. Note that one can similarly apply hierarchical clustering for high dimensional data using PCA. For this, in Step 3 of Algorithm 3, one can simply apply the standard hierarchical clustering procedure to the transformed data $\boldsymbol{Z}_1,\ldots,\boldsymbol{Z}_n$.

Remark 4.5. Ding and He [7] explored a connection between K-means clustering and PCA. They showed that principal components are actually the continuous solution of the cluster membership indicators in the K-means clustering method, that is, the PCA dimension reduction automatically performs data clustering according to the K-means objective function. Using empirical examples, they illustrated that when dimensions are reduced, the clustering result significantly improves (see Table 2 in their paper). Also, more recently, Mukherjee and Zhang [22] wrote a paper on the power of PCA in clustering problems. They showed that PCA can significantly reduce the distance of data points belonging to the same clusters while reducing inter-cluster distances relatively mildly. Although PCA can intuitively help clustering especially in high dimensions. However, Chang [6] argued that, under some assumptions, the set of PC's with the largest eigenvalues (variances) does not necessarily capture cluster structure info. In fact, one should be cautious that PCA may not necessarily improve cluster quality and sometimes clustering based on first PC's can be even worse than clustering without PCA, depending on data. Yeung and Ruzzo [31] provides an interesting discussion and example on this. It should be noted that the effect of PCA on clustering really depends on the application as well as on the clustering algorithm and similarity metrics used.

Assessing clustering result:

In the literature, there are some measures for assessing clustering results and for comparing different clustering methods. In order to compare clustering results against external criteria, a measure of agreement is needed. One of the simple measures for this is the Rand index, proposed by Rand [26], which assesses the degree of agreement between two partitions of the same set of objects. Given a set of n objects $S = \{O_1, \ldots, O_n\}$, suppose $U = \{u_1, \ldots, u_R\}$ and $V = \{v_1, \ldots, v_C\}$ represent two different partitions of the objects in S such that $\cup_{i=1}^{R} u_i = \cup_{j=1}^{C} v_j = S$ and $u_i \cap u_{i'} = \emptyset = v_j \cap v_{j'}$ for $1 \leq i \neq i' \leq R$ and $1 \leq j \neq j' \leq C$. In our case, one of the partitions is the external criterion and one is a clustering result. Let a be the number of pairs of objects that are placed in the same element in partition U and in the same element in partition V. Also, let d be the number of pairs of objects in different elements in partitions U and V. The Rand

index is simply the fraction of agreement, that is

$$R = \frac{a+d}{\binom{n}{2}}.$$

The Rand index lies between 0 and 1. When the two partitions are identical, the Rand index is 1. A problem with Rand index is that the expected value of the Rand index of two random partitions does not take a constant value. The adjusted Rand index, proposed by Hubert and Arabie [16], corrects for this by assuming the general form

$$R_{\text{adj}} = \frac{R - \text{E}(R)}{\max(R) - \text{E}(R)},$$

where $\text{E}(R)$ is the expected value of the Rand index and $\max(R) = 1$. Similar to the Rand index, a larger adjusted Rand index means a higher correspondence between the two partitions. Calculation of both the Rand index and the adjusted Rand index are available in standard software including R. In our Computer Practical session, we will show examples on how these measures can be used for comparing high dimensional clustering results. The Rand index is particularly useful in simulation studies where the true clusters are known and one can assess the performance of different clustering methods against the truth.

APPENDIX A

Basic concepts and fundamental formulae

This appendix provides some basic concepts and fundamental formulae used in the lecture notes.

Definition A.1. (L_q-norm) For the real numbers $1 \le q < \infty$, the L_q-norm of a vector $\boldsymbol{x} = (x_1, \ldots, x_p)$ is defined as

$$\|\boldsymbol{x}\|_q^1 = \Big(\sum_{j=1}^{p} |x_j|^q \Big)^{1/q}.$$

Note that $\|\boldsymbol{x}\|_q^q = \sum_{j=1}^{p} |x_j|^q$.

Definition A.2. (L_∞-norm) The L_∞-norm (or maximum norm) of a vector $\boldsymbol{x} = (x_1, \ldots, x_p)$ is the limit of L_q-norms when $q \to \infty$, which would be as follows

$$\|\boldsymbol{x}\|_\infty = \max\{|x_1|, \ldots, |x_p|\}.$$

Definition A.3. (Lipschitz function) A function f such that

$$|f(x) - f(y)| \le K|x - y|$$

for all real values x and y, where K is a constant independent of x and y, is called a Lipschitz function. For example, any function with a bounded first derivative must be Lipschitz.

Definition A.4. (General Lipschitz condition) A multivariate function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be K-Lipschitz over region $\mathscr{R} \in \mathbb{R}^n$ if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathscr{R}$

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\|_2^1 \le K\|\boldsymbol{x} - \boldsymbol{y}\|_2^1.$$

Note that a Lipschitz function $f(\boldsymbol{x})$ does not change too drastically with respect to $\boldsymbol{x}$.

Some useful inequalities for norms:

For a p-dimensional vector $\boldsymbol{x}$, we have

$$\|\boldsymbol{x}\|_2^1 \le \|\boldsymbol{x}\|_1^1 \le \sqrt{p}\|\boldsymbol{x}\|_2^1$$
$$\|\boldsymbol{x}\|_\infty \le \|\boldsymbol{x}\|_2^1 \le \sqrt{p}\|\boldsymbol{x}\|_\infty$$
$$\|\boldsymbol{x}\|_\infty \le \|\boldsymbol{x}\|_1^1 \le p\|\boldsymbol{x}\|_\infty,$$

or simply $\|\boldsymbol{x}\|_\infty \le \|\boldsymbol{x}\|_2^1 \le \|\boldsymbol{x}\|_1^1 \le \sqrt{p}\|\boldsymbol{x}\|_2^1 \le p\|\boldsymbol{x}\|_\infty$.

Holder's inequality:

For any two arbitrary vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, we have for all $1 \le q, r \le \infty$

$$\left|\langle \boldsymbol{u}, \boldsymbol{v}\rangle\right| \le \|\boldsymbol{u}\|_q^1 \|\boldsymbol{v}\|_r^1, \qquad \frac{1}{q} + \frac{1}{r} = 1,$$

where $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \boldsymbol{u}^T\boldsymbol{v}$ is the inner product of $\boldsymbol{u}$ and $\boldsymbol{v}$.

Cauchy-Schwarz inequality:

Cauchy-Schwarz inequality is a special case of Holder's inequality with $q = 2$ and $r = 2$, that is,

$$\left|\langle \boldsymbol{u}, \boldsymbol{v}\rangle\right| \le \|\boldsymbol{u}\|_2^1 \|\boldsymbol{v}\|_2^1.$$

Markov inequality:

For any non-decreasing positive function $\phi : \mathbb{R} \to \mathbb{R}^+$ and any real-valued random variable X, we have

$$P(X \ge t) \le \frac{\mathrm{E}\big(\phi(X)\big)}{\phi(t)} \quad \text{for all } t \in \mathbb{R}.$$

In particular, $P(X \ge t) \le \frac{\mathrm{E}(X)}{t}$ for a non-negative random variable X and $t > 0$. Also, for any $\lambda > 0$, we have by setting $\phi(x) = e^{\lambda x}$

$$P(X \ge t) \le e^{-\lambda t}\mathrm{E}(e^{\lambda X}) \quad \text{for all } t \in \mathbb{R}.$$

Tail inequality for Gaussian distribution:

Let Z be a standard Gaussian (normal) random variable. For any constant $c \ge 0$, we have

$$P(Z > c) \le e^{-c^2/2}.$$

For a proof, see `Problem Sheets`.

Matrix differentiation:

Let $\boldsymbol{x}$ be a vector, and further suppose that matrix $\boldsymbol{A}$ and vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are all not functions of $\boldsymbol{x}$. The following matrix differentiation results hold:

$$\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{x}} = \boldsymbol{0}$$

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{I}$$

$$\frac{\partial \boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{A}$$

$$\frac{\partial \boldsymbol{x}^T\boldsymbol{A}}{\partial \boldsymbol{x}} = \boldsymbol{A}^T$$

$$\frac{\partial \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{x}^T\left(\boldsymbol{A}+\boldsymbol{A}^T\right) \quad \text{the row representation}$$

$$\frac{\partial \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}} = \left(\boldsymbol{A}+\boldsymbol{A}^T\right)\boldsymbol{x} \quad \text{the column representation}$$

$$\frac{\partial \boldsymbol{a}^T\boldsymbol{x}\boldsymbol{b}}{\partial \boldsymbol{x}} = \boldsymbol{a}\boldsymbol{b}^T$$

$$\frac{\partial \boldsymbol{a}^T\boldsymbol{x}^T\boldsymbol{b}}{\partial \boldsymbol{x}} = \boldsymbol{b}\boldsymbol{a}^T$$

$$\frac{\partial \boldsymbol{a}^T\boldsymbol{x}^T\boldsymbol{x}\boldsymbol{b}}{\partial \boldsymbol{x}} = \boldsymbol{x}^T\left(\boldsymbol{a}\boldsymbol{b}^T+\boldsymbol{b}\boldsymbol{a}^T\right) \quad \text{the row representation}$$

$$\frac{\partial \boldsymbol{a}^T\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{b}}{\partial \boldsymbol{x}} = \left(\boldsymbol{a}\boldsymbol{b}^T+\boldsymbol{b}\boldsymbol{a}^T\right)\boldsymbol{x} \quad \text{the column representation.}$$

Little o and big O notations:

(a) Let f and g be two real-valued functions. Suppose $g(x)$ is non-zero. We write

$$f(x) = o\big(g(x)\big) \quad \text{as } x \to \infty$$

meaning "$f(x)$ is little o of $g(x)$", if

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = 0.$$

Note that this means $g(x)$ grows much faster than $f(x)$ as $x \to \infty$.

(b) Suppose $g(x)$ is strictly positive for all large enough values of x. We write

$$f(x) = O\big(g(x)\big) \quad \text{as } x \to \infty$$

meaning "$f(x)$ is big O of $g(x)$", if there exists a positive real number M and a real number x_0 such that

$$|f(x)| \le Mg(x) \quad \text{for all } x \ge x_0.$$

Note that this means $|f(x)|$ is bounded above by $g(x)$ as $x \to \infty$.

Little o in probability and big O in probability notations:

(a) Let $\{X_n\}$ be a sequence of random variables and $\{a_n\}$ denote a set of constants. We write

$$X_n = o_P(a_n) \quad \text{as } n \to \infty$$

to mean "X_n/a_n converges to zero in probability" as $n \to \infty$, that is

$$\frac{X_n}{a_n} \xrightarrow{P} 0,$$

or

$$\lim_{n\to\infty} P\Big(\Big|\frac{X_n}{a_n}\Big| > \epsilon\Big) = 0 \quad \text{for all } \epsilon > 0.$$

Note that $X_n = o_P(a_n)$ is equivalent to $X_n/a_n = o_P(1)$.

(b) We write

$$X_n = O_P(a_n) \quad \text{as } n \to \infty$$

to mean the set of values "X_n/a_n is stochastically bounded", that is, for any $\epsilon > 0$ there exist positive real numbers M and N such that

$$P\Big(\Big|\frac{X_n}{a_n}\Big| > M\Big) < \epsilon \quad \text{for all } n > N.$$

Bibliography

[1] Greg W Anderson, Alice Guionnet, and Ofer Zeitouni. *An introduction to random matrices*. Number 118. Cambridge University Press, 2010.

[2] Zhidong D Bai. Methodologies in spectral analysis of large dimensional random matrices, a review. In *Advances in Statistics*, pages 174–240. World Scientific, 2008.

[3] Jinho Baik and Jack W Silverstein. Eigenvalues of large sample covariance matrices of spiked population models. *Journal of Multivariate Analysis*, 97:1382–1408, 2006.

[4] Jinho Baik, Gérard Ben Arous, and Sandrine Péché. Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices. *The Annals of Probability*, 33:1643–1697, 2005.

[5] Peter Bühlmann and Sara Van De Geer. *Statistics for high-dimensional data*. Springer Series in Statistics. Springer, New York, 2011.

[6] Wei-Chien Chang. On using principal components before separating a mixture of two multivariate normal distributions. *Journal of the Royal Statistical Society: Series C*, 32:267–275, 1983.

[7] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the 21st International Conference on Machine Learning*, pages 1–8, 2004.

[8] Julian J Faraway. *Linear models with R, Second Edition*. Chapman and Hall/CRC, 2014.

[9] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*. Springer Science & Business Media, 2012.

[10] Christophe Giraud. *Introduction to high-dimensional statistics*. CRC Press, 2021.

[11] Peter Hall, James Stephen Marron, and Amnon Neeman. Geometric representation of high dimension, low sample size data. *Journal of the Royal Statistical*

Society Series B, 67:427–444, 2005.

[12] John A Hartigan and Manchek A Wong. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society: Series C*, 28:100–108, 1979.

[13] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.

[14] Mohamed Hebiri and Sara van de Geer. The smooth-lasso and other l1+ l2-penalized methods. *Electronic Journal of Statistics*, 5:1184–1226, 2011.

[15] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

[16] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[17] Ian T Jolliffe, Nickolay T Trendafilov, and Mudassir Uddin. A modified principal component technique based on the lasso. *Journal of Computational and Graphical Statistics*, 12:531–547, 2003.

[18] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[19] P Macnaughton-Smith, WT Williams, MB Dale, and LG Mockett. Dissimilarity analysis: a new technique of hierarchical sub-division. *Nature*, 202:1034–1035, 1964.

[20] Vladimir Alexandrovich Marchenko and Leonid Andreevich Pastur. Distribution of eigenvalues for some sets of random matrices. *Matematicheskii Sbornik*, 114: 507–536, 1967.

[21] Mohammad Sal Moslehian. Ky Fan inequalities. *Linear and Multilinear Algebra*, 60:1313–1325, 2012.

[22] Chandra Sekhar Mukherjee and Jiapeng Zhang. Compressibility: Power of PCA in clustering problems beyond dimensionality reduction. *arXiv preprint arXiv:2204.10888*, 2022.

[23] Debashis Paul. Asymptotics of the leading sample eigenvalues for a spiked covariance model. 2006. Available online at “http://anson.ucdavis.edu/ debashis/techrep/eigenlimit.pdf”.

[24] Debashis Paul. Asymptotics of sample eigenstructure for a large dimensional spiked covariance model. *Statistica Sinica*, pages 1617–1642, 2007.

[25] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine, Series 6*, 2:559–572, 1901.

[26] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66:846–850, 1971.

[27] Karl Sjostrand, Egill Rostrup, Charlotte Ryberg, Rasmus Larsen, Colin Studholme, Hansjoerg Baezner, Jose Ferro, Franz Fazekas, Leonardo Pantoni, Domenico Inzitari, et al. Sparse decomposition and modeling of anatomical shape variation. *IEEE Transactions on Medical Imaging*, 26:1625–1635, 2007.

[28] Terence Tao. *Topics in random matrix theory*. American Mathematical Society, 2023.

[29] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58:267–288, 1996.

[30] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B*, 63:411–423, 2001.

[31] Ka Yee Yeung and Walter L Ruzzo. An empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 17:763–774, 2001.

[32] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B*, 67:301–320, 2005.

[33] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15:265–286, 2006.