PIOTR ZWIERNIK

METHODS FOR MULTIVARIATE DATA (STA437)

Introduction

0.0.1 About this course

These are rudimentary notes for STA437/2005 that contain more technical part of the lecture that will be presented in class on blackboard. Some multivariate methods will be presented on slides and so not treated in detail here.

Traditionally multivariate statistics is taught with a lot of focus on the multivariate normal distribution and the underlying statistical inference. I will be more brief on this part trying to get some basic overview of more modern methods for analyzing multivariate data.

My idea for this course is to mix two styles: (i) give a deeper theoretical insight into the underlying linear algebra and canonical multivariate distributions/models, (ii) give some taste for modern state-of-the art methods. For this reason, the lecture will be mostly given on blackboard but occasionally I will present some slides to give a more high-level overview of more recent material. The tutorials will also mix theory and practical considerations.

0.0.2 Prerequisites

To understand better what you are expected to know, note that one of the prerequisites for STA437 is STA302. A prerequisite for STA302 is MAT223 or MAT224 or MAT240. Students should have knowledge of linear algebra from these courses and its applications to linear models from STA302.

Multivariate statistics very heavily relies on matrix algebra and, more generally, on linear algebra. I will try to briefly recall some key facts when we need them but there will be no time to discuss these results in detail so they are assumed. Please make sure you have all the right background. There are no shortcuts here. The more you understand linear algebra, the more successful you will be in understanding this course and many other data analysis methods you will encounter in the future. The material gathered in Appendix A of [1] should be more than enough. For deeper understanding check, for example, Chapter I in [2] or Jason Siefken's course book for MAT223.

[1] Kanti V Mardia, John T Kent, and Charles C Taylor. *Multivariate analysis*, volume 88. John Wiley & Sons, 2024

[2] Gilbert Strang. *Linear Algebra and Learning from Data*. Wellesley-Cambridge Press, Wellesley, MA, 2019. ISBN 9780692196380

0.0.3 Complementary courses

There are several courses that are complementary to STA437/STA2005. For example, it is natural to discuss some classification algorithms in the context of multivariate statistics. However, this is covered in detail in STA314 so we will mostly omit this here. Some important examples are k-means, hierarchical clustering, and linear discriminant analysis[3]. Another related course, STA414/2104, focuses on unsupervised learning and we recommend students to take this course to get a full picture where the methods we learn here can be applied.

0.0.4 Computational statistics

Although the lecture focuses on theory and methods, the computational aspect of this course is critical for building and verifying the correct intuition. Occasionally, we provide `R` code for your convenience. If you are not fluent in `R` , I recommend using AI to generate complementary code or to translate it into Python or Julia.

To report typos/mistakes, write me: `piotr.zwiernik@utoronto.ca`

I would like to thank Luis Sierra Muntané for his help with improving the notes.

Last update: January 7, 2025

[3] We will talk about Gaussian mixtures, which give access to classification methods that, in a sense, generalize both k-means and LDA.

Contents

1
Preliminaries

We start by recalling basic notation and facts. This part of the lecture notes is quite fast-paced because we refresh material that should be known from earlier courses. Please fill out all the details.

1.1 Basic Linear Algebra

We use standard notations such as $\mathbb{R}^m$ to denote the set of all m-dimensional real vectors $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbb{R}^{n\times m}$ to denote the set of all $n \times m$ real matrices $A = (A_{ij})$. All vectors in $\mathbb{R}^m$ are treated as column vectors, that is, we take $\mathbb{R}^m = \mathbb{R}^{m\times 1}$.

Recall that a matrix $A \in \mathbb{R}^{n\times m}$ represents a linear function from $\mathbb{R}^m$ to $\mathbb{R}^n$. For every $\mathbf{x} \in \mathbb{R}^m$, the vector $A\mathbf{x}$ lies in $\mathbb{R}^n$. Checking that the function $f(\mathbf{x}) = A\mathbf{x}$ is linear should be straightforward[1].

1.1.1 Matrix-Vector Multiplication $A\mathbf{x}$

Let $\mathbf{x} \in \mathbb{R}^m$ be a vector, with $\mathbf{x} = (x_1, \ldots, x_m)$, and let $A \in \mathbb{R}^{n\times m}$ be a matrix. Recall that the matrix-vector product $A\mathbf{x}$ is defined by the rule:

$$(A\mathbf{x})_i = \sum_{j=1}^{m} A_{ij} x_j,$$

which states that the i-th entry of the vector $A\mathbf{x} \in \mathbb{R}^n$ is the inner product of the i-th row of A with the vector $\mathbf{x}$.

An important alternative interpretation of $A\mathbf{x}$ is that it is a linear combination of the columns of A with coefficients given by the entries of $\mathbf{x}$:

$$A\mathbf{x} = x_1\mathbf{a}_1 + \cdots + x_m \boldsymbol{a}_m, \tag{1.1}$$

where $\mathbf{a}_j \in \mathbb{R}^m$ denotes the j-th column of A.

← Exercise 1.4.5

Example 1.1.1. *In the least squares problem from linear regression, we have a data matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$ with rows $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and a vector of observations $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$. The least squares estimator $\hat{\beta} \in \mathbb{R}^p$ minimizes*

Recall that $\|\mathbf{x}\| = \sqrt{\mathbf{x}^\top \mathbf{x}}$ is the Euclidean distance.

[1] Make sure you see why.

$$\|\mathbf{y} - \mathbf{X}\beta\|^2 \;=\; \sum_{i=1}^{n}(y_i - \mathbf{x}_i^\top \beta)^2,$$

where $\mathbf{X}\beta$ *represents the prediction. The interpretation of* $\mathbf{X}\beta$ *as a linear combination of the columns of* $\mathbf{X}$ *helps us understand that the least squares problem finds the point in the column space of* $\mathbf{X}$ *that is closest to* $\mathbf{y}$*.*

Example 1.1.2. *Let* $\boldsymbol{e}_1, \dots, \boldsymbol{e}_m \in \mathbb{R}^m$ *be the canonical unit basis. We have* $A\boldsymbol{e}_i = \boldsymbol{a}_i$ *for* $i = 1, \dots, m$ *and so* A *maps the canonical unit basis of* $\mathbb{R}^m$ *to* $\boldsymbol{a}_1, \dots, \boldsymbol{a}_m$*. If* $m = n$ *and the columns of* A *are linearly independent, the vectors* $\boldsymbol{a}_1, \dots, \boldsymbol{a}_m$ *also form a basis of* $\mathbb{R}^m$ *and so* A *represents a change of basis.*

1.1.2 *Rank-One Matrix* $\mathbf{x}\mathbf{y}^\top$

A **rank-one matrix** is formed by the outer product of two vectors. Specifically, if $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$, the outer product $\mathbf{x}\mathbf{y}^\top \in \mathbb{R}^{n\times m}$ is defined as:

$$A = \mathbf{x}\mathbf{y}^\top \quad \text{with} \quad A_{ij} = x_i y_j \quad \text{for all } i = 1, \dots, n,\ j = 1, \dots, m.$$

The resulting matrix has rank[2] at most one because all rows (or columns) of the matrix are scalar multiples of each other.

1.1.3 *Matrix Multiplication* AB

Matrix multiplication extends the concept of matrix-vector multiplication to two matrices. Let $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{m\times p}$. The product $AB \in \mathbb{R}^{n\times p}$ is defined by the rule:

$$(AB)_{ik} = \sum_{j=1}^{m} A_{ij}B_{jk},$$

which means that the (i,k)-th entry of AB is the inner product of the i-th row of A with the k-the column of B.

Alternatively, we can view matrix multiplication as a sum of rank-one matrices. Let $\mathbf{a}_1, \dots, \mathbf{a}_m$ be the columns of A and $\mathbf{b}_1, \dots, \mathbf{b}_m$ be the rows of B. Then:

$$AB \;=\; \sum_{j=1}^{m} \mathbf{a}_j \mathbf{b}_j^\top, \tag{1.2}$$

which expresses AB as the sum of rank-one matrices formed from the columns of A and the rows of B.

← Exercise 1.4.8

Example 1.1.3. *Suppose* $\mathbf{X} \in \mathbb{R}^{n\times p}$ *is a data matrix with rows* $\mathbf{x}_1, \dots, \mathbf{x}_n$*. Then the matrix* $\frac{1}{n}\mathbf{X}^\top\mathbf{X}$ *can be written as:*

$$\frac{1}{n}\mathbf{X}^\top\mathbf{X} \;=\; \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top.$$

[2] Recall that the rank of A is the number of linearly independent rows (or equiv. columns).

This matrix will play an important role in this course. Note that if the rows of $\mathbf{X}$ *are independent and identically distributed (i.i.d.) samples from some distribution with mean zero and covariance matrix* $\Sigma = \mathbb{E}[\mathbf{x}_i\mathbf{x}_i^\top]$*, then by the law of large numbers, we have:*

Here $\xrightarrow{\text{p}}$ denotes convergence in distribution.

$$\frac{1}{n}\mathbf{X}^\top\mathbf{X} \;=\; \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top \;\xrightarrow{p}\; \Sigma \quad as \quad n \to \infty.$$

1.1.4 Singular Value Decomposition (SVD)

The **singular value decomposition (SVD)** is one of the most important linear algebra tools in this course. It decomposes any matrix into a product of three matrices: two orthogonal matrices and a diagonal matrix.

Denote by $O(m)$ the set of orthogonal $m \times m$ matrices:

Defining equation $UU^\top = I_m$ implies that $U^{-1} = U^\top$ and so $U^\top U = I_m$ too.

$$O(m) = \{U \in \mathbb{R}^{m\times m} : UU^\top = I_m\}.$$

An orthogonal matrix has the property that its rows (and columns) form an orthonormal basis for $\mathbb{R}^m$. Note that if $U \in O(m)$ then $\det(U) = \pm 1$, which follows from multiplicativity of the determinant and the fact that $\det(I_m) = 1$. An orthogonal matrix U is called a rotation matrix if $\det(U) = 1$ but this terminology is often used for all orthogonal matrices.

Theorem 1.1.4 (Singular Value Decomposition)**.** *Let* $A \in \mathbb{R}^{n\times m}$ *with* $n \geq m$. [3] *Then there exist orthogonal matrices* $U \in O(n)$ *and* $V \in O(m)$ *such that:*

$$A = UDV^\top,$$

where $D \in \mathbb{R}^{n\times m}$ *is a "diagonal" matrix with non-negative entries on the diagonal:*

$\mathbf{0}_n \in \mathbb{R}^n$ is the zero vector and $\mathbf{1}_n \in \mathbb{R}^n$ is the vector of ones.

$$D = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_m \\ \mathbf{0}_{n-m} & \cdots & \mathbf{0}_{n-m} \end{bmatrix}, \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0.$$

The values $\sigma_1, \ldots, \sigma_m$ *are called the* ***singular values*** *of* A*. The singular values are always uniquely defined.*

Denote the columns of U by $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ and the columns of V by $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$. Note that, by any $i = 1, \ldots, m$,

$$A\boldsymbol{v}_i \;=\; UD\boldsymbol{e}_i \;=\; \sigma_i U\boldsymbol{e}_i \;=\; \sigma_i\boldsymbol{u}_i$$

and so each $\boldsymbol{v}_i$ is transformed into a scaled version of $\boldsymbol{u}_i$, $i = 1, \ldots, m$. We refer to $\boldsymbol{v}_i$'s are right singular vectors and to $\boldsymbol{u}_i$'s as the left singular vectors.

[3] The case $n \leq m$ can be treated analogously.

The singular value decomposition expresses A as a sum of rank-one matrices:

$$A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top + \cdots + \sigma_m \mathbf{u}_m \mathbf{v}_m^\top.$$

Note that $\mathbf{u}_{m+1}, \ldots, \mathbf{u}_n$ do not feature here so they are pretty much irrelevant.

1.1.5 Spectrum of a symmetric matrix

Let $A \in \mathbb{R}^{m \times m}$ be a square matrix. A non-zero vector $\mathbf{v} \in \mathbb{R}^m$ is called an **eigenvector** of A if there exists a scalar $\lambda \in \mathbb{R}$ such that:

$$A\mathbf{v} = \lambda \mathbf{v}. \tag{1.3}$$

Think about a geometric meaning of this equation.

In this case, λ is called an **eigenvalue** of A corresponding to the eigenvector $\mathbf{v}$.

The matrix A has at most m eigenvalues (counting multiplicities). The eigenvalues are the real roots of the characteristic polynomial $\det(A - \lambda I_m)$.[4]

← Exercise 1.4.7

Denote by $\mathbb{S}^m$ the set of all $m \times m$ symmetric matrices.

$\mathbb{S}^m = \{A \in \mathbb{R}^{m \times m} : A = A^\top\}$

Lemma 1.1.5. *If $A \in \mathbb{S}^m$, then all its eigenvalues are always real, and the eigenvectors corresponding to distinct eigenvalues are orthogonal.*

Proof. Since $A \in \mathbb{S}^m$, it is symmetric, so $A = A^\top$. Suppose $A\mathbf{v} = \lambda\mathbf{v}$, where $\lambda = a + ib$ with $a, b \in \mathbb{R}$ (i.e., assume for the sake of contradiction that λ could be complex). Recall that the standard inner product in $\mathbb{C}^m$ is $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^m x_i \bar{y}_i$, where $\bar{y}$ is the complex conjugate of y (if $y = a + bi$ then $\bar{y} = a - bi$). If $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ then $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y}$ and so

$$\mathbf{v}^\top A \mathbf{v} = \langle v, Av \rangle = \langle v, \lambda v \rangle = \overline{\lambda} \|\mathbf{v}\|^2.$$

But, by the symmetry of A,

$$\mathbf{v}^\top A \mathbf{v} = (A\mathbf{v})^\top \mathbf{v} = \langle Av, v \rangle = \langle \lambda v, v \rangle = \lambda \|\mathbf{v}\|^2.$$

This implies

$$\lambda \|\mathbf{v}\|^2 = \overline{\lambda} \|\mathbf{v}\|^2.$$

Since $\|\mathbf{v}\|^2 \neq 0$, it follows that $\lambda = \overline{\lambda}$, which implies that $\lambda \in \mathbb{R}$. Therefore, all eigenvalues of A are real.

Now, for the orthogonality of eigenvectors corresponding to distinct eigenvalues, let $\mathbf{v}_1$ and $\mathbf{v}_2$ be eigenvectors corresponding to distinct eigenvalues λ_1 and λ_2 of A. Then

$$A\mathbf{v}_1 = \lambda_1 \mathbf{v}_1 \quad \text{and} \quad A\mathbf{v}_2 = \lambda_2 \mathbf{v}_2.$$

Taking the inner product of the first equation with $\mathbf{v}_2$ and the second with $\mathbf{v}_1$, we get

$$\mathbf{v}_2^\top A \mathbf{v}_1 = \lambda_1 \mathbf{v}_2^\top \mathbf{v}_1 \quad \text{and} \quad \mathbf{v}_1^\top A \mathbf{v}_2 = \lambda_2 \mathbf{v}_1^\top \mathbf{v}_2.$$

[4] We can extend this definition to complex eigenvalues and eigenvectors. The fundamental theorem of algebra tells us that the characteristic polynomial always has m complex roots (counting with multiplicities). This is not so important for our course.

Since A is symmetric, $\mathbf{v}_2^\top A\mathbf{v}_1 = \mathbf{v}_1^\top A\mathbf{v}_2$. Therefore,

$$\lambda_1 \mathbf{v}_2^\top \mathbf{v}_1 = \lambda_2 \mathbf{v}_1^\top \mathbf{v}_2.$$

Since $\lambda_1 \neq \lambda_2$, it must be that $\mathbf{v}_1^\top \mathbf{v}_2 = 0$, so $\mathbf{v}_1$ and $\mathbf{v}_2$ are orthogonal. This completes the proof. □

This directly gives the following fundamental result.

Theorem 1.1.6 (Spectral Theorem for Symmetric Matrices)**.** *If $A \in \mathbb{S}^m$, then there exists an orthogonal matrix $U \in O(m)$ and a diagonal matrix $\Lambda \in \mathbb{R}^{m\times m}$ such that:*

$$A = U\Lambda U^\top,$$

where the diagonal entries of Λ, denoted by $\lambda_1, \ldots, \lambda_m$, are the eigenvalues of A. The columns of U, denoted by $\mathbf{u}_1, \ldots, \mathbf{u}_m$, are the corresponding eigenvectors of A, i.e.,

$$A\mathbf{u}_i = \lambda_i \mathbf{u}_i \quad \textit{for all } i = 1, \ldots, m.$$

The spectral theorem provides a full diagonalization of a symmetric matrix, allowing us to express it as a sum of projections onto its eigenvectors:

$$A = \sum_{i=1}^{m} \lambda_i \mathbf{u}_i \mathbf{u}_i^\top.$$

Question: In what sense is the matrix $\mathbf{u}_i\mathbf{u}_i^\top$ a projection matrix?

Each term in this sum is a rank-one matrix corresponding to an eigenvalue and its associated eigenvector.

1.1.6 Positive definite matrices and quadratic forms

Every symmetric matrix $A \in \mathbb{S}^p$ defines a quadratic form

$$q(\mathbf{x}) \;=\; \mathbf{x}^\top A\mathbf{x} \;=\; \sum_{i,j=1}^{p} A_{ij}x_i x_j.$$

By definition, A is called **positive semi-definite (PSD)** if $q(\mathbf{x}) \geq 0$ for all $\mathbf{x}$. If the inequality is strict for all non-zero $\mathbf{x}$, then A is **positive definite (PD)**. The set of PD matrices is denoted by $\mathbb{S}_+^m$ and the set of PSD matrices is denoted by $\overline{\mathbb{S}}_+^m$.

Notation $\overline{\mathbb{S}}_+^m$ suggests that the set of PSD matrices is the Euclidean closure of the set of PD matrices, which is indeed true.
← Exercise 1.4.15

1.2 Random vectors and matrices

This section covers the fundamental concepts of random vectors, random matrices, and their associated mean and covariance structures.

1.2.1 Mean and Covariance

Let $X = (X_1, \dots, X_m) \in \mathbb{R}^m$ be a random vector with a probability density function $f(\mathbf{x})$. The expectation (or mean) of a scalar-valued function $g(X)$ is defined as:

For discrete distributions we can take f to be the probability mass function and replace integration with summation.

$$\mathbb{E}[g(X)] = \int_{\mathbb{R}^m} g(\mathbf{x}) f(\mathbf{x}) \, d\mathbf{x}.$$

We assume that all necessary integrals converge, ensuring that the expectation is finite.

Now, consider a random vector $X = (X_1, \dots, X_m) \in \mathbb{R}^m$ or a random matrix $S = (S_{ij}) \in \mathbb{R}^{n \times m}$. When we refer to $\mathbb{E}[X]$ or $\mathbb{E}[S]$, we mean that the expectation is applied element-wise:

$$\mathbb{E}[X] = (\mathbb{E}[X_i])_{i=1}^m, \quad \mathbb{E}[S] = (\mathbb{E}[S_{ij}])_{i,j}.$$

We denote the mean vector of $X \in \mathbb{R}^m$ as:

$$\mu = \mathbb{E}[X] = (\mathbb{E}[X_1], \dots, \mathbb{E}[X_m]) = (\mu_1, \dots, \mu_m).$$

The covariance matrix of X, denoted by Σ, is defined as:

$$\Sigma = \mathrm{var}(X) = \mathbb{E}[(X - \mu)(X - \mu)^\top].$$

The diagonal elements of Σ represent the variances of the individual components of X:

$$\Sigma_{jj} = \mathbb{E}[(X_j - \mu_j)^2] = \mathrm{var}(X_j),$$

while the off-diagonal elements represent the covariances between the components:

$$\Sigma_{jk} = \mathbb{E}[(X_j - \mu_j)(X_k - \mu_k)] = \mathrm{cov}(X_j, X_k).$$

Recall from Section 1.1.6 the definition of PSD matrices. The covariance matrix Σ is always positive semi-definite. Indeed, for any vector $\mathbf{u} \in \mathbb{R}^m$, we have:

$$\mathbf{u}^\top \mathrm{var}(X) \mathbf{u} = \sum_{i,j} u_i u_j \mathbb{E}(X_i - \mu_i)(X_j - \mu_j) = \mathbb{E}[\sum_i u_i (X_i - \mu_i) \sum_j u_j (X_j - \mu_j)] \;=\; \mathbb{E}(u^\top (X - \mu))^2,$$

which is always nonnegative. The covariance matrix Σ is positive definite unless there is a linear combination of the components of X that is degenerate.

Example 1.2.1. *Consider a covariance matrix $\Sigma \in \mathbb{S}^m$. Since covariance matrices are symmetric and positive semi-definite, they admit a spectral decomposition:*

$$\Sigma = U \Lambda U^\top,$$

*where $\Lambda = \mathit{diag}(\lambda_1, \dots, \lambda_m)$ contains the eigenvalues (which are nonnegative). This decomposition is useful in multivariate statistics, for example, when performing **Principal Component Analysis (PCA)**.*

More generally, for two random vectors $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}^q$, the covariance matrix between X and Y is given by:

$$\mathrm{cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])^\top] \in \mathbb{R}^{p \times q}.$$

In particular, the variance of X is just the covariance of X with itself: $\mathrm{var}(X) = \mathrm{cov}(X, X)$.

1.2.2 Linearity of Expectation and Bilinearity of Covariance

The following properties of expectation are straightforward but important. They can be verified by checking each coordinate:

Lemma 1.2.2. *Let $X \in \mathbb{R}^m$ be a random vector, and $S \in \mathbb{R}^{p \times q}$ be a random matrix. Let $\mathbf{b} \in \mathbb{R}^n$ be a vector and let $A \in \mathbb{R}^{n \times p}$, $B \in \mathbb{R}^{q \times m}$ be matrices. Then*

$$\mathbb{E}[AX + \mathbf{b}] = A\mathbb{E}[X] + \mathbf{b}.$$

Moreover,

$$\mathbb{E}[ASB] = A\mathbb{E}[S]B.$$

Proof. For the first equality, we check it on each coordinate. The i-th entry on the left is

$$\mathbb{E}(\sum_{j=1}^{m} A_{ij} X_j + b_i) \;=\; \sum_{j=1}^{m} A_{ij}\mathbb{E}(X_j) + b_i,$$

which is the same as the i-th entry of $A\mathbb{E}[X] + \boldsymbol{b}$ on the right. For the second equality, we see that the (i, j)-th entry of the matrix on the left is

$$\mathbb{E}[\sum_{k,l} A_{ik} S_{k,l} B_{l,j}] \;=\; \sum_{k,l} A_{ik}\mathbb{E}[S_{k,l}] B_{l,j}$$

with the same conclusion as before. □

For covariance matrices we have the following ($X \in \mathbb{R}^p, Y \in \mathbb{R}^q$):

1. $\mathrm{cov}(X, Y) = \mathrm{cov}(Y, X)^\top$ (symmetry).
2. $\mathrm{cov}(aX + bX', Y) \;=\; a\,\mathrm{cov}(X, Y) + b\,\mathrm{cov}(X', Y)$ for any scalars $a, b \in \mathbb{R}$ (bilinearity).

Both properties follow easily from known properties in the univariate case, which can be checked on each coordinate. More generally, we get the following lemma.

Lemma 1.2.3. *Consider random vectors $X \in \mathbb{R}^p$, $Y \in \mathbb{R}^q$ and matrices $A \in \mathbb{R}^{n \times p}$, $B \in \mathbb{R}^{q \times m}$, then:*

$$\mathrm{cov}(AX, BY) = A\,\mathrm{cov}(X, Y)B^\top$$

and

$$\mathrm{var}(AX) \;=\; A\,\mathrm{var}(X)A^\top.$$

Exercise 1.2.4. *Prove this lemma. Hint: Use the definition of the covariance and Lemma 1.2.2.*

1.3 Sample Quantities

We now turn to the sample analogs of the mean and covariance matrix, which are important when working with observed data. Suppose we observe an i.i.d. sample $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^m$. We collect these observed vectors into a data matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, where each row corresponds to one observation.

Two sample statistics are particularly important: the sample mean and the sample covariance matrix.

1.3.1 Sample Mean

The sample mean vector is defined as:

$$\bar{\mathbf{x}}_n \;=\; \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \;=\; \frac{1}{n}\mathbf{X}^\top \mathbf{1}_n,$$

where $\mathbf{1}_n$ is the vector of all ones in $\mathbb{R}^n$. This vector summarizes the central location of the sample.

1.3.2 Sample Covariance Matrix

The sample covariance matrix is an estimator of the population covariance matrix. It measures the variability and linear relationships between the components of the random vector across the sample. It is defined as:

$$S_n = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}}_n)(\mathbf{x}_i - \bar{\mathbf{x}}_n)^\top. \qquad (1.4)$$

We define the **centering matrix** as the matrix

$$H \;=\; I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top.$$

The matrix H is symmetric, idempotent ($H^2 = H$), and projects any vector in $\mathbb{R}^n$ onto the space orthogonal to the vector $\mathbf{1}_n$. This matrix is used to remove the effect of the sample mean from the data and it gives us a useful alternative formulation of S_n.

← Exercise 1.4.19

Lemma 1.3.1. *The sample covariance matrix can be written as:*

$$S_n = \frac{1}{n}\mathbf{X}^\top H \mathbf{X}.$$

Proof. The rows of the matrix

$$\mathbf{X} - \mathbf{1}_n \bar{\mathbf{x}}_n^\top \;=\; \mathbf{X} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top \mathbf{X} \;=\; H\mathbf{X}$$

are precisely $\mathbf{x}_i - \bar{\mathbf{x}}_n$. Using Subsection 1.1.3, we see that S_n defined in (1.4) has a matrix representation $\frac{1}{n}\mathbf{X}^\top H^\top H\mathbf{X}$, which concludes the proof by the fact that H is symmetric and idempotent. □

Exercise 1.3.2. *Show that the kernel of H is the span of $\mathbf{1}_n$. Conclude that rank$(H) = n - 1$.*

Recall that the rank of a matrix is the maximal number of linearly independent row/columns.

In statistical software like `R`, the function `scale(X, center = TRUE, scale = FALSE)` can be used to center the data matrix $\mathbf{X}$ by subtracting the column means.

1.3.3 Properties of Sample Statistics

Now, suppose that the sample $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^m$ comes from a population with the same distribution as the random vector $X \in \mathbb{R}^m$. As before, we collect the sample as rows in the data matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$. Suppose that $\mathbb{E}[X] = \mu$ and $\operatorname{var}(X) = \Sigma$ are the population mean and covariance, respectively.

We now examine some properties of the sample statistics.

Lemma 1.3.3. *The sample mean $\bar{\mathbf{x}}_n$ is an unbiased estimator of the population mean:*

$$\mathbb{E}[\bar{\mathbf{x}}_n] = \mu.$$

Proof. By the linearity of expectation:

$$\mathbb{E}[\bar{\mathbf{x}}_n] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\right] = \frac{1}{n}\sum_{i=1}^n \mathbb{E}[\mathbf{x}_i] = \frac{1}{n}\sum_{i=1}^n \mu = \mu.$$

□

Next, we compute the variance of the sample mean.

Lemma 1.3.4. *The variance of the sample mean is:*

$$\operatorname{var}(\bar{\mathbf{x}}_n) = \frac{1}{n}\Sigma.$$

Proof. Since the components of the sample are i.i.d., we can apply the bilinearity of covariance:

$$\begin{aligned}\operatorname{var}\left(\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\right) &= \operatorname{cov}\left(\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i, \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\right) = \tfrac{1}{n^2}\sum_{i,j}\operatorname{cov}(\mathbf{x}_i, \mathbf{x}_j) \\ &= \tfrac{1}{n^2}\sum_{i=1}^n \operatorname{cov}(\mathbf{x}_i, \mathbf{x}_i) = \frac{1}{n^2}\sum_{i=1}^n \operatorname{var}(\mathbf{x}_i) = \frac{1}{n^2}\cdot n\Sigma = \frac{1}{n}\Sigma\end{aligned}$$

Note that $\operatorname{cov}(\mathbf{x}_i, \mathbf{x}_j) = 0$ by independence.

□

These two lemmas show that $\bar{\mathbf{x}}_n$ is a decent estimator of μ and its distribution quickly concentrates around μ. To show one version of this concentration phenomenon, recall the most basic probabilistic inequality.

← Exercise 1.4.20

Lemma 1.3.5 (Markov inequality). *If Z is a non-negative random variable, then*

$$\mathbb{P}(Z \geq t) \;\leq\; \frac{\mathbb{E}Z}{t}.$$

Proof. We have[5]

$$t\mathbb{P}(Z \geq t) \;=\; \mathbb{E}[t\mathbb{1}(Z \geq t)] \;\leq\; \mathbb{E}[Z\mathbb{1}(Z \geq t)] \;\leq\; \mathbb{E}Z,$$

where the last inequality follows because $Z \geq 0$. □

Proposition 1.3.6. *The sample mean satisfies the following concentration inequality*

$$\mathbb{P}(\|\bar{\mathbf{x}}_n - \mu\| \geq t) \;\leq\; \frac{\operatorname{tr}(\Sigma)}{nt^2}.$$

Proof. Consider $Y = \|\bar{\mathbf{x}}_n - \mu\|$ so that

$$Y^2 = \|\bar{\mathbf{x}}_n - \mu\|^2 = (\bar{\mathbf{x}}_n - \mu)^\top(\bar{\mathbf{x}}_n - \mu) \;=\; \operatorname{tr}((\bar{\mathbf{x}}_n - \mu)(\bar{\mathbf{x}}_n - \mu)^\top).$$

We use the fact that $\mathbb{R} = \mathbb{R}^{1\times 1}$ and so $(\bar{\mathbf{x}}_n - \mu)^\top(\bar{\mathbf{x}}_n - \mu) = \operatorname{tr}((\bar{\mathbf{x}}_n - \mu)^\top(\bar{\mathbf{x}}_n - \mu))$. Moreover $\operatorname{tr}(AB) = \operatorname{tr}(BA)$ whenever AB and BA are both well defined.

We have

$$\mathbb{E}Y^2 \;=\; \operatorname{tr}(\mathbb{E}((\bar{\mathbf{x}}_n - \mu)(\bar{\mathbf{x}}_n - \mu)^\top)) \;=\; \operatorname{tr}(\operatorname{var}(\bar{\mathbf{x}}_n)) \;=\; \tfrac{1}{n}\operatorname{tr}(\Sigma).$$

Thus, using the Markov inequality, we get

$$\mathbb{P}(\|\bar{\mathbf{x}}_n - \mu\| \geq t) \;=\; \mathbb{P}(Y^2 \geq t^2) \;\leq\; \frac{\operatorname{tr}(\Sigma)}{nt^2}.$$

□

This is only a very simple concentration bound which works under minimal assumptions. Much sharper bounds can be found is the distribution of the sample is more structured.

Finally, we consider the bias of the sample covariance matrix.

Lemma 1.3.7. *The sample covariance matrix S_n is a biased estimator of the population covariance matrix Σ. Its expectation is:*

$$\mathbb{E}[S_n] = \frac{n-1}{n}\Sigma.$$

Proof. Expanding S_n in terms of $\mathbf{x}_i - \mu$ and $\bar{\mathbf{x}}_n - \mu$, we get

$$S_n = \frac{1}{n}\sum_{i=1}^{n}\left((\mathbf{x}_i - \mu) - (\bar{\mathbf{x}}_n - \mu)\right)\left((\mathbf{x}_i - \mu) - (\bar{\mathbf{x}}_n - \mu)\right)^\top.$$

Expanding this expression, we find:

$$S_n = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^\top - (\bar{\mathbf{x}}_n - \mu)(\bar{\mathbf{x}}_n - \mu)^\top.$$

Taking expectations, we get

$$\mathbb{E}[S_n] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^\top] - \mathbb{E}[(\bar{\mathbf{x}}_n - \mu)(\bar{\mathbf{x}}_n - \mu)^\top].$$

[5] We use the fact that $\mathbb{E}\mathbb{1}\{X \in A\} = \mathbb{P}(X \in A)$

Since $\mathbb{E}[(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^\top] = \Sigma$ and $\mathbb{E}[(\bar{\mathbf{x}}_n - \mu)(\bar{\mathbf{x}}_n - \mu)^\top] = \frac{\Sigma}{n}$, we have

$$\mathbb{E}[S_n] \;=\; \Sigma - \frac{1}{n}\Sigma \;=\; \frac{n-1}{n}\Sigma.$$

Simplifying, we find

$$\mathbb{E}[S_n] = \frac{n}{n-1}\Sigma - \frac{\Sigma}{n-1} = \frac{n-1}{n}\Sigma.$$

□

This result shows that S_n is a biased estimator of Σ with $\mathbb{E}(S_n - \Sigma) = -\frac{1}{n}\Sigma$, which is asymptotically negligible.

1.3.4 Correlation and Sample Correlation

In many cases, it is useful to measure the linear relationship between the components of a random vector using the **correlation** rather than the covariance. The correlation matrix normalizes the covariance matrix so that all diagonal elements are equal to 1.

If Σ is a covariance matrix, we define the diagonal matrix $D_\Sigma = \text{diag}(\Sigma_{11}, \ldots, \Sigma_{mm})$. The **correlation matrix** R is given by:

$$R = D_\Sigma^{-1/2} \Sigma D_\Sigma^{-1/2}.$$

The elements of R are:

$$R_{ij} = \begin{cases} 1 & \text{if } i = j, \\ \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}} & \text{if } i \neq j. \end{cases}$$

The correlation matrix measures the strength of the linear relationship between pairs of variables, with R_{ij} taking values in the range $[-1, 1]$. The value $R_{ij} = 1$ indicates perfect positive correlation, while $R_{ij} = -1$ indicates perfect negative correlation.

← Exercise 1.4.21

The **sample correlation matrix** is defined in the same way as the population correlation matrix, but using the sample covariance matrix S_n instead of Σ. That is, the sample correlation matrix $\hat{R}$ is given by:

$$\hat{R} = D_{S_n}^{-1/2} S_n D_{S_n}^{-1/2},$$

where D_{S_n} is the diagonal matrix of the variances of the components in the sample.

The correlation matrix is often preferred over the covariance matrix when the variables have different units or scales, as it provides a dimensionless measure of linear dependence.

1.4 Exercises

In this section we provide some additional exercises that will help you to refresh the basic concepts and practice the new ones.

Exercise 1.4.1. *Consider a matrix $A \in \mathbb{R}^{3\times 3}$ and a vector $\mathbf{x} = (1, 2, 3) \in \mathbb{R}^3$. Let*

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 5 & 6 & 0 \end{pmatrix}.$$

Compute $A\mathbf{x}$ using both interpretations of matrix-vector multiplication: (i) by taking inner products of rows with $\mathbf{x}$, and (ii) as a linear combination of the columns of A.

Exercise 1.4.2. *Let $\mathbf{x} = (1, 2, 3) \in \mathbb{R}^3$ and $\mathbf{y} = (4, 5, 6) \in \mathbb{R}^3$. Compute the matrix $\mathbf{x}\mathbf{y}^\top$. Argue why this matrix has rank one. Compute $\operatorname{tr}(\mathbf{x}\mathbf{y}^\top)$ and $\mathbf{x}^\top\mathbf{y}$. Is it a coincidence that these two numbers are equal?*

Exercise 1.4.3. *Suppose $A = \mathbf{u}\mathbf{v}^\top$ is a rank-one matrix where $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{v} \in \mathbb{R}^m$. Show that for any vector $\mathbf{x} \in \mathbb{R}^m$, the vector $A\mathbf{x}$ is a scalar multiple of $\mathbf{u}$. What does this imply about the image of A?*

Exercise 1.4.4. *Let $A \in \mathbb{R}^{3\times 2}$ be the matrix:*

$$A = \begin{pmatrix} 1 & 2 \\ 0 & 3 \\ 4 & 0 \end{pmatrix}.$$

Compute the singular value decomposition (SVD) of A by finding orthogonal matrices U and V, and a diagonal matrix D such that $A = UDV^\top$. (Also try to do it in R/Python.)

Exercise 1.4.5. *Prove (1.1) by showing that the i-th entry of both expressions for $A\mathbf{x}$ is the same. Specifically, verify that the equality holds for each coordinate $i = 1, \ldots, n$.*

Exercise 1.4.6. *Let $A \in \mathbb{R}^{n\times m}$ and $\mathbf{x} \in \mathbb{R}^m$. Suppose A represents a linear transformation, and $\mathbf{x}$ is a unit vector ($\|\mathbf{x}\| = 1$). Explain the geometric meaning of the vector $A\mathbf{x}$.*

a) What can you say about $A\mathbf{x}$ if A is orthogonal? (here $m = n$)

b) What if A is a diagonal matrix with positive entries?

Exercise 1.4.7. *Show that eigenvalues defined in (1.3) can be equivalently characterized as the roots of the characteristic polynomial $\det(A - \lambda I_m)$.*

Exercise 1.4.8. *Prove (1.2) by expanding both sides of the equation and verifying that the (i, k)-th entries match for each i and k.*

Exercise 1.4.9. *Let $A \in \mathbb{S}^2$ be the matrix*

$$A = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}.$$

Compute the eigenvalues and eigenvectors of A by solving the characteristic equation and verify the spectral theorem by expressing A as $A = U\Lambda U^\top$, where U is an orthogonal matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

Exercise 1.4.10. *Let $A \in \mathbb{R}^{n\times m}$. Express the singular values of $AA^\top$ and $A^\top A$ in terms of the singular values of A.*

Exercise 1.4.11. *Let $A \in \mathbb{S}^m$ be a symmetric matrix with eigenvalues $\lambda_1, \ldots, \lambda_m$.*

a) Show that the largest eigenvalue of A can be characterized as $\lambda_{max} = \max_{\mathbf{x}:\|\mathbf{x}\|=1} \mathbf{x}^\top A\mathbf{x}$.

b) Using this, prove that $\lambda_{max} \geq \frac{1}{m} tr(A)$.

Exercise 1.4.12. *Prove that the covariance matrix Σ is always positive semi-definite. Specifically, show that for any non-zero vector $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{u}^\top \Sigma \mathbf{u} \geq 0$. Explain why this result implies that all eigenvalues of a covariance matrix are non-negative.*

Exercise 1.4.13. *Suppose you have the following data points in $\mathbb{R}^2$: $(1,2), (2,4), (3,6)$. Compute the sample mean vector and the sample covariance matrix for this dataset.*

Exercise 1.4.14. *Consider the matrix*

$$A \in \mathbb{R}^{5\times 5} : A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{pmatrix}.$$

Use the singular value decomposition (SVD) to find a rank-2 approximation of A. Express this approximation as a sum of two rank-one matrices.

Exercise 1.4.15. *Show that $A \in \mathbb{S}^m$ is positive definite if and only if all its eigenvalues are strictly positive. Hint: You can use Theorem 1.1.6.*

Exercise 1.4.16. *Let $A \in \mathbb{S}^m_+$ be a symmetric positive definite matrix with eigenvalues $\lambda_1, \ldots, \lambda_m$.*

a) Prove that adding a small perturbation ϵI (where I is the identity matrix and $\epsilon > 0$) to A preserves positive definiteness.

b) How do the eigenvalues of $A + \epsilon I$ relate to those of A ?

Exercise 1.4.17. *Show that if the eigenvalues of a symmetric matrix $A \in \mathbb{S}^m$ are $\lambda_1, \dots, \lambda_m$, then for any $k \in \mathbb{N}$, the eigenvalues of A^k are $\lambda_1^k, \dots, \lambda_m^k$. If A is invertible (i.e., all $\lambda_i \neq 0$), the same result holds for all integer powers $k \in \mathbb{Z}$.*

Exercise 1.4.18. *Let $\mathbf{X} \in \mathbb{R}^{n \times m}$. Show that the matrix $\mathbf{X}^\top \mathbf{X}$ has rank less than or equal to n. Conclude that it cannot be PD if $n < m$.*

Exercise 1.4.19. *Show that the matrix $H = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$ is symmetric, idempotent ($H^2 = H$), and projects any vector in $\mathbb{R}^n$ onto the space orthogonal to the vector $\mathbf{1}_n$.*

Exercise 1.4.20. *The mean squared error (MSE) of the sample mean is defined as $\mathrm{MSE} = \mathbb{E}\|\bar{x}_n - \mu\|^2$. Show that if we allow the underlying dimension m grow together with n then the MSE may not be negligible.*

Exercise 1.4.21. *Prove that the correlation matrix R satisfies $R_{ij} \in [-1, 1]$ for all i, j. Hint: One idea is to use the fact that R is positive semidefinite.*

Exercise 1.4.22. *Show that if $Y = aX + b$ then $\mathrm{corr}(X, Y) = \pm 1$.*

2 Multivariate Normal Distribution

Multivariate normal distribution is the most important distribution in multivariate statistics. Here we give a relatively streamlined treatment. If you are interested in more details, see Chapter 4 in [1].

2.1 Definition and basic properties

The multivariate normal distribution is fundamental in multivariate statistics due to several key reasons:

1. It is highly tractable: closed under linear transformations, conditioning, and marginalization.

2. It provides a good approximation for sample distributions via the Central Limit Theorem.

3. It frequently arises naturally in the modelling of physical phenomena. See, for example, the work of Gauß in the context of measurement errors, the work of Einstein on Brownian motion, Galton board, and many other examples.

As a result, the multivariate Gaussian distribution appears in many generative models, such as (Bayesian) linear regression, Gaussian Mixture Models, Probabilistic Principal Component Analysis (PPCA), Factor Analysis, Linear Dynamical Systems, Gaussian Processes, Gaussian Conditional Random Fields, autoencoders, etc.

2.1.1 Recall Univariate Gaussian

Recall that $X \sim N(\mu, \sigma^2)$ has the density

$$f_X(x) \;=\; \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}.$$

If $X \sim N(\mu, \sigma^2)$ then $\mathbb{E}X = \mu$ and $\operatorname{var}(X) = \sigma^2$.

If $X_j \sim N(\mu_j, \sigma_j^2)$ for $j = 1, \ldots, m$ are independent, then the vector $X = (X_1, \ldots, X_m)$ has the density[2] for $\mathbf{x} = (x_1, \ldots, x_m)$

[1] Kanti V Mardia, John T Kent, and Charles C Taylor. *Multivariate analysis*, volume 88. John Wiley & Sons, 2024

[2] Recall that if X has independent entries then its density is the product of densities of the individual entries.

$$f_X(\mathbf{x}) \;=\; \prod_{j=1}^m \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{1}{2}\frac{(x_j-\mu_j)^2}{\sigma_j^2}} \;=\; \frac{1}{(2\pi)^{m/2}}(\det(\Sigma))^{-1/2} e^{-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)},$$

where $\Sigma = \operatorname{diag}(\sigma_1^2, \dots, \sigma_m^2)$. Here we used the fact that the determinant of a diagonal matrix is simply the products of its diagonal entries.

2.1.2 *Definition of Multivariate Normal*

The vector $X = (X_1, \dots, X_m)$ is said to have a multivariate normal distribution with mean vector $\mu \in \mathbb{R}^m$ and covariance matrix $\Sigma \in \mathbb{S}_+^m$ if it has the density

$$f_X(\mathbf{x}) \;=\; \frac{1}{(2\pi)^{m/2}}(\det(\Sigma))^{-1/2} e^{-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)}. \tag{2.1}$$

We denote this distribution by $N_m(\mu, \Sigma)$. The random vector $Z \sim N_m(\mathbf{0}, I_m)$ is called standard normal and its density if

$$\phi(z) \;=\; \frac{1}{(2\pi)^{m/2}} e^{-\frac{1}{2}\|z\|^2}.$$

Note that $\|z\|^2 = z^\top z$.

It is extremely useful to think about any multivariate normal distribution as an affine transformation of the standard normal distribution.

Proposition 2.1.1. *If $Z \sim N_m(\mathbf{0}, I_m)$ then $X = \mu + \Sigma^{1/2} Z \sim N_m(\mu, \Sigma)$.*

Proof. Recall the formula for the change of variables: If Z is a random vector with density p_z and $X = G(Z)$, with G bijective and differentiable, then the density of X is

$$p_x(\mathbf{x}) \;=\; p_z(G^{-1}(\mathbf{x}))|\det \nabla G^{-1}|.$$

We apply this in our situation, where $G(z) = \mu + \Sigma^{1/2} z$ and $G^{-1}(\mathbf{x}) = \Sigma^{-1/2}(\mathbf{x} - \mu)$. In this case $\nabla G^{-1} = \Sigma^{-1/2}$ and so

$$p_x(\mathbf{x}) \;=\; \phi(\Sigma^{-1/2}(\mathbf{x} - \mu))(\det \Sigma)^{-1/2},$$

which gives exactly the formula in (2.1). □

← Exercise 2.7.2

It is easy to check that if $X \sim N_m(\mu, \Sigma)$ then

$$\mathbb{E}X = \mu \qquad \text{and} \qquad \operatorname{var}(X) = \Sigma.$$

This could be derived by computing the corresponding integrals, e.g. $\mathbb{E}X = \int_{\mathbb{R}^m} \mathbf{x} f_X(\mathbf{x}) d\mathbf{x}$. However, working with Gaussian distributions we should get accustomed to more algebraic proofs. For example, show first that if $Z \sim N_m(\mathbf{0}, I_m)$ then $\mathbb{E}Z = \mathbf{0}$ and $\operatorname{var}(Z) = I_m$, which can be directly inferred from the univariate result and by independence of the components of Z. To conclude the general result, we can then use Lemma 1.2.2, Lemma 1.2.3, and Proposition 2.1.1.

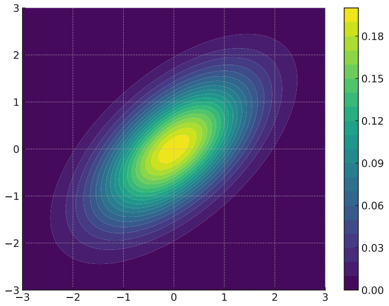


Figure 2.1: Contour plot of the two-dimensional density of the Gaussian distribution with mean $(0, 0)$ and correlation $\rho = 0.6$.

2.1.3 Geometric Interpretation

The contours of the density f_X in (2.1) are ellipsoids (c.f. Figure 2.1.2) defined by the set of points $\mathbf{x}$ such that

$$(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu) = c^2.$$

This is an equation for an ellipsoid[3]. To see this explicitly, consider the spectral decomposition $\Sigma = U\Lambda U^\top$ so that $\Sigma^{-1} = U\Lambda^{-1}U^\top$. Define a new variable $\mathbf{y} = U^\top(\mathbf{x}-\mu)$ (an affine change of coordinates by translation and rotation), and then:

$$(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu) \;=\; \mathbf{y}^\top \Lambda^{-1}\mathbf{y} \;=\; \sum_{j=1}^{m} \frac{y_j^2}{\lambda_j}.$$

Hence, the contours are ellipses aligned with the eigenvectors of Σ, with axis lengths proportional to the square roots of the eigenvalues of Σ.

2.1.4 Mahalanobis Distance

Fix a covariance matrix Σ. An important concept in multivariate normal theory is the **Mahalanobis distance**

If $\Sigma = I_m$, we recover the Euclidean distance.

$$\|\mathbf{x}-\mathbf{y}\|_\Sigma = \sqrt{(\mathbf{x}-\mathbf{y})^\top \Sigma^{-1}(\mathbf{x}-\mathbf{y})}.$$

This distance takes into account the covariance structure of the data.

Recall that χ^2_m has stochastic representation $Z_1^2 + \ldots + Z_m^2$, where Z_i are i.i.d. $N(0,1)$.

Proposition 2.1.2. *If $X \sim N_m(\mu,\Sigma)$, then $(X-\mu)^\top\Sigma^{-1}(X-\mu) \sim \chi^2_m$.*

We leave the proof as an exercise.

← Exercise 2.7.13

2.1.5 Characteristic Function

If X is a random variable then the characteristic function of X is a function defined by $\phi_X(s) = \mathbb{E}e^{isX}$, where i is the imaginary unit ($i^2 = -1$). For a random vector $X = (X_1, \ldots, X_m)$ in $\mathbb{R}^m$ we define the characteristic function as

$$\phi_X(\mathbf{t}) \;=\; \mathbb{E}e^{i\mathbf{t}^\top X}, \qquad \text{where} \;\; \mathbf{t} = (t_1, \ldots, t_m).$$

One of the important properties of characteristic functions is that they define the underlying probability distribution uniquely.

Recall that the characteristic function of $Z \sim N(0,1)$ is:

$$\phi_Z(s) \;:=\; \mathbb{E}[e^{isZ}] = e^{-s^2/2}.$$

From this, we can derive the characteristic function for $X = \mu + \sigma Z \sim N(\mu, \sigma^2)$:

$$\phi_X(s) = \mathbb{E}[e^{isX}] = e^{is\mu - \sigma^2 s^2/2}. \tag{2.2}$$

[3] Recall that an ellipsoid in $\mathbb{R}^m$ is a quadratic surface which, for fixed $\lambda_1, \ldots, \lambda_m, c$, is defined by the equation

$$\frac{y_1^2}{\lambda_1} + \cdots + \frac{y_m^2}{\lambda_m} \;=\; c^2.$$

For the multivariate case, let $Z = (Z_1, \ldots, Z_m) \sim N_m(\mathbf{0}, I_m)$. Then:

$$\phi_Z(\mathbf{t}) \;=\; \mathbb{E}[e^{i\mathbf{t}^\top Z}] \;=\; \prod_{j=1}^m \mathbb{E}e^{it_jZ_j} \;=\; \prod_{j=1}^m e^{-\frac{1}{2}t_j^2} \;=\; e^{-\frac{1}{2}\|\mathbf{t}\|^2}.$$

Thus, for $X = \mu + \Sigma^{1/2}Z$, we get:

$$\phi_X(\mathbf{t}) \;=\; \mathbb{E}e^{i\mathbf{t}^\top(\mu+\Sigma^{1/2}Z)} \;=\; e^{i\mathbf{t}^\top\mu}\mathbb{E}e^{i(\Sigma^{1/2}\mathbf{t})^\top Z} \;=\; e^{i\mathbf{t}^\top\mu}\phi_Z(\Sigma^{1/2}\mathbf{t}) \;=\; e^{i\mathbf{t}^\top\mu-\frac{1}{2}\mathbf{t}^\top\Sigma\mathbf{t}}, \tag{2.3}$$

which gives the formula for the characteristic function of the multivariate normal distribution for general μ and Σ.

2.1.6 *Alternative Characterization of the Gaussian distribution*

The characteristic function provides an alternative characterization of the multivariate normal distribution. Specifically, we say that $X \sim N_m(\mu, \Sigma)$ if for every $\mathbf{u} \in \mathbb{R}^m$, the linear combination $\mathbf{u}^\top X \sim N(\mathbf{u}^\top\mu, \mathbf{u}^\top\Sigma\mathbf{u})$.

To prove that this defines the same distribution, let $\phi_X(\mathbf{t})$ denote the characteristic function of X. Since $\mathbf{u}^\top X \sim N(\mathbf{u}^\top\mu, \mathbf{u}^\top\Sigma\mathbf{u})$, by (2.2), for any $\mathbf{u} \in \mathbb{R}^m$, we have:

$$\phi_{\mathbf{u}^\top X}(s) \;=\; e^{is(\mathbf{u}^\top\mu)-\frac{1}{2}(\mathbf{u}^\top\Sigma\mathbf{u})s^2}.$$

On the other hand,

$$\phi_{\mathbf{u}^\top X}(s) = \mathbb{E}[e^{is\mathbf{u}^\top X}] = \phi_X(s\mathbf{u}).$$

Comparing these two expressions, we conclude that $\phi_X(\mathbf{t}) = e^{i\mathbf{t}^\top\mu-\frac{1}{2}\mathbf{t}^\top\Sigma\mathbf{t}}$ as in (2.3), which proves the claim.

2.2 *Linear Transformation and Marginal Distribution*

As we said earlier, if $Z \sim N_m(\mathbf{0}, I_m)$ then $\mu + \Sigma^{1/2}Z \sim N_m(\mu, \Sigma)$. In this section, we generalize this result. For any matrix $A \in \mathbb{R}^{p\times m}$ and vector $b \in \mathbb{R}^p$, if $X \sim N_m(\mu, \Sigma)$, then the linear transformation $AX + b$ is distributed as:

$$AX + b \sim N_p(A\mu + b, A\Sigma A^\top). \tag{2.4}$$

This again follows easily by using the characteristic function. Let $Y = AX + b$ then

$$\phi_Y(\mathbf{s}) \;=\; \mathbb{E}e^{i\mathbf{s}^\top(AX+b)} \;=\; e^{i\mathbf{s}^\top b}\phi_X(A^\top\mathbf{s}) \;=\; e^{i\mathbf{s}^\top(A\mu+b)-\frac{1}{2}\mathbf{s}^\top A\Sigma A^\top\mathbf{s}},$$

which is the characteristic function of the Gaussian distribution with mean $A\mu + b$ and covariance $A\Sigma A^\top$.

Consider now a subvector $X' = (X_1, \dots, X_p)$ of the vector $X = (X_1, \dots, X_p, \dots, X_m)$ with $p < m$. We have $X' = AX$ where $A = [I_p | \mathbf{0}_{p\times(m-p)}] \in \mathbb{R}^{p\times m}$, where $\mathbf{0}_{p\times(m-p)}$ denotes the matrix of zeros with p rows and $m-p$ columns. Using (2.4), we get that X' has Gaussian distribution with mean $\mu' = A\mu$ and covariance $\Sigma' = A\Sigma A^\top$. Note that $\mu' = (\mu_1, \dots, \mu_p)$ and Σ' is the submatrix of Σ corresponding to rows/columns in $\{1, \dots, p\}$.

More generally, for any subset $I \subseteq \{1, \dots, p\}$ denote by X_I the subvector of X with coordinates indexed by I. Similarly, by $\Sigma_{I,I}$ denote the submatrix of Σ with rows/columns in I. We get the following result.

Proposition 2.2.1. *Suppose $X \sim N_m(\mu, \Sigma)$ and let $I \subseteq \{1, \dots, p\}$. Then the distribution of X_I is Gaussian with mean μ_I and covariance $\Sigma_{I,I}$.*

← Exercise 2.7.15

We now discuss an important result regarding independence. Recall that if $X_i \perp\!\!\!\perp X_j$ then $\operatorname{cov}(X_i, X_j) = 0$. The reverse conclusion is typically not true but it is true for multivariate Gaussian distributions. Indeed, suppose $X \sim N_m(\mu, \Sigma)$ and suppose that $\Sigma_{ij} = 0$ for some i, j. By Proposition 2.2.1, the marginal distribution of (X_i, X_j) is Gaussian with mean (μ_i, μ_j) and covariance

$$\Sigma_{I,I} = \begin{bmatrix} \Sigma_{ii} & 0 \\ 0 & \Sigma_{jj} \end{bmatrix}.$$

Since $\Sigma_{I,I}$ is diagonal, so is its inverse, and so the joint density of (X_i, X_j) factorizes, showing that $X_i \perp\!\!\!\perp X_j$.

More generally, we have the following result:

Lemma 2.2.2. *Suppose $X \sim N_m(\mu, \Sigma)$. Then $AX \perp\!\!\!\perp BX$ if and only if $A\Sigma B^\top = 0$.*

The proof is left as an exercise.

← Exercise 2.7.16

2.3 Conditional Distribution and Conditional independence

In the previous section we saw that Gaussian distributions are closed under taking margins. In this section we show that they are also closed under conditioning. We then discuss conditional independence in Gaussian distributions.

2.3.1 Conditional distribution

Suppose $X \sim N_m(\mu, \Sigma)$. Split $X = (X_A, X_B)$ into a p-dimensional subvector X_A and a $(m-p)$-dimensional subvector X_B. This induces corresponding splits in the mean vector $\mu = (\mu_A, \mu_B)$ and the covariance matrix:

$$\Sigma = \begin{bmatrix} \Sigma_{A,A} & \Sigma_{A,B} \\ \Sigma_{B,A} & \Sigma_{B,B} \end{bmatrix}.$$

Theorem 2.3.1. *The marginal distribution of X_A is Gaussian with mean μ_A and covariance $\Sigma_{A,A}$. The conditional distribution of $X_B|X_A = x_A$ is Gaussian with mean:*

$$\mathbb{E}(X_B \mid X_A = x_A) \;=\; \mu_B + \Sigma_{B,A}\Sigma_{A,A}^{-1}(x_A - \mu_A),$$

and covariance:

$$\mathrm{var}(X_B|X_A = x_A) = \Sigma_{B,B} - \Sigma_{B,A}\Sigma_{A,A}^{-1}\Sigma_{A,B}.$$

Proof. The first part was proved earlier. For the second part, define the new variable $X_{B\cdot A} := X_B - \Sigma_{B,A}\Sigma_{A,A}^{-1}X_A$. Using basic properties of Gaussian distributions, it can be shown that

$$X_{B\cdot A} \sim N_{m-p}(\mu_B - \Sigma_{B,A}\Sigma_{A,A}^{-1}\mu_A, \Sigma_{B,B} - \Sigma_{B,A}\Sigma_{A,A}^{-1}\Sigma_{A,B}).$$

Note that

$$\mathrm{cov}(X_{B\cdot A}, X_A) \;=\; \mathrm{cov}(X_B, X_A) - \Sigma_{B,A}\Sigma_{A,A}^{-1}\Sigma_{A,A} \;=\; \Sigma_{B,A} - \Sigma_{B,A} \;=\; 0$$

and so $X_{B\cdot A} \perp\!\!\!\perp X_A$. Thus, we can decompose X_B as a sum of two independent terms

$$X_B = X_{B\cdot A} + \Sigma_{21}\Sigma_{11}^{-1}X_A.$$

In particular, given $X_A = x_A$, the conditional distribution of X_B is Gaussian with mean and covariance as stated in the theorem. □

2.3.2 *Conditional Independence*

Let $A = \{i, j\}$ and consider splitting the vector X into X_A and X_B, where $B = V \setminus A$. The conditional covariance matrix of X_A given X_B is:

$$\Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}.$$

The variables X_i and X_j are conditionally independent given the variables in X_B if and only if the conditional covariance matrix has a zero off-diagonal entry. Equivalently, we have the condition:

$$\Sigma_{ij} = \Sigma_{iB}\Sigma_{BB}^{-1}\Sigma_{Bj}.$$

This leads to several useful equivalent conditions:

Proposition 2.3.2. *Suppose $X \sim N(0, \Sigma)$. Fix $i, j \in V$ and let $B = V \setminus \{i, j\}$. The following conditions are equivalent:*

(i) $X_i \perp\!\!\!\perp X_j | X_B$

(ii) $\Sigma_{ij} = \Sigma_{iB}\Sigma_{BB}^{-1}\Sigma_{Bj}$

(iii) $K_{ij} = 0$*, where* $K = \Sigma^{-1}$ *is the precision matrix.*

(iv) The regression coefficient $\beta_{i \cdot B\cup\{j\}} = \Sigma_{i,B\cup\{j\}}\Sigma^{-1}_{B\cup\{j\},B\cup\{j\}}$ *has a zero entry in the j-th index.*

We are not going to prove this in class but the proof is relatively elementary with a bit of annoying bookkeeping. We provide it for your convenience.

Proof. The equivalence of (i) and (ii) is clear from the discussion above. For the other implications we rely on standard formulas for block matrices. To show equivalence with (iii) we use the formula for the inverse of a block matrix that we develop independently in Section 2.3.3. From the formula it follows that

$$K_{A,A} \;=\; (\Sigma_{A,A} - \Sigma_{A,B}\Sigma_{B,B}^{-1}\Sigma_{B,A})^{-1}. \qquad (2.5)$$

We have (ii) if and only if $\Sigma_{A,A} - \Sigma_{A,B}\Sigma_{B,B}^{-1}\Sigma_{B,A}$ is diagonal. By (2.5), this is equivalent to $K_{A,A}$ being diagonal, which is equivalent to (iii). For the equivalence between (iii) and (iv) use Proposition 2.3.3 again to conclude that

$$K_{i,B\cup\{j\}} \;=\; -K_{ii}\Sigma_{i,B\cup\{j\}}(\Sigma_{B\cup\{j\},B\cup\{j\}})^{-1}.$$

Since $K_{ii} > 0$, then $K_{ij} = 0$ if and only if the j-th entry of $\Sigma_{i,B\cup\{j\}}\Sigma^{-1}_{B\cup\{j\},B\cup\{j\}}$ is zero. □

The relationship between conditional independence and zeros in the precision matrix K is fundamental in **Gaussian graphical models**, which we will cover later in this course; see Chapter 7.

2.3.3 *Partitioned Inverse**

Let $X \sim N(0,\Sigma)$ and consider the partition of X into two subvectors X_A and X_B. Set

$$K \;=\; \Sigma^{-1} \;=\; \begin{bmatrix} K_{A,A} & K_{A,B} \\ K_{B,A} & K_{B,B} \end{bmatrix},$$

where K is the precision matrix.

Proposition 2.3.3. *We have the following formulas for the blocks of the precision matrix in terms of the blocks of* Σ*:*

$$\begin{aligned} K_{A,A} &= (\Sigma_{A,A} - \Sigma_{A,B}\Sigma_{B,B}^{-1}\Sigma_{B,A})^{-1}, \\ K_{A,B} &= -K_{A,A}\Sigma_{A,B}\Sigma_{B,B}^{-1}, \\ K_{B,B} &= \Sigma_{B,B}^{-1} + \Sigma_{B,B}^{-1}\Sigma_{B,A}K_{A,A}\Sigma_{A,B}\Sigma_{B,B}^{-1} \end{aligned}$$

Proof. We verify that these formulas satisfy $\Sigma K = I_m$. For instance, consider multiplying the first block row of Σ by the first block column of K:

$$\Sigma_{A,A}K_{A,A} + \Sigma_{A,B}K_{B,A} = \Sigma_{A,A}K_{A,A} - \Sigma_{A,B}\Sigma_{B,B}^{-1}\Sigma_{B,A}K_{A,A} = (\Sigma_{A,A} - \Sigma_{A,B}\Sigma_{B,B}^{-1}\Sigma_{B,A})K_{A,A} = I.$$

Similarly, multiplying the first block row of Σ by the second block column of K gives:

$$\begin{aligned}\Sigma_{A,A}K_{A,B} + \Sigma_{A,B}K_{B,B} &= -\Sigma_{A,A}K_{A,A}\Sigma_{A,B}\Sigma_{B,B}^{-1} + \Sigma_{A,B}\Sigma_{B,B}^{-1} + \Sigma_{A,B}\Sigma_{B,B}^{-1}\Sigma_{B,A}K_{A,A}\Sigma_{A,B}\Sigma_{B,B}^{-1} \\ &= -(\Sigma_{A,A} - \Sigma_{A,B}\Sigma_{B,B}^{-1}\Sigma_{B,A})K_{A,A}\Sigma_{A,B}\Sigma_{B,B}^{-1} + \Sigma_{A,B}\Sigma_{B,B}^{-1} \\ &= 0.\end{aligned}$$

The remaining cases can be checked similarly. □

This partitioned inverse result is useful in various scenarios, such as Gaussian graphical models and Bayesian networks. We already saw an application in the previous subsection.

2.4 *Estimation of Parameters*

Suppose we have data $\mathbf{X} \in \mathbb{R}^{n\times m}$, where the rows $\mathbf{x}_i \in \mathbb{R}^m$ are i.i.d. samples from $N_m(\mu, \Sigma)$. A natural way to estimate μ and Σ is through the Maximum Likelihood Estimator (MLE). In this section, we show that the MLEs $\widehat{\mu}$ and $\widehat{\Sigma}$ satisfy:

$$\widehat{\mu} \;=\; \bar{\mathbf{x}}_n \;=\; \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i,$$

and

$$\widehat{\Sigma} \;=\; S_n \;=\; \frac{1}{n}\sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top.$$

2.4.1 *The Likelihood Function*

Let $f(\mathbf{x})$ denote the density of $N_m(\mu, \Sigma)$ as given in (2.1). After taking logs, we get

$$\log f(\mathbf{x}) = -\tfrac{p}{2}\log(2\pi) + \tfrac{1}{2}\log\det(\Sigma^{-1}) - \tfrac{1}{2}(\mathbf{x}-\mu)^\top\Sigma^{-1}(\mathbf{x}-\mu).$$

The log-likelihood of the data matrix $\mathbf{X}$ is:

$$\ell(\mu, \Sigma) = \sum_{i=1}^n \log f(\mathbf{x}_i) = \text{const} + \frac{n}{2}\log\det(\Sigma^{-1}) - \frac{1}{2}\sum_{i=1}^n (\mathbf{x}_i - \mu)^\top\Sigma^{-1}(\mathbf{x}_i - \mu).$$

2.4.2 *Optimize over μ*

Using basic calculus, the derivative of the log-likelihood with respect to μ is:[4]

$$\nabla_\mu \ell(\mu, \Sigma) = \Sigma^{-1}\left(\sum_{i=1}^n \mathbf{x}_i - n\mu\right).$$

Setting this equal to zero gives the MLE for μ:

$$\widehat{\mu} \;=\; \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i \;=\; \bar{\mathbf{x}}_n.$$

We get this result irrespective of what Σ is as the only thing we used is that Σ is invertible[5].

2.4.3 *Optimize over Σ*

Knowing that $\hat{\mu} \;=\; \bar{\mathbf{x}}_n$, we now optimize $\ell(\bar{\mathbf{x}}, \Sigma)$ over Σ. For this, we first rewrite $\ell(\bar{\mathbf{x}}, \Sigma)$ in a suitable form. It is convenient to list the following useful observation, which it is as a positive definite matrix.
Useful observation: We know that the trace satisfies $\operatorname{tr}(AB) = \operatorname{tr}(BA)$ whenever both multiplications are well defined. In particular, for any two $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ we have

$$\mathbf{x}^\top \mathbf{y} \;=\; \operatorname{tr}(\mathbf{x}^\top \mathbf{y}) \;=\; \operatorname{tr}(\mathbf{y}\mathbf{x}^\top),$$

where $\mathbf{y}\mathbf{x}^\top \in \mathbb{R}^{m\times m}$.

This can be checked easily directly as $\operatorname{tr}(\mathbf{x}\mathbf{y}^\top) = \sum_i (\mathbf{x}\mathbf{y}^\top)_{ii} = \sum_i x_i y_i = \mathbf{x}^\top \mathbf{y}$.

Using this, we get

Note that the trace is a linear function: $\operatorname{tr}(aA + bB) = a\operatorname{tr}(A) + b\operatorname{tr}(B)$.

$$\begin{aligned}
\ell(\bar{\mathbf{x}}, \Sigma) &= \text{const} + \tfrac{n}{2}\log\det(\Sigma^{-1}) - \tfrac{1}{2}\sum_{i=1}^n \operatorname{tr}((\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top \Sigma^{-1}) \\
&= \text{const} + \tfrac{n}{2}\log\det(\Sigma^{-1}) - \tfrac{n}{2}\operatorname{tr}(\tfrac{1}{n}\sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top \Sigma^{-1}) \\
&= \text{const} + \tfrac{n}{2}\log\det(\Sigma^{-1}) - \tfrac{n}{2}\operatorname{tr}(S_n\Sigma^{-1}),
\end{aligned}$$

where $S = \frac{1}{n}\sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$ is the sample covariance matrix.

To optimize $\ell(\widehat{\mu}, \Sigma)$ over Σ, we rewrite the log-likelihood in terms of $K = \Sigma^{-1}$:

$$\ell(\widehat{\mu}, K) = \text{const} + \frac{n}{2}\log\det(K) - \frac{n}{2}\operatorname{tr}(S_n K). \tag{2.6}$$

We are not going to show this formally here but this function is a concave function of $K = \Sigma^{-1}$. In particular, every local maximum is a global maximum.
Useful observation: Another useful observation is that for any two $A, B \in \mathbb{S}^m$

$$\operatorname{tr}(AB) \;=\; \sum_{i,j=1}^m A_{ij}B_{ij}.$$

[4] Verify that $\nabla_\theta(b^\top\theta) \;=\; b$ and $\nabla_\theta \theta^\top K\theta = 2K\theta$.
← Exercise 2.7.19

[5] Recall that $\ker(A) = \{\mathbf{x} : A\mathbf{x} = \mathbf{0}\}$ and A is invertible if and only if $\ker(A) = \{\mathbf{0}\}$.

In particular, this defines an inner product on $\mathbb{S}^m$, which is called the trace inner product[6].

Using this, we easily check that $\nabla_K \operatorname{tr}(S_n K) = S_n$. It can be also shown that $\nabla_K \log\det(K) = K^{-1} = \Sigma$; for a formal derivation check p. 641 here. Thus, taking the derivative of $\ell(\bar{\mathbf{x}}, K)$ with respect to K and setting it equal to zero gives the MLE for Σ: $\widehat{\Sigma} = S_n$ if S_n is positive definite. In the case when $n < m$, the matrix S_n cannot be positive definite and so the MLE does not exist.

2.4.4 Independence of $\bar{\mathbf{x}}$ and S

If $\mathbf{x}_1, \dots, \mathbf{x}_n$ is a sample from $N(\mu, \Sigma)$ then $\bar{\mathbf{x}}_n \sim N(\mu, \frac{1}{n}\Sigma)$ and the sample covariance matrix S_n is another random quantity of interest.

Proposition 2.4.1. *In the Gaussian sample, the sample mean* $\bar{\mathbf{x}} = \frac{1}{n}\mathbf{X}^\top \mathbf{1}_n$ *and the sample covariance* $S = \frac{1}{n}(\mathbf{X} - \mathbf{1}_n\bar{\mathbf{x}}^\top)^\top(\mathbf{X} - \mathbf{1}_n\bar{\mathbf{x}}^\top)$ *are independent.*

Proof. Recall that $S = \frac{1}{n}\mathbf{X}^\top H\mathbf{X} = \frac{1}{n}(H\mathbf{X})^\top(H\mathbf{X})$. Thus, it is enough to show that $\bar{x}_n$ is independent of $H\mathbf{X}$. Since both $\bar{x}_n$ and $H\mathbf{X}$ have Gaussian distributions, independence can be checked by computing the covariance. Since the k-th row of $H\mathbf{X}$ is $\mathbf{x}_k - \bar{\mathbf{x}}$ we check

Recall $H = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top \in \bar{\mathbb{S}}_+^m$ and $H^2 = H$

$$\operatorname{cov}(\bar{\mathbf{x}}, \mathbf{x}_k - \bar{\mathbf{x}}) \;=\; \operatorname{cov}(\bar{\mathbf{x}}, \mathbf{x}_k) - \operatorname{var}(\bar{\mathbf{x}}) \;=\; \frac{1}{n}\operatorname{var}(\mathbf{x}_k) - \frac{1}{n}\Sigma \;=\; 0.$$

□

2.5 Hotelling's T^2 Distribution*

Hotelling's T^2 distribution is a multivariate extension of the univariate Student's t-distribution. It is particularly useful in testing hypotheses about the mean in a single population or equality of means in two multivariate populations.

2.5.1 Testing equality of means in two univariate populations

Recall that if you have a pair of samples x_i, $i = 1, \dots, n$ from $N(\mu_x, \sigma^2)$ and y_i, $i = 1, \dots, m$ from $N(\mu_y, \sigma^2)$ then we can test $H_0 : \mu_x = \mu_y$ using the two-sample t-test. To see this, define

$$S \;=\; \frac{1}{n+m-2}\Big[\sum_{i=1}^n (x_i - \bar{x})^2 + \sum_{j=1}^m (y_j - \bar{y})^2\Big] \;=\; \frac{n-1}{n+m-2}S_1 + \frac{m-1}{n+m-2}S_2$$

to be the pooled sample variance estimator of σ^2. Define the test statistic

$$t \;=\; \frac{\bar{x} - \bar{y}}{\sqrt{S(\frac{1}{n} + \frac{1}{m})}}. \tag{2.7}$$

To analyze its distribution, we first formulate the following lemma.

[6] In the literature it is common to denote $\langle A, B\rangle := \operatorname{tr}(AB)$. More generally, if $A, B \in \mathbb{R}^{n\times m}$ then $\operatorname{tr}(AB^\top)$ also defines an inner product.

Lemma 2.5.1. *If $x_1, \dots, x_n$ is a random sample from $N(\mu, \sigma^2)$ then $\frac{1}{\sigma^2}\sum_{i=1}^n (x_i - \bar{x})^2 \sim \chi^2_{n-1}$.*

Proof. Recall that $H = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n$ is a symmetric PSD matrix satisfying $H^2 = H$. Let $\mathbf{x} = (x_1, \dots, x_n)$ and recall that $H\mathbf{x}$ has entries $x_i - \bar{x}$. It follows that $\sum_{i=1}^n (x_i - \bar{x})^2 = \mathbf{x}^\top H\mathbf{x}$. Let $H = U\Lambda U^\top$. Since H is a projection matrix, $\Lambda = \Lambda^2$ and so the diagonal entries of Λ are either 0 or 1. Since $\operatorname{rank}(H) = n-1$ it follows that $\Lambda_{11} = 0$ and $\Lambda_{ii} = 1$ for $i = 2, \dots, n-1$. Define $\mathbf{z} = U^\top\mathbf{x}$. Since $\mathbf{x} \sim N_n(\mu\mathbf{1}, \sigma^2 I_n)$, it follows that $\mathbf{z} \sim N_n(\mu U^\top\mathbf{1}, \sigma^2 I_n)$. Moreover, $U^\top\mathbf{1} = \sqrt{n}\mathbf{e}_1$, where $\mathbf{e}_1 = (1, 0, \dots, 0)$. Denote $\mathbf{z}_{n-1} = (z_2, \dots, z_n)$ obtained from $\mathbf{z}$ by dropping the first coordinate. Then, in particular,

As many proofs in multivariate statistics, this one combines some basic results in linear algebra with properties of the Gaussian distribution.

$$\frac{1}{\sigma}\mathbf{z}_{n-1} \;\sim\; N_{n-1}(\mathbf{0}, I_{n-1}).$$

We have

$$\frac{1}{\sigma^2}\mathbf{x}^\top H\mathbf{x} \;=\; \frac{1}{\sigma^2}\mathbf{z}^\top\Lambda\mathbf{z} \;=\; \frac{1}{\sigma^2}\mathbf{z}_{n-1}^\top\mathbf{z}_{n-1}$$

and we already noted that $\mathbf{z}_{n-1}^\top\mathbf{z}_{n-1} \sim \chi^2_{n-1}$. □

Proposition 2.5.2. *Under H_0, the statistic t, defined in (2.7), has t-Student distribution with $n+m-2$ degrees of freedom.*

Proof. Recall that t-student distribution with r degrees of freedom has stochastic representation

$$t_r \;=\; \frac{U}{\sqrt{V/r}} \quad \text{with } U \sim N(0,1), \quad V \sim \chi^2_r, \quad U \perp\!\!\!\perp V.$$

$U \perp\!\!\!\perp V$ denotes independence of U, V

Recall that $\bar{x} \sim N(\mu_x, \sigma^2/n)$, $\bar{y} \sim N(\mu_y, \sigma^2/m)$ and so, by independence, under H_0

$$\bar{x} - \bar{y} \;\sim\; N(0, \sigma^2(\tfrac{1}{n} + \tfrac{1}{m})) \quad \text{and so} \quad U := \frac{\bar{x} - \bar{y}}{\sigma\sqrt{\frac{1}{n} + \frac{1}{m}}} \;\sim\; N(0,1).$$

Define $V = \frac{n+m-2}{\sigma^2}S$ and note that

$$t \;=\; \frac{\frac{\bar{x}-\bar{y}}{\sigma\sqrt{\frac{1}{m}+\frac{1}{n}}}}{\sqrt{\frac{1}{\sigma^2}S}} \;=\; \frac{U}{\sqrt{\frac{V}{m+n-2}}}.$$

We have

$$V \;=\; \frac{n+m-2}{\sigma^2}S \;=\; \frac{n-1}{\sigma^2}S_1 + \frac{m-1}{\sigma^2}S_2.$$

By Lemma 2.5.1, $\frac{n-1}{\sigma^2}S_1 \sim \chi^2_{n-1}$ and $\frac{m-1}{\sigma^2}S_2 \sim \chi^2_{m-1}$. Moreover, both these random variables are independent. It follows that $V \sim \chi^2_{m+n-2}$, which concludes the proof. □

2.5.2 *Hotelling distribution*

The Hotelling distribution describes the behavior of Hotelling's T^2 -statistic, a key tool in multivariate hypothesis testing. It generalizes the Student's t-distribution to the multivariate case and is commonly used to test hypotheses about the mean vector of a multivariate normal population.

Although we do not derive the Hotelling's distributions from the first principles here, we link it to another standard distribution called the **F-distribution**. Recall that if $X_1 \sim \chi^2_{d_1}$ and $X_2 \sim \chi^2_{d_2}$ are two independent chi-squared random variables, then the random variable: $F_{d_1,d_2} = \frac{(X_1/d_1)}{(X_2/d_2)}$ follows an F -distribution with d_1 and d_2 degrees of freedom.

The Hotelling's $T^2(d_1, d_2)$ is a simple rescaling of the F-distribution.

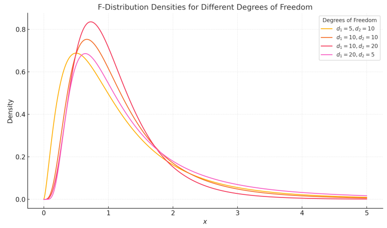


Figure 2.2: The density of F_{d_1,d_2} for different combinations of d_1, d_2.

Definition 2.5.3. *The Hotelling's distribution $T^2(d_1, d_2)$ satisfies*

$$T^2(d_1, d_2) \;=\; \frac{d_1 d_2}{d_2 - d_1 + 1} F_{d_1, d_2 - d_1 + 1}.$$

Given a sample $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from $N_p(\mu, \Sigma)$ with $n \geq p$ (so that S_n is invertible) define the following statistics:[7]

$$D^2 \;=\; (n-1)(\bar{\mathbf{x}}_n - \mu)^\top \mathbf{S}_n^{-1}(\bar{\mathbf{x}}_n - \mu),$$

Remark 2.5.4. *Note that D^2 is invariant under nonsingular linear transformations $\mathbf{x} \mapsto A\mathbf{x} + \boldsymbol{b}$.*

Theorem 2.5.5. *We have $D^2 \sim T^2(p, n-1)$.*

We omit the proof. If you are interested in details, check Corollary 4.5.6 in [8].

2.5.3 *Multivariate Test for Equality of Means*

Consider now a sample $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from $N_p(\mu_x, \Sigma)$ and a sample $\mathbf{y}_1, \ldots, \mathbf{y}_m$ from $N_p(\mu_y, \Sigma)$. We can extend the ideas from the univariate case to define a multivariate test. Let

$$S = \frac{1}{n+m-2}\left[nS_x + mS_y\right],$$

where S_x and S_y are the sample covariance matrices for the two samples[9]. Define the test statistic:

$$D^2 = (\bar{\mathbf{x}} - \bar{\mathbf{y}})^\top S^{-1}(\bar{\mathbf{x}} - \bar{\mathbf{y}}).$$

Theorem 2.5.6. *Under $H_0 : \mu_x = \mu_y$, we have*

$$\tfrac{nm}{n+m} D^2 \sim T^2(p, n+m-2).$$

[7] Recall that $(\bar{x}_n - \mu) \perp\!\!\!\perp S_n$.

[8] Kanti V Mardia, John T Kent, and Charles C Taylor. *Multivariate analysis*, volume 88. John Wiley & Sons, 2024

[9] Note that S is an unbiased estimate of Σ.

For details, if you are interested, see Theorem 4.6.1 in [10].

In practice, we can implement this test in `R` using the function `HotellingsT2(X, Y)` in the package `ICSNP`. Note that the assumption of equal covariance matrices for the two populations can sometimes be problematic, but when the assumption approximately holds, this multivariate test is often more powerful than performing a series of univariate tests.

2.6 *Gaussian Processes in Multivariate Statistics*

Gaussian Processes (GPs) are a powerful tool in statistics and machine learning for modeling continuous, multivariate data. A Gaussian Process is a generalization of the Gaussian distribution over a finite set of points to a distribution over functions. In this section we give only basic definitions and some practical examples. We omit many important computational aspects. Our goal is simply to illustrate that very basic results on the multivariate normal distribution, lead to powerful techniques.

2.6.1 *Definition of a Gaussian Process*

A Gaussian Process on the set T can be defined as follows:

Definition 2.6.1. *A* ***Gaussian Process*** *is a collection of random variables $\mathbb{X}_T := (X_t)_{t \in T}$ such that for any collection of points $t_1, \ldots, t_n$ in T, the vector $(X_{t_1}, \ldots, X_{t_n})$ has a multivariate normal distribution.*

Effectively a Gaussian process $\mathbb{X}_T$ is defined by specifying the **mean function** $m : T \to \mathbb{R}$ such that $m(t) = \mathbb{E}X_t$ and the kernel function $k : T \times T \to \mathbb{R}$ such that $k(t, t') = \operatorname{cov}(X_t, X_{t'})$. Although T can be pretty much arbitrary, we will assume that it is a metric space. Often $T = \mathbb{R}^m$.

Note that the kernel function must be symmetric ($k(t, t') = k(t', t)$) and it satisfies that for all $n \in \mathbb{N}$ and all $t_1, \ldots, t_n \in T$ the matrix $\Sigma \in \mathbb{S}^n$ defined by $\Sigma_{ij} = k(t_i, t_j)$ is positive definite. Such functions are very well studied in mathematics and their theory is very rich.

The choice of the kernel function $k(t, t')$ plays a crucial role in determining the properties of functions sampled from the GP, such as smoothness and periodicity. Popular choices include the squared exponential kernel and the Matérn kernel. If $T \subseteq \mathbb{R}^m$ then the exponential kernel is defined as

$$k_{\mathrm{E}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\tfrac{1}{\ell}\|\mathbf{x} - \mathbf{x}'\|\right),$$

The exponential kernel generates functions that are not differentiable. This makes it suitable for modeling rough or jagged processes.

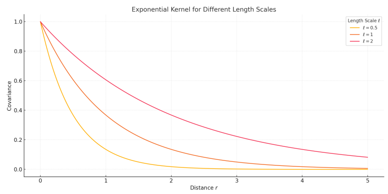


Figure 2.3: The exponential kernel as a function of $r = \|\mathbf{x} - \mathbf{x}'\|$ for different values of ℓ and $\sigma = 1$.

[10] Kanti V Mardia, John T Kent, and Charles C Taylor. *Multivariate analysis*, volume 88. John Wiley & Sons, 2024

Matérn kernel is defined as

$$k_{\mathrm{M}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)^{\nu} K_{\nu} \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right),$$

where $\nu > 0$ controls the smoothness of the kernel, $\ell > 0$ is the length scale, controlling the distance over which points are correlated, $\sigma^2 > 0$ is the signal variance, K_ν is the modified Bessel function of the second kind of order ν, and $\Gamma(\cdot)$ is the gamma function.

Note that in both cases the correlation drops as the distance between the points $\mathbf{x}, \mathbf{x}'$ increases.

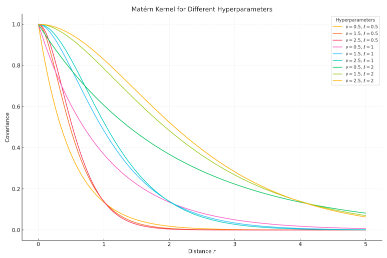


Figure 2.4: The Matérn kernel as a function of $r = \|\mathbf{x} - \mathbf{x}'\|$ for different choices of the hyperparameters.

2.6.2 *Example: Gaussian Processes for Spatial Data in R*

Gaussian Processes are commonly applied in spatial statistics to model observations that are correlated across geographic locations. In this example, we use Gaussian Processes to model temperature data observed at various locations, capturing spatial dependencies across the region.

Suppose we observe temperatures across a grid of spatial locations. We can model the temperature at any given location as a sample from a Gaussian Process with a spatial covariance structure, such as the exponential or Matérn kernel, which are popular choices for spatial modeling.

```
# Load required packages
library(geoR)     # For spatial data simulation
library(fields)   # For spatial GP modeling and visualization

# Set up a grid of locations
set.seed(42)
n <- 100
locs <- expand.grid(x = seq(0, 1, length.out = n), y = seq(0, 1,
↪  length.out = n))

# Generate spatially correlated temperature data
true_cov <- exp(-rdist(locs) / 2)  # Exponential covariance function
temp <- t(chol(true_cov)) %*% rnorm(n * n)

# Plot the spatial data
image(matrix(temp, n, n), main = "Simulated Temperature Data",
      xlab = "Longitude", ylab = "Latitude")
```

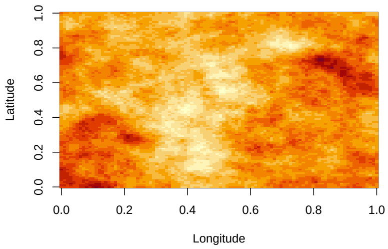


Figure 2.5: Simulated Gaussian Process on a 2D grid with the exponential kernel.

In this code:

- We set up a grid of $100^2 = 10{,}000$ locations over a 1×1 area.
- We generate spatially correlated temperature data with an exponential covariance function.

In simpler terms, the grid defined points $\mathbf{x}_{ij} \in [0,1]^2$ for $i, j = 1, \ldots, 100$. Now a big covariance matrix can be constructed whose entries correspond to the covariance between the sites $\mathbf{x}_{ij}$ and $\mathbf{x}_{kl}$. Now we generate a single distribution from this big 10000-dimensional Gaussian distribution. The resulting simulated temperature distribution is given in Figure 2.5.

Remark 2.6.2. *Handling a large Gaussian distribution as described above can be computationally costly, so in practice, the sample is generated entry by entry by leveraging the properties of Gaussian distributions in a smart way.*

Next, we use Gaussian Processes to make predictions about the temperature at unobserved locations by fitting a GP model to our simulated data.

On a high level this is quite straightforward. Add to the earlier grid any additional points you would like to include in the predictions. Using the kernel function, compute the joint distribution of the training and test data. Then, using the standard formulas, compute the conditional distribution of the test data given the training data (this is all Gaussian!). The conditional mean gives you predictions and conditional covariance helps you quantify uncertainty.

In practice, you would also like to learn hyperparameters of the kernel function. This could be done using the maximum likelihood approach. For computational reasons often working with Gaussian Processes, we rely on Bayesian statistics. There are some standard packages to make this more convenient. Consider for example the following code.

```
# Fit a Gaussian Process model
library(spBayes)
fit <- spLM(temp ~ 1, coords = as.matrix(locs), starting = list("phi" = 1,
↪  "sigma.sq" = 1), tuning = list("phi" = 0.1), priors = list("phi.Unif"
↪  = c(0.01, 1)))

# Predict at new locations
new_locs <- expand.grid(x = seq(0, 10, length.out = 50), y = seq(0, 10,
↪  length.out = 50))
pred <- spPredict(fit, pred.coords = as.matrix(new_locs))

# Visualize the predictions
image(matrix(pred$p.y.predictive, 50, 50), main =
↪  "Predicted Temperature Data", xlab = "Longitude", ylab = "Latitude")
```

In this step:

- We fit a Gaussian Process model to the spatial temperature data using the `spLM` function from the `spBayes` package.
- We make predictions at a set of new locations and visualize the GP-based temperature predictions.

This example illustrates the flexibility of Gaussian Processes for spatial modeling. By choosing a spatially appropriate kernel, we can capture the dependencies across locations and make predictions with associated uncertainty. This is especially valuable in multivariate statistics, where the correlations among measurements across space can be crucial for accurate modeling and inference.

2.6.3 *Example: Nonparametric regression using GPs**

In machine learning, Gaussian Processes are often used for non-parametric regression. This is outside of scope of this class so we provide this example only to complete the picture but it can be safely omitted.

Consider the following code. Here, unlike in the previous example, we code all relevant formulas from scratch.

```
# Load required libraries
library(MASS)    # For multivariate normal sampling
library(ggplot2) # For fancy plotting

# Define a squared exponential (RBF) kernel
rbf_kernel <- function(x, y, length_scale = 1, sigma_f = 1) {
  sigma_f^2 * exp(-0.5 * (outer(x, y, "-")^2) / length_scale^2)}

# Generate training data from noisy sine function
set.seed(42)
n_train <- 10
x_train <- seq(0, 10, length.out = n_train)
y_train <- sin(x_train) + rnorm(n_train, sd = sqrt(0.1))

# Define test points where predictions are needed
x_test <- seq(0, 10, length.out = 100)

# Compute covariance matrices
length_scale <- 1  # Length scale of the RBF kernel
sigma_f <- 1       # Signal variance
sigma_n <- sqrt(0.1)     # Noise variance

K <- rbf_kernel(x_train, x_train, length_scale, sigma_f) + diag(sigma_n^2,
↪  n_train)
K_s <- rbf_kernel(x_train, x_test, length_scale, sigma_f)
K_ss <- rbf_kernel(x_test, x_test, length_scale, sigma_f)

# Predictive mean and covariance
K_inv <- solve(K)
mu_s <- t(K_s) %*% K_inv %*% y_train  # Transpose K_s for predictive mean
cov_s <- K_ss - t(K_s) %*% K_inv %*% K_s  # Correct covariance calculation

# Sample from the posterior predictive distribution
n_samples <- 5
y_samples <- mvrnorm(n_samples, mu_s, cov_s)

# Create a data frame for the test points and GP predictions
gp_data <- data.frame(x = x_test,GP_Mean = as.vector(mu_s),True_Function =
↪  sin(x_test))

# Add the GP samples to the data frame
```

```
gp_samples_df <- data.frame(x = rep(x_test, times = n_samples), Sample =
 ↪  factor(rep(1:n_samples, each = length(x_test))),Value =
 ↪  as.vector(t(y_samples)))

# Create a data frame for the training points
train_data <- data.frame(x = x_train, y = y_train)

# Plot using ggplot2
ggplot() +
  # True function
  geom_line(data = gp_data, aes(x = x, y = True_Function),
            color = "black", linetype = "dashed", size = 1, alpha = 0.7) +
  # GP mean
  geom_line(data = gp_data, aes(x = x, y = GP_Mean),
            color = "lightpink2", size = 1.2) +
  # GP samples
  geom_line(data = gp_samples_df, aes(x = x, y = Value, group = Sample),
            color = "springgreen3", alpha = 0.5) +
  # Training points
  geom_point(data = train_data, aes(x = x, y = y),
             color = "blue", size = 2) +
  # Labels and theme
  labs(title = "Gaussian Process Regression",
       x = "x", y = "f(x)") +
  theme_minimal(base_size = 14) +
  theme(legend.position = "none")
```

This code first defines a grid of $n = 10$ equally spaced points $x_1, \ldots, x_n$ in the interval $[0, 10]$. For each of these points, we generate an observation $y_i \sim N(f(x_i), \sigma^2)$ with $\sigma^2 = 0.1$. These are the blue dots in Figure 2.6. These are our noisy observations from the function $f(x) = \sin(x)$ (the dashed line).

Having the training data, the test data and a fixed kernel function (here the Gaussian kernel) we can compute the covariance matrix in the underlying Gaussian process (over all the point). Then we can sample from the conditional distribution[11] of the test data given the training data. Doing it multiple times gives us the greenish curves in Figure 2.6. We can also compute the conditional mean and also plot it. This is the pink curve.

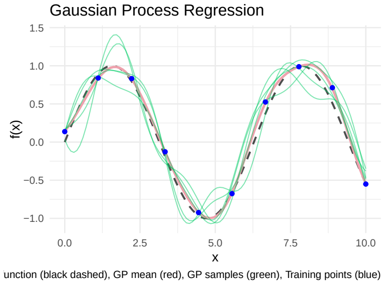


Figure 2.6: Illustration for GPs applied for non-parametric regression.

2.7 *Exercises*

Exercise 2.7.1. *Given $X \sim N_m(\mu, \Sigma)$, show that $X - \mu$ has the same distribution as $\Sigma^{1/2} Z$, where $Z \sim N_m(0, I_m)$.*

Exercise 2.7.2. *Suppose we want to generate a sample of n observations from $N_m(0, \Sigma)$. Using R or Python, write a code that does it by sampling independently a bunch of univariate $N(0, 1)$ variables and transforming them appropriETLY.*

Exercise 2.7.3. *Consider two random vectors $X \sim N_m(\mu_X, \Sigma_X)$ and $Y \sim N_q(\mu_Y, \Sigma_Y)$. Show that if X and Y are independent, the joint distribution*

[11] All the relevant formulas already appeared earlier!

of (X, Y) is multivariate normal with mean (μ_X, μ_Y) and block diagonal covariance matrix.

Exercise 2.7.4. *Let $X_1, X_2 \sim N_m(\mu, \Sigma)$ and define $Y = X_1 + X_2$. Find the distribution of Y.*

Exercise 2.7.5. *Derive the formula for the covariance matrix of the residuals when performing a linear regression on a multivariate normal sample.*

Exercise 2.7.6. *Let $X \sim N_m(\mu, \Sigma)$. Use the spectral decomposition of Σ to transform X into independent standard normal variables.*

Exercise 2.7.7. *Let $X \sim N_m(\mu, \Sigma)$ and suppose that Σ is singular (i.e., Σ is not invertible). Prove that in this case, X lies on a subspace of $\mathbb{R}^m$. What is the dimension of this subspace?*

Exercise 2.7.8. *Let $X \sim N_m(\mu, \sigma^2 I_m)$ and let U be any orthogonal matrix. Show that the distribution X is the same as the distribution of UX.*

Exercise 2.7.9. *Let $X_1, X_2 \sim N_m(\mu, \Sigma)$ be independent. Consider the random vector $Z = X_1 + X_2$. Prove that the covariance matrix of Z is 2Σ.*

Exercise 2.7.10. *Suppose $X \sim N_m(\mu, \Sigma)$. Show that the linear regression coefficients of X_1 on $X_2, \ldots, X_m$ can be expressed in terms of the blocks of the covariance matrix Σ. What is the relationship between these coefficients and conditional variances?*

Exercise 2.7.11. *Suppose $X \sim N_m(\mu, \Sigma)$. Show that the correlation matrix $\Sigma_{corr} = D^{-1/2}\Sigma D^{-1/2}$, where D is the diagonal matrix of variances, is also positive semidefinite. What does this tell us about the structure of correlation matrices?*

Exercise 2.7.12. *Prove that the determinant of the covariance matrix Σ for a multivariate normal distribution $N_m(\mu, \Sigma)$ is related to the volume of the ellipsoid corresponding to the 1-standard deviation contour of the distribution. Derive the explicit formula for this volume.*

Exercise 2.7.13. *Prove Proposition 2.1.2.* Hint*: Let $Z = \Sigma^{-1/2}(X - \mu)$. Show that $Z \sim N_m(\mathbf{0}, I_m)$ and use the fact that the sum of the squares of independent standard normal variables has a chi-squared distribution.*

Exercise 2.7.14. *Let $X \sim N_m(\mu, \Sigma)$ and $Y \sim N_q(\nu, \Lambda)$ be independent random vectors. Show that the joint distribution of (X, Y) is $N_{p+q}((\mu, \nu), \Sigma \oplus \Lambda)$, where $\oplus$ denotes the block diagonal operation.*

Exercise 2.7.15. *Complete all the details of the proof of Proposition 2.2.1.*

Exercise 2.7.16. *Prove Lemma 2.2.2.*

Exercise 2.7.17. *Suppose $X_1, X_2 \sim N_m(0, \Sigma)$ are independent and $Y = X_1 - X_2$. Derive the distribution of Y. How does the covariance structure of Y compare to that of X_1 and X_2 individually?*

Exercise 2.7.18. *Let $X \sim N_m(\mu, \Sigma)$. Show that for any $a \in \mathbb{R}^m$, the probability $\mathbb{P}(a^\top X > c)$ depends on both $a^\top \mu$ and $a^\top \Sigma a$. Derive a formula for this probability.*

Exercise 2.7.19. *Consider the functions $f, g : \mathbb{R}^n \to \mathbb{R}$ given by $f(\mathbf{x}) = \boldsymbol{a}^\top \mathbf{x}$ and $g(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x}$ for $\boldsymbol{a} \in \mathbb{R}^n$ and $A \in \mathbb{S}^n$. Show that $\nabla f(\mathbf{x}) = \boldsymbol{a}$ and $\nabla g(\mathbf{x}) = 2A\mathbf{x}$.*

3
Non-normal distributions

Multivariate Gaussian is obviously not the only distribution widely considered to model multivariate continuous data. In this chapter, we provide a very brief treatment of some of the key ideas involving distributions beyond the multivariate normal.

3.1 Elliptical distributions

3.1.1 Spherical distributions

Recall that $O(m)$ denotes the group[1,2] of $m \times m$ orthogonal matrices, that is, $U \in \mathbb{R}^{m\times m}$ such that $U^\top U = I_m$. A random vector $X \in \mathbb{R}^m$ is said to have a *spherical distribution* if for any $U \in O(m)$, the random vector X is equal in distribution to UX, i.e.,

$$X \overset{d}{=} UX \quad \forall U \in O(m).$$

One example of a spherical distribution is the standard normal distribution $Z \sim N_m(0, I_m)$. More generally, if $Z \sim N_m(0, I_m)$ and $\tau > 0$ is a positive random variable independent of Z, then[3]

$$X = \frac{1}{\sqrt{\tau}} Z$$

has a spherical distribution. To see this, let $U \in O(m)$. Then

$$UX \;=\; \frac{1}{\sqrt{\tau}} UZ \;\overset{d}{=}\; \frac{1}{\sqrt{\tau}} Z \;=\; X,$$

since $UZ \overset{d}{=} Z$ and $UZ \perp\!\!\!\perp \tau$. Therefore, X has the same distribution as UX, satisfying the definition of spherical symmetry.

Moment structure: The spherical symmetry implies that the mean of X must be zero and the covariance matrix must be proportional to the identity matrix, i.e.,

← Exercise 3.4.1

$$\mathbb{E}(X) = 0, \quad \mathrm{var}(X) = cI_m$$

for some constant $c \geq 0$. In the case of $X = \frac{1}{\sqrt{\tau}} Z$ with $Z \sim N(0, I_m)$, we have $\mathrm{var}(X) = \mathbb{E}[\tau^{-1}] I_m$.

← Exercise 3.4.2

[1] The set $O(m)$ is a group because $U, V \in O(m)$ then $UV^{-1} \in O(m)$. Check it!

[2] Note that if $U \in O(m)$ then $|\det(U)| = 1$.

[3] This is sometimes called the common variance model. Note that the entries of Z are independent. Although the entries of X will be uncorrelated (can you show it?), they will not be independent.

3.1.2 Independence of $\|X\|$ and $X/\|X\|$

An important property of spherical distributions is the *independence* of the norm of X, denoted $\|X\| := \sqrt{X^\top X}$, and the direction of X, represented by the unit vector $X/\|X\|$. To see this, note that for any $U \in O(m)$,

$$\frac{X}{\|X\|} \overset{d}{=} \frac{UX}{\|UX\|} = \frac{UX}{\|X\|} = U\frac{X}{\|X\|}.$$

Thus, $\frac{X}{\|X\|}$ is invariant under orthogonal transformations, meaning it must have a rotationally symmetric distribution. The only rotationally symmetric distribution on the unit sphere is the uniform distribution. Therefore, $\frac{X}{\|X\|}$ has a uniform distribution on the unit sphere in $\mathbb{R}^m$, independent of the length $\|X\|$.

3.1.3 Formal argument using polar coordinates*

To formally show that $\|X\|$ and $X/\|X\|$ are independent, we use a change of coordinates to *polar coordinates*.

Polar coordinates: In $\mathbb{R}^2$, polar coordinates are given by:

$$x_1 = r\cos(\theta), \quad x_2 = r\sin(\theta), \quad r > 0, \quad 0 \le \theta < 2\pi.$$

This generalizes to $\mathbb{R}^m$ with the transformation:

$$\mathbf{x} = r\mathbf{u}(\boldsymbol{\theta}),$$

where $\boldsymbol{\theta} = (\theta_1, \dots, \theta_{m-1})$ are the angular coordinates, and $r = \|\mathbf{x}\|$ is the radial coordinate. Specifically, for $i = 1, \dots, m$ we have[4]:

$$u_i(\boldsymbol{\theta}) = \cos(\theta_i)\prod_{j=0}^{i-1}\sin(\theta_j), \quad \sin(\theta_0) = 1, \quad \cos(\theta_m) = 1.$$

The angles θ_j are constrained by:

$$0 \le \theta_j \le \pi, \quad j = 1, \dots, m-2, \quad 0 \le \theta_{m-1} < 2\pi.$$

Recall that for a function $f : \mathbb{R}^n \to \mathbb{R}^m$, its Jacobian at $\mathbf{x} \in \mathbb{R}^n$, if exists, is the matrix $\mathrm{J}f(\mathbf{x})$ of the partial derivatives $\mathrm{J}f(\mathbf{x}) = [\frac{\partial f_i}{\partial x_j}] \in \mathbb{R}^{m\times n}$. The determinant of the Jacobian matrix of the polar coordinates transformation $\mathbf{x} = r\boldsymbol{u}(\boldsymbol{\theta})$ is

$$J(r, \boldsymbol{\theta}) = r^{m-1}\prod_{i=2}^{m-1}\sin^{m-i}(\theta_{i-1}).$$

The reason why we care about this determinant is that it appears in the change of variables formula.

[4] Check that $\|\mathbf{x}\| = r$.

Independence of radial and angular components: Suppose X has a density $f(\mathbf{x})$ that only depends on $\|\mathbf{x}\|$, i.e., $f(\mathbf{x}) = g(\|\mathbf{x}\|^2)$ for some function g. When changing variables to polar coordinates, we find:

$$f(\mathbf{x})d\mathbf{x} = g(r^2)r^{m-1}dr \prod_{i=2}^{m-1} \sin^{m-i}(\theta_{i-1})d\boldsymbol{\theta}.$$

Thus, we see that the density separates into a product of a radial component $g(r^2)r^{m-1}dr$ and an angular component, showing that $\|\mathbf{x}\|$ and $\mathbf{x}/\|\mathbf{x}\|$ are independent.

3.1.4 *Elliptical distributions as affine transformations of spherical distributions*

An *elliptical distribution* is a generalization of the multivariate normal distribution. A random vector $X \in \mathbb{R}^m$ is said to have an elliptical distribution, denoted $X \sim E(\mu, \Sigma)$, if X can be written as:

$$X = \mu + \Sigma^{1/2}Z,$$

where Z is a spherical random vector (e.g., standard normal) and $\Sigma^{1/2}$ is the square root of a positive semi-definite matrix Σ. This form includes the multivariate normal as a special case; recall Proposition 2.1.1.

Why do we care? The main reason is that elliptical distributions allow us to model distributions that look similar to multivariate normal but have much heavier tails. This makes it suitable to model extreme events or data with outliers, which are common in many real-world applications such as finance, insurance, and environmental studies. For example, in finance, the heavy tails of elliptical distributions can capture the higher likelihood of extreme losses or gains compared to the Gaussian model. By maintaining properties like symmetry and linear correlation structures, elliptical distributions strike a balance between flexibility and mathematical tractability, making them a valuable tool for robust modelling in high-dimensional settings.

3.1.5 *Scale mixture of normals*

A key example of elliptical distributions is the **scale mixture of normals**, which admits the stochastic representation:

$$X = \mu + \frac{1}{\sqrt{\tau}}\Sigma^{1/2}Z,$$

where $Z \sim N_m(0, I_m)$ and $\tau > 0$ is a random variable independent of Z. This framework captures several important distributions:

- If $\tau \equiv 1$, we recover the multivariate normal distribution.

- If $\tau \sim \frac{1}{k}\chi^2_k$, then X follows a multivariate t-distribution with k degrees of freedom.

- If $k = 1$, we obtain the multivariate Cauchy distribution.

- If $\tau \sim \text{Exp}(1)$, we get the multivariate Laplace distribution.

Note that X may be a complicated distribution but conditionally on τ, X is simply Gaussian. Thus, many efficient approaches to working with scale mixture of normals, use this structure.

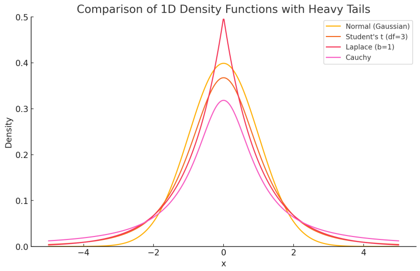


Figure 3.1: Examples of 1-d densities that are generalized by the scale mixture of normals. Note that the tails are generally heavier than for the Gaussian distribution.

3.1.6 *Covariance and correlation matrix*

Note that in elliptical distributions, Σ is called the *scale matrix* and does not directly represent the covariance of X. However, it holds that:

$$\text{Var}(X) = c\Sigma$$

for some constant $c > 0$. Therefore, the correlation structure of X is governed by Σ, even though the exact covariance requires calculating c.

3.2 *Copula models*

A copula is a statistical tool that allows us to model the dependence structure between random variables separately from their marginal distributions. This is especially useful when modeling multivariate data where the marginal distributions are not Gaussian.[5]

Definition 3.2.1. *A copula function (or simply a copula) is a multivariate distribution function with uniform marginals. A copula function in m dimensions is typically written $C(\mathbf{u})$, $\mathbf{u} = (u_1, \ldots, u_m)$.*

3.2.1 *Sklar's theorem*

The foundation of copula theory is *Sklar's Theorem*.

Theorem 3.2.2 (Sklar, 1959)**.** *Given a continuous random vector $X = (X_1, \ldots, X_m)$ with joint c.d.f. F and marginal c.d.f.s $F_1, \ldots, F_m$, there exists a unique copula C such that for all $\mathbf{x} = (x_1, \ldots, x_m)$*

$$F(x_1, \ldots, x_m) \;=\; C(F_1(x_1), \ldots, F_m(x_m)). \tag{3.1}$$

Conversly, for any given one-dimensional c.d.f.s $F_1, \ldots, F_m$ and a copula C, (3.1) defines an m-variate c.d.f. F with marginal c.d.f.s $F_1, \ldots, F_m$.

The copula $C : [0,1]^m \to [0,1]$ captures the dependence between the components of $\mathbf{X}$, while the marginal distributions $F_1, \ldots, F_m$ describe the behavior of each component independently.

[5] Recall that the CDF of the uniform $U(0,1)$ distribution is $F(u) = u$ and so $C : [0,1]^m \to [0,1]$ must satisfy $C(u_1, 1, \ldots, 1) = u_1$.

To understand Sklar's theorem, consider the case when $m = 1$. Here $C(u) = u$ for $u \in [0,1]$ and the CDF of X is indeed $C(F(x)) = F(x)$. Note also that, if X is a univariate continuous random variable with c.d.f. F[6] then $F(X) \sim U(0,1)$. Indeed,

$$\mathbb{P}(F(X) \leq u) \;=\; \mathbb{P}(X \leq F^{-1}(u)) \;=\; F(F^{-1}(u)) \;=\; u.$$

Let now $X = (X_1, \ldots, X_m)$ be a random vector with c.d.f.

$$F(x_1, \ldots, x_m) \;=\; \mathbb{P}(X_1 \leq x_1, \ldots, X_m \leq x_m).$$

Set $U_i = F_i(X_i)$. Then $U = (U_1, \ldots, U_m)$ has marginal uniform distributions with c.d.f.

$$\begin{aligned}\mathbb{P}(U_1 \leq u_1, \ldots, U_m \leq u_m) &= \mathbb{P}(X_1 \leq F_1^{-1}(u_1), \ldots, X_m \leq F_m^{-1}(u_m)) \\ &= F(F_1^{-1}(u_1), \ldots, F_m^{-1}(u_m)) \\ &=: C(\mathbf{u}).\end{aligned} \tag{3.2}$$

Note that C defined here is the same as C in (3.1) and Sklar's theorem states that such C is defined uniquely.

3.2.2 *Invariance under monotone transformations*

Suppose that $Y_i = f_i(X_i)$, where each f_i is a strictly increasing (and so invertible) transformation. It is important to note that X and Y define the same copula. Indeed, let F, G be the CDFs of X and Y respectively with individual CDFs denotes as F_i, G_i. In the light of (3.2), it is enough to show that

$$F(F_1^{-1}(u_1), \ldots, F_m^{-1}(u_m)) \;=\; G(G_1^{-1}(u_1), \ldots, G_m^{-1}(u_m))$$

Now note that

$$G_i(y_i) = \mathbb{P}(Y_i \leq y_i) = \mathbb{P}(X_i \leq f_i^{-1}(y_i)) = F_i(f_i^{-1}(y_i)),$$

which shows that $G_i = F_i \circ f_i^{-1}$, or equivalently, $G_i^{-1} = f_i \circ F_i^{-1}$. Similarly,

$$G(y_1, \ldots, y_m) \;=\; F(f_1^{-1}(y_1), \ldots, f_m^{-1}(y_m))$$

Using these two equations we get that

$$G(G_1^{-1}(u_1), \ldots, G_m^{-1}(u_m)) \;=\; F(f_1^{-1}(G_1^{-1}(u_1)), \ldots, f_m^{-1}(G_m^{-1}(u_m))) \;=\; F(F_1^{-1}(u_1), \ldots, F_m^{-1}(u_m))$$

as claimed.

3.2.3 *Density of a copula*

The p.d.f. $c(\mathbf{u})$ of a copula $C(\mathbf{u})$ is obtained by differentiation. We

Recall that for a continuous distribution with CDF $F(x_1, \ldots, x_m)$, its density $f(x_1, \ldots, x_m)$ is given by

$$f(x_1, \ldots, x_m) \;=\; \frac{\partial^m}{\partial x_1 \cdots \partial x_m} F(x_1, \ldots, x_m).$$

[6] If X is continuous then F is strictly increasing almost surely.

have

$$c(\boldsymbol{u}) \;=\; \frac{\partial^m C(\boldsymbol{u})}{\partial u_1 \cdots \partial u_m}.$$

Using the fact that $C(\boldsymbol{u}) = F(F_1^{-1}(u_1), \dots, F_m^{-1}(u_m))$ and the chain rule, we have

$$c(\mathbf{u}) \;=\; \frac{f(\mathbf{x})}{\prod_{i=1}^m f_i(x_i)}. \tag{3.3}$$

3.2.4 Gaussian copula

The *Gaussian copula* is derived from the multivariate normal distribution. Suppose $\mathbf{X} \sim N_m(\boldsymbol{\mu}, \Sigma)$ is a normal random vector. Since a copula is unchanged if comonents undergo a monotone transformation, w.l.o.g. we may assume $\mu_i = 0$ and $\Sigma_{ii} = 1$ for all i so that the marginal distributions are standard normal. The CDF of $N(0,1)$ is denoted by $\Phi(\mathbf{x})$.

Using formula in (3.3) the corresponding density is ← Exercise 3.4.3

$$c(\boldsymbol{u};\Sigma) \;=\; \det(\Sigma)^{-1/2} \exp\{-\tfrac{1}{2}\boldsymbol{x}^\top(\Sigma^{-1} - I_m)\boldsymbol{x}\}, \quad \text{where } x_i = \Phi^{-1}(u_i). \tag{3.4}$$

The most popular choice for Σ is the equicorrelation matrix $\Sigma = (1-\rho)I_m + \rho\mathbf{1}\mathbf{1}^\top$, where $-\frac{1}{m-1} \le \rho \le 1$. For this choice $\det(\Sigma) = (1-\rho)^{m-1}(1+(m-1)\rho)$. So for example the density of a two-dimensional Gaussian copula is

$$c(\boldsymbol{u};\rho) \;=\; \frac{1}{\sqrt{1-\rho^2}} \exp\left\{-\frac{\rho^2(x_1^2+x_2^2) - 2\rho x_1 x_2}{2(1-\rho^2)}\right\}. \tag{3.5}$$

3.2.5 Archimedean copula*

In addition to Gaussian copulas, there are other families of copulas, such as *Archimedean copulas*, which are popular in applications due to their simplicity. An Archimedean copula is defined using a generator function $\phi : (0,1) \to [0,\infty)$ that is strictly decreasing and convex with $\phi(0^+) = \infty$ and $\phi(1^-) = 0$. Set $\psi = \phi^{-1}$, which is monotone decreasing on $(0,\infty$ with $\psi(0^+) = 1$ and $\phi(\infty^-) = 0$. Assume that ψ is completely monotone function, that is, suppose that ψ is infinitely differentiable, and that the derivatives alternate in sign: $(-1)^k\psi^{(k)}(v) > 0$ for all $0 < v < \infty$, $k \ge 1$.

The m-variate Archimedean copula generated by ϕ is given by:

$$C(u_1, \dots, u_m) = \psi\left(\phi(u_1) + \cdots + \phi(u_m)\right).$$

It can be shown that C defines a valid CDF with PDF

$$c(\boldsymbol{u}) \;=\; \psi^{(m)}(v)\prod_{i=1}^m \phi'(u_i), \quad \text{where } v_i = \phi(u_i),\; v = v_1 + \ldots + v_m.$$

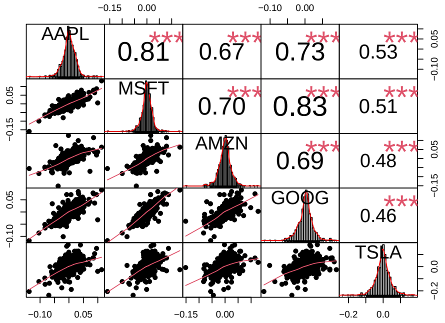
Some well-known Archimedean copulas include:

- **Clayton copula:** $\phi(t) = \frac{1}{\theta}(t^{-\theta} - 1)$ for $\theta > 0$.
- **Gumbel copula:** $\varphi(t) = (-\log t)^{\theta}$ for $\theta \geq 1$.
- **Frank copula:** $\varphi(t) = -\log\left(\frac{e^{-\theta t}-1}{e^{-\theta}-1}\right)$ for $\theta \neq 0$.

3.2.6 *Example*

Just to illustrate how this can be used. We use copulas to model the dependence structure of returns of five random stocks. First plot their correlation structure.

```r
library(quantmod)
library(copula)
library(PerformanceAnalytics)

# Define stock tickers and date range
tickers <- c("AAPL", "MSFT", "AMZN", "GOOG", "TSLA")
start_date <- "2020-01-01"
end_date <- "2023-01-01"

# Download adjusted closing prices
getSymbols(tickers, from = start_date, to = end_date, src = "yahoo")
prices <- do.call(merge, lapply(tickers, function(x) Ad(get(x))))

# Calculate daily log returns
returns <- na.omit(ROC(prices, type = "continuous"))
colnames(returns) <- tickers

# Plot the returns
chart.Correlation(returns, histogram = TRUE, pch = 19)
```



We fit a Gaussian copula to the multivariate log returns. First, convert returns to pseudo-observations by mapping to the unit interval $[0, 1]$, as required by copula models.

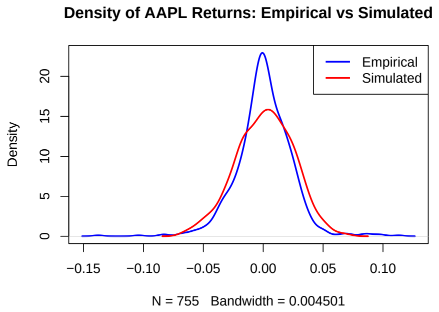
```r
# Convert to pseudo-observations
u <- pobs(as.matrix(returns))

# Fit a Gaussian copula
cop_model <- normalCopula(dim = ncol(returns))  # Gaussian copula
fit <- fitCopula(cop_model, u, method = "ml")  # Maximum likelihood

# Extract fitted parameters (correlation matrix)
rho_matrix <- getSigma(fit@copula)
print("Fitted correlation matrix:")
print(rho_matrix)
```

```
> print(rho_matrix)
      [,1]  [,2]  [,3]  [,4]  [,5]
[1,] 1.000 0.655 0.655 0.655 0.655
[2,] 0.655 1.000 0.655 0.655 0.655
[3,] 0.655 0.655 1.000 0.655 0.655
[4,] 0.655 0.655 0.655 1.000 0.655
[5,] 0.655 0.655 0.655 0.655 1.000
```

The correlation matrix looks very simple but remember that we still have the marginal distributions, which adds additional flexibility. We can now simulate new returns based on the fitted copula model and the original marginal distributions.

```r
# Simulate from the fitted Gaussian copula
simulated_copula <- rCopula(nrow(returns), fit@copula)
```

```
# Transform copula samples to the original marginals
simulated_returns <- data.frame(
  AAPL = qnorm(simulated_copula[,1], mean = mean(returns$AAPL), sd =
   ↪  sd(returns$AAPL)),
  MSFT = qnorm(simulated_copula[,2], mean = mean(returns$MSFT), sd =
   ↪  sd(returns$MSFT)),
  AMZN = qnorm(simulated_copula[,3], mean = mean(returns$AMZN), sd =
   ↪  sd(returns$AMZN)),
  GOOG = qnorm(simulated_copula[,4], mean = mean(returns$GOOG), sd =
   ↪  sd(returns$GOOG)),
  TSLA = qnorm(simulated_copula[,5], mean = mean(returns$TSLA), sd =
   ↪  sd(returns$TSLA)))

# Compare empirical and simulated correlations
empirical_corr <- cor(returns)
simulated_corr <- cor(simulated_returns)

cat("Empirical Correlation Matrix:\n")
print(empirical_corr)
cat("\nSimulated Correlation Matrix:\n")
print(simulated_corr)

# Scatterplot matrix of simulated returns
pairs(simulated_returns, main = "Simulated Returns (Gaussian Copula)", pch
 ↪  = 19, col = rgb(0, 0, 1, 0.5))

# Overlaid density plots for one stock (e.g., AAPL)
plot(density(returns$AAPL), col = "blue", main =
 ↪  "Density of AAPL Returns: Empirical vs Simulated", lwd = 2)
lines(density(simulated_returns$AAPL), col = "red", lwd = 2)
legend("topright", legend = c("Empirical", "Simulated"), col = c("blue",
 ↪  "red"), lwd = 2)
```



3.2.7 *Applications of copula models*

Copulas have a wide range of applications, especially in fields where understanding the dependence between random variables is crucial. Some important applications include:

- **Finance and risk management:** In finance, copulas are used to model the joint behavior of asset returns or risk factors. They are particularly useful for modeling tail dependencies, which refer to the joint likelihood of extreme losses or gains. The Gaussian copula, for instance, was famously used in pricing collateralized debt obligations (CDOs), although the limitations of copulas in capturing extreme co-movements were highlighted during the financial crisis of 2007-2008.

- **Insurance and actuarial science:** In insurance, copulas allow actuaries to model dependencies between different types of claims, such as health claims and life insurance claims. By understanding the joint risk, insurers can set more appropriate premiums and reserves for correlated risks.

- **Environmental sciences:** Copulas are used to model the dependence structure between extreme environmental events, such as floods or droughts. This is important for designing infrastructure that can withstand correlated extreme events. For instance, the joint probability of high rainfall and river overflow can be better understood with copulas, leading to more informed decisions in flood risk management.

- **Multivariate survival analysis:** In medical statistics and survival analysis, copulas are used to model the time until the occurrence of multiple types of events (such as death, disease relapse, or recovery) that may be dependent on each other. Archimedean copulas are particularly popular in this context due to their tractability.

3.3 *Gaussian mixture*

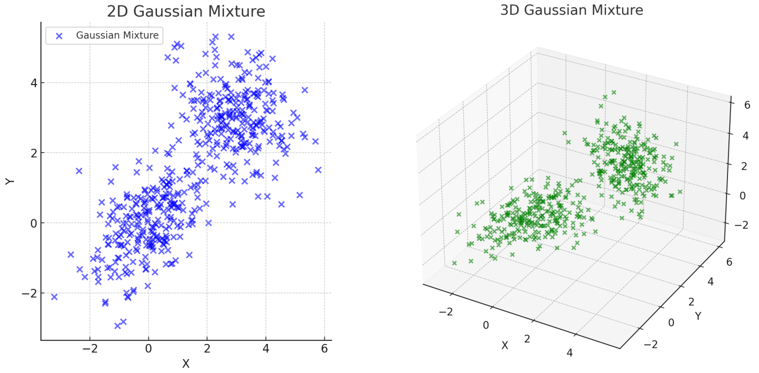
A Gaussian Mixture Model (GMM) is a probabilistic model that assumes the data is generated from a mixture of several Gaussian distributions, each with its own parameters. Formally, a random vector $X \in \mathbb{R}^m$ is said to follow a GMM if its density function is a weighted sum of multivariate Gaussian densities:

← Exercise 3.4.4

$$f(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}_m(\mathbf{x}; \mu_k, \Sigma_k), \tag{3.6}$$

where $\pi_k \geq 0$ are the mixture weights such that $\sum_{k=1}^{K} \pi_k = 1$, and $\mathcal{N}_m(\mathbf{x}; \mu_k, \Sigma_k)$ denotes the multivariate normal density with mean $\mu_k \in \mathbb{R}^m$ and covariance matrix $\Sigma_k \in \mathbb{S}_{+}^{m}$.

Figure 3.3 provides two simple examples of 2D and 3D datasets that are generated from mixtures of two Gaussian components.



Remark 3.3.1. *At this point it is not clear how to handle such a complicated distribution efficiently. For example, given fixed parameters values,*

how can we generate samples? We will see in Section 3.3.2 that this model admits a simple latent variable representation, which can be exploited in various ways.

← Exercise 3.4.5

3.3.1 *Why use Gaussian mixtures?*

Gaussian mixtures offer several advantages:

- **Flexibility:** They can model complex distributions by combining simple Gaussian components. A GMM can approximate almost any continuous probability distribution by increasing the number of components K.
- **Multimodality:** GMMs are particularly useful for modeling data with multiple modes (clusters), where each Gaussian component models one mode.
- **Easily interpretable:** Each Gaussian component has an intuitive interpretation (mean, covariance, weight). It is a natural model if the population comes from different sub-populations.

Gaussian mixtures form a popular model based clustering method. In this context, for simplicity, we often assume $\Sigma_k = \Sigma$ for $k = 1, \ldots, K$.

3.3.2 *Gaussian Mixture as a latent variable model*

Let $Z \in \{1, \ldots, K\}$ be a discrete random variable such that $\mathbb{P}(Z = k) = \pi_k$ for $k = 1, \ldots, K$. Moreover, suppose that the conditional distribution of X given $Z = k$ is $N_m(\mu_k, \Sigma_k)$. Then the marginal distribution of X is the same as given in (3.6) (check!)

Remark 3.3.2. *Many popular probabilistic machine learning methods model data distribution $p(x)$ from a joint $p(x, z)$, which is specified by setting the marginal distribution $p(z)$ of a latent variable and the conditional $p(x|z)$. Models of that form lead to many highly tractable exact and approximate estimation procedures.*

3.3.3 *Likelihood inference for GMMs*

Suppose we have data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^m$ from the Gaussian mixture model. Fitting a GMM to a dataset involves estimating the parameters $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$, which by default is done by maximizing the log-likelihood function

$$\ell(\theta) = \sum_{i=1}^n \log f(\mathbf{x}_i) = \sum_{i=1}^n \log(\sum_{k=1}^K \pi_k N_m(\boldsymbol{x}_i; \mu_k, \Sigma_k)).$$

The likelihood function for GMMs is a complicated multimodal function of the parameters θ and maximizing it directly may be hard. In fact it is not even bounded! However, we can use the latent representation of the GMM to perform the Expectation-Maximization (EM) algorithm.

Remark 3.3.3. *Note that we assume that K is fixed throughout the whole analysis. In practice, K is not always known. In this case, learning K is a separate task, which we do not discuss here in detail.*

3.3.4 *Expectation-Maximization (EM) Algorithm*

Suppose we have data $\mathbf{x}_1, \dots, \mathbf{x}_n$ with distribution $f_X(\mathbf{x}; \theta)$. The goal is to estimate the parameter θ that maximizes the likelihood. In cases where some variables are not directly observed, we can model the observed data as being the marginal distribution of a larger joint distribution. Specifically, we assume that the data X is generated from a latent model, where (X, Z) are jointly distributed, and Z represents unobserved (latent) variables.

The EM algorithm is particularly useful when the marginal likelihood of X is difficult to compute, but the complete data likelihood (i.e., the joint likelihood of X and Z) is easier to work with. Consider the compolete data $(\boldsymbol{x}_1, \boldsymbol{z}_1), \dots, (\boldsymbol{x}_n, \boldsymbol{z}_n)$, where we observe only $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ as before. Assume that the joint model depends on the parameter vector θ.

The algorithm iteratively estimates θ by alternating between two steps:

(0) Initialize θ^0, a reasonable initial guess for the parameter.

(E-step) Compute the expected value of the complete data log-likelihood ℓ_F

$$\ell_F(\theta) \;=\; \sum_{i=1}^{n} \log f(\boldsymbol{x}_i, \boldsymbol{z}_i; \theta)$$

with respect to the posterior distribution of the latent variables Z, given the observed data X and the current parameter estimate θ^0. This gives the function:

$$Q(\theta; \theta_0) = \mathbb{E}_{Z|X,\theta^0}[\ell_F(\theta)|\mathbf{X}, \theta^0] \;=\; \sum_{i=1}^{n} \mathbb{E}_{Z|X,\theta^0}[\log f(\boldsymbol{x}_i, Z; \theta)|X = \boldsymbol{x}_i, \theta^0].$$

(M-step) Maximize the expected complete log-likelihood with respect to θ:

$$\theta^1 = \arg\max_{\theta} Q(\theta; \theta_0).$$

Update $\theta^0 \leftarrow \theta^1$ and repeat until convergence.

Under mild conditions, it can be shown that each iteration of the EM algorithm increases the observed data log-likelihood. Specifically, after each E-step and M-step, the likelihood satisfies the inequality:

$$\ell(\theta^{t+1}) \geq \ell(\theta^t).$$

This property ensures that the algorithm makes progress toward maximizing the likelihood at every step.

While the EM algorithm is guaranteed to increase the likelihood at each step, it may not necessarily converge to the global maximum. Instead, the algorithm converges to a **local maximum** of the likelihood function. Convergence can be understood through the following results:

1. Monotonicity: The likelihood function increases monotonically at each step of the algorithm. This is guaranteed by the fact that the E-step constructs a lower bound on the likelihood that is maximized in the M-step.

2. Local Convergence: Under mild regularity conditions (e.g., the likelihood is smooth), the sequence of parameter estimates θ^t generated by the EM algorithm converges to a stationary point of the likelihood function. However, this stationary point might be a local maximum, a saddle point, or (rarely) a local minimum.

3. Global Convergence: The EM algorithm does not guarantee global convergence, meaning that different initializations may lead to different local optima. For this reason, it is common to try several random initializations to increase the chances of finding a global maximum.

3.3.5 *EM Algorithm for Gaussian Mixture Models (GMM)*

The complete data log-likelihood for GMMs is given by:

$$\ell_F(\theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}(\mathbf{z}_i = k) \left[\log \mathcal{N}(\mathbf{x}_i; \mu_k, \Sigma_k) + \log \pi_k \right].$$

Here, $\mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k)$ is the Gaussian probability density function for component k with mean μ_k and covariance matrix Σ_k.

E-step

In the E-step, we compute the expected value of the complete data log-likelihood, given the current parameter estimates $\theta^0 = \{\pi_k^0, \mu_k^0, \Sigma_k^0\}$. Specifically,

$$\begin{aligned} Q(\theta; \theta_0) &= \sum_{i=1}^{n} \sum_{k=1}^{k} \mathbb{E}_{Z|X,\theta_0}[\mathbb{1}(Z = k)|X = \boldsymbol{x}_i, \theta_0] \left[\log \mathcal{N}(\mathbf{x}_i; \mu_k, \Sigma_k) + \log \pi_k \right] \\ &= \sum_{i=1}^{n} \sum_{k=1}^{k} \mathbb{P}(z_i = k|X = \boldsymbol{x}_i, \theta_0) \left[\log \mathcal{N}(\mathbf{x}_i; \mu_k, \Sigma_k) + \log \pi_k \right] \end{aligned}$$

we compute the posterior probabilities, or the responsibilities, that each point $\mathbf{x}_i$ belongs to component k:

$$w_{ik} = \mathbb{P}(\mathbf{z}_i = k|\mathbf{x}_i, \theta^0) = \frac{\pi_k^0 \mathcal{N}(\mathbf{x}_i; \mu_k^0, \Sigma_k^0)}{\sum_{l=1}^K \pi_l^0 \mathcal{N}(\mathbf{x}_i; \mu_l^0, \Sigma_l^0)}.$$

Using these responsibilities, the expected complete log-likelihood is:

$$Q(\theta; \theta_0) = \sum_{i=1}^n \sum_{k=1}^K w_{ik} \left[\log \mathcal{N}(\mathbf{x}_i; \mu_k, \Sigma_k) + \log \pi_k\right].$$

M-step

In the M-step, we maximize $Q(\theta; \theta_0)$ with respect to the parameters $\theta = \{\pi_k, \mu_k, \Sigma_k\}$. This leads to the following updates:

$$\begin{aligned} \pi_k &= \frac{1}{n} \sum_{i=1}^n w_{ik}, \\ \mu_k &= \frac{\sum_{i=1}^n w_{ik} \mathbf{x}_i}{\sum_{i=1}^n w_{ik}}, \\ \Sigma_k &= \frac{\sum_{i=1}^n w_{ik} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^\top}{\sum_{i=1}^n w_{ik}}. \end{aligned}$$

Deriving the updates for π_k and μ_k is relatively straightforward. The update on Σ_k requires either some persistence or some experience with calculus of matrix-functions. We live this as an exercise.

These updates have intuitive interpretations: $\hat{\mu}_k$ is the weighted average of the data points assigned to component k, where the weights are the posterior probabilities w_{ik}; $\hat{\Sigma}_k$ is the weighted covariance matrix of the points assigned to component k; $\hat{\pi}_k$ is the proportion of points assigned to component k.

The EM algorithm iterates between the E-step and M-step until convergence, at which point the log-likelihood no longer increases significantly.

3.3.6 *Example: EM Algorithm for Gaussian Mixture Models (GMM)*

Problem Setup Suppose we have a dataset where each observation is drawn from a mixture of two Gaussian distributions. The goal is to estimate the parameters (means, covariances, and mixing proportions) of these Gaussian components using the Expectation-Maximization (EM) algorithm.

Assume the data consists of n points in 2D, where each observation $\mathbf{x}_i$ comes from one of two Gaussian distributions. Let π_k represent the mixing proportion of the k-th Gaussian component, μ_k the mean vector, and Σ_k the covariance matrix for component k, with $k = 1, 2$. The dataset is a mixture of these distributions with some overlap.

Data Simulation We can generate a 2D dataset from two Gaussian distributions, one with

$$\mu_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \qquad \Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

and the other with

$$\mu_1 = \begin{bmatrix} 7 \\ 7 \end{bmatrix}, \qquad \Sigma_1 = \begin{bmatrix} 1 & -0.3 \\ -0.3 & 1 \end{bmatrix}$$

The mixing parameter is $\pi_1 = 0.4 = 1 - \pi_2$. We use the following R code:

```
set.seed(123)
n <- 300 # number of points
pi_true <- c(0.4, 0.6) # true mixing proportions
mu_true <- list(c(2, 2), c(7, 7)) # true means
sigma_true <- list(matrix(c(1, 0.5, 0.5, 1), 2), matrix(c(1, -0.3, -0.3,
↪  1), 2)) # covariances

# Sample labels from a multinomial distribution
z <- rbinom(n, size = 1, prob = pi_true[2])
x <- matrix(0, nrow = n, ncol = 2)

# Generate points from the two Gaussians
for (i in 1:n) {
  if (z[i] == 0) {
    x[i, ] <- MASS::mvrnorm(1, mu_true[[1]], sigma_true[[1]])
  } else {x[i, ] <- MASS::mvrnorm(1, mu_true[[2]], sigma_true[[2]])}
}

# Plotting the data points with color corresponding to the group
plot(x, col = z + 1, pch = 19, main =
↪  "Generated Data from a Gaussian Mixture",xlab = "X1", ylab = "X2")
legend("topright", legend = c("Group 1", "Group 2"), col = c(1, 2), pch =
↪  19)
```

This code generates a dataset from two Gaussian distributions with different means and covariances.

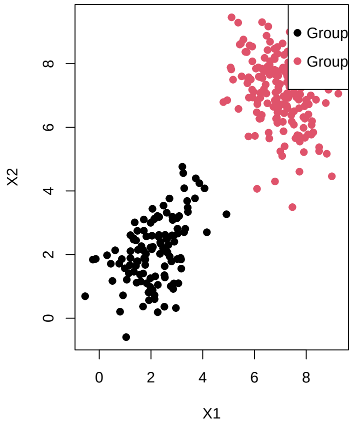
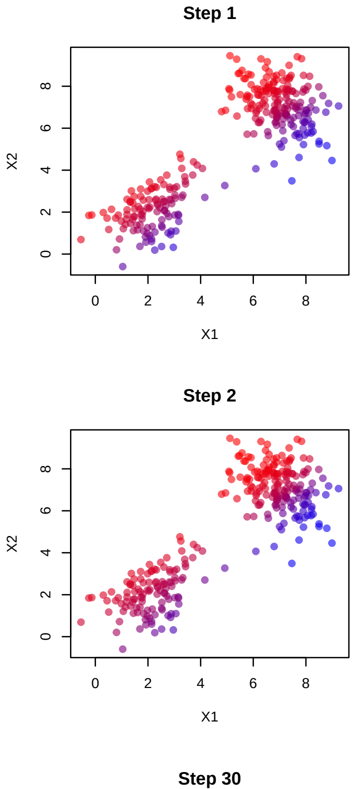


Figure 3.2: This is how the data looks like when we color points by group assigment. In practice, this group assigment is not observed.

EM Algorithm for GMMs Now we apply the EM algorithm to estimate the parameters of the Gaussian mixture model. The following R code implements the EM algorithm:

```
# Function to calculate Gaussian density
dGaussian <- function(x, mu, sigma) {exp(-0.5 * t(x - mu) %*% solve(sigma)
↪  %*% (x - mu)) / sqrt(det(2 * pi * sigma))}

# Initial estimates
K <- 2  # number of components
pi_hat <- c(0.5, 0.5)  # initial guess for mixing proportions
mu_hat <- list(c(0, 0), c(1, -1))  # initial guess for means
sigma_hat <- list(diag(2), diag(2))  # initial guess for covariances

# Modified EM algorithm to extract responsibilities at specific iterations
em_gmm_with_responsibilities <- function(x, K, pi_hat, mu_hat, sigma_hat,
↪  steps_to_plot, max_iter = 100, tol = 1e-4) {
```



```
  n <- nrow(x)
  d <- ncol(x)

  log_likelihoods <- numeric(max_iter)
  responsibility_snapshots <- list()

  for (iter in 1:max_iter) {
    # E-step: Compute the responsibilities
    gamma <- matrix(0, n, K)
    for (i in 1:n) {
      for (k in 1:K) {
        gamma[i, k] <- pi_hat[k] * dGaussian(x[i, ], mu_hat[[k]],
        ↪   sigma_hat[[k]])
      }
      gamma[i, ] <- gamma[i, ] / sum(gamma[i, ])  # Normalize
    }

    # Save responsibilities if requested
    if (iter %in% steps_to_plot) {
      responsibility_snapshots[[as.character(iter)]] <- gamma
    }

    # M-step: Update parameters
    N_k <- colSums(gamma)
    pi_hat <- N_k / n
    for (k in 1:K) {
      mu_hat[[k]] <- colSums(gamma[, k] * x) / N_k[k]
      sigma_hat[[k]] <- matrix(0, d, d)
      for (i in 1:n) {
        sigma_hat[[k]] <- sigma_hat[[k]] + gamma[i, k] * (x[i, ] -
        ↪   mu_hat[[k]]) %*% t(x[i, ] - mu_hat[[k]])
      }
      sigma_hat[[k]] <- sigma_hat[[k]] / N_k[k]
    }

    # Log-likelihood computation
    log_likelihood <- 0
    for (i in 1:n) {
      temp <- 0
      for (k in 1:K) {
        temp <- temp + pi_hat[k] * dGaussian(x[i, ], mu_hat[[k]],
        ↪   sigma_hat[[k]])
      }
      log_likelihood <- log_likelihood + log(temp)
    }

    log_likelihoods[iter] <- log_likelihood

    # Check for convergence
    if (iter > 1 && abs(log_likelihoods[iter] - log_likelihoods[iter - 1])
    ↪   < tol) {
      log_likelihoods <- log_likelihoods[1:iter]
      break
    }
  }

  list(pi = pi_hat, mu = mu_hat, sigma = sigma_hat, log_likelihood =
  ↪   log_likelihoods, responsibilities = responsibility_snapshots)
}

# Run the EM algorithm and save responsibilities after specific steps
```

```
steps_to_plot <- c(1, 2, 30)
result <- em_gmm_with_responsibilities(x, K, pi_hat, mu_hat, sigma_hat,
↪  steps_to_plot)

# Plotting responsibilities
par(mfrow = c(3, 1))
for (step in steps_to_plot) {
  gamma <- result$responsibilities[[as.character(step)]]
  colors <- rgb(gamma[, 1], 0, gamma[, 2], alpha = 0.6)  # Red for
  ↪  component 1, Blue for component 2
  plot(x, col = colors, pch = 19, main = paste("Step", step), xlab = "X1",
  ↪  ylab = "X2")
}
par(mfrow = c(1, 1))

# Plot log-likelihood progression
plot(result$log_likelihood, type = "o", main =
↪  "Log-Likelihood Progression", xlab = "Iteration", ylab =
↪  "Log-Likelihood")
```

When playing around with this simulation you can pay closer attention to the following problems:

- Are the estimated values of the parameters close to the true value?
- It this model identifiable?
- In this simulation we chose $K = 2$ both in simulations and in the estimation step. In practice we do not know K so we may want to check a bunch of different values. Think how this could be done but in an ad hoc and formal way.

3.4 *Exercises*

Exercise 3.4.1. *Show that if X has spherical distribution, then* $\mathbb{E}X = 0$ *and* $\mathrm{var}(X) = cI_m$ *for some* $c \geq 0$.

Exercise 3.4.2. *Suppose* $X = \frac{1}{\sqrt{\tau}}Z$ *with* $Z \sim N(0, I_m)$ *with* $Z \perp\!\!\!\perp \tau$. *Show that* $\mathrm{var}(X) = \mathbb{E}[\tau^{-1}]I_m$.

Exercise 3.4.3. *Verify (3.4) that gives the density of the Gaussian copula model. Then check also (3.5).*

Exercise 3.4.4. *Equation (3.6) provides the density of the Gaussian mixture model. Verify this function indeed integrates to 1.*

Exercise 3.4.5. *Propose an algorithm to generate samples from this model for fixed values of all parameters. Hint: Check Section 3.3.2.*

4
Principal Component Analysis

The primary challenge in multivariate statistics is managing a large number of variables, potentially in the millions. One approach is to model these variables directly using high-dimensional statistical techniques. Alternatively, dimensionality reduction methods can be applied to derive a smaller set of variables that capture the most significant relationships in the data. These derived variables often serve as effective substitutes for the original data.

This chapter provides a comprehensive introduction to Principal Component Analysis (PCA). While the lecture slides illustrate simple examples of how PCA can project a multivariate dataset onto two dimensions for visualization and insight, the following sections carefully develop the underlying theory.

4.1 Principal components

We start with the following problem. Given a random p-vector $X \sim (\mu, \Sigma)$ find a scalar random variable $Z = u^\top X$ with the largest variance. This is not well defined if we allow $u \in \mathbb{R}^p$. Thus, we constrain to $u \in \mathbb{R}^p$ such that $\|u\| = 1$. We say that u is a unit vector. Unit vectors define directions.

We write $X \sim (\mu, \Sigma)$ to denote that $\mathbb{E}X = \mu$ and $\mathrm{var}(X) = \Sigma$.

Recall $\mathrm{var}(u^\top X) = u^\top \Sigma u$. Thus we solve a constrained problem

$$\text{maximize} \quad u^\top \Sigma u \quad \text{subject to } u^\top u = 1.$$

Since we optimize a continuous function over a compact set, this optimum must be attained. We encode the constraint $\|u\| = 1$ as $\|u\|^2 = u^\top u = 1$ to avoid dealing with square roots.

The Lagrangian is

$$L = u^\top \Sigma u - \lambda(u^\top u - 1)$$

and every optimum of the original problem must be a stationary point of the Lagrangian. We get

$$\nabla_u L = 2\Sigma u - 2\lambda u.$$

This shows that every stationary point of L must correspond to an eigenvector of Σ with λ being the corresponding eigenvalue. If u is such an eigenvector then the value of the original function is

$$u^\top \Sigma u \;=\; \lambda u^\top u \;=\; \lambda.$$

Thus, the stationary point of L that gives the maximal value of the original function is the eigenvector corresponding to the maximal eigenvalue of Σ (which will be always strictly positive).

Suppose that we have now identified u_1 as the first principal direction and λ_1 as the corresponding variance.

We can now look for other linear combinations uncorrelated with $u_1^\top X$ with maximum variance. The uncorrelation condition is

$$0 \;=\; \mathrm{cov}(u_1 X, uX) \;=\; u_1^\top \Sigma u \;=\; \lambda_1 u_1^\top u.$$

Since $\lambda_1 > 0$, equivalently $u_1^\top u = 0$. This amount to solving

$$\text{maximize} \quad u^\top \Sigma u \quad \text{subject to } u^\top u = 1 \text{ and } u_1^\top u = 0.$$

Here the Lagrangian is

$$L \;=\; u^\top \Sigma u - \lambda(u^\top u - 1) - \nu u_1^\top u.$$

The stationary points satisfy $\Sigma u - \lambda u \;=\; \frac{\nu}{2} u_1$. If $\nu \;\neq\; 0$ we get a contradiction with $u^\top u_1 \;=\; 0$[1] and so $\nu \;=\; 0$. But in this case, we easily show that u_2 has to be the eigenvector of Σ corresponding to the second largest eigenvalue λ_2. And we can continue in this way obtaining m random variables $Z_1 \;=\; u_1^\top X, \ldots, Z_m \;=\; u_m^\top X$. By construction, the variables Z_i are uncorrelated and they are ordered by the size of their variance.

Recall the spectral theorem in Section 1.1.5, which states that we can decompose $\Sigma \;=\; U \Lambda U^\top$, where U is an orthogonal matrix and Λ is diagonal with positive diagonal entries. The *columns* of U are eigenvectors and the diagonal entries of Λ are the corresponding eigenvalues. If the entries of Λ are in decreasing order, the columns of U are the subsequent principal directions.

4.1.1 *Principal components for observed data*

Suppose we have samples $\mathbf{x}_1, \ldots, \mathbf{x}_n \;\in\; \mathbb{R}^p$. Write them as rows in a data matrix $\mathbf{X} \;\in\; \mathbb{R}^{n \times p}$. We then standardize the data so that each column has mean zero and variance one[2]. Denote the normalized data matrix by $\tilde{\mathbf{X}}$.

We can then compute the sample covariance matrix $S_n \;=\; \frac{1}{n} \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$ and proceed exactly as above replacing Σ with S_n. We again write the spectral decomposition $S_n \;=\; U \Lambda U^\top$, where U is the matrix of loadings.

[1] Multiplying by $u_1^\top$ from the left, we get $u_1^\top \Sigma u - \lambda u_1^\top u \;=\; \nu/2$, or equivalently, $(\lambda_1 - \lambda) u_1^\top u = \nu/2$.

[2] In practice we normalize data so that $S_{jj} \;=\; 1$ for $j \;=\; 1, \ldots, p$. To see why this is important perform a simple experiment: Take any preferred dataset and multiply one of the observed variables by a large number (e.g. by 1000 when changing units from kilograms to grams). See how this affects the results.

We can now use it to project the data to a d-dimensional subspace with $d < p$. Let $\boldsymbol{u}_1, \dots, \boldsymbol{u}_d$ be the first d principal directions (the first d columns of U). The **scores** are the images of the centered data points $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n$ in the coordinate system defined by the d loadings. These are the points, $\mathbf{y}_1, \dots, \mathbf{y}_n$ given by

$$\mathbf{y}_j \;=\; (\boldsymbol{u}_1^\top \boldsymbol{x}_j, \dots, \boldsymbol{u}_d^\top \boldsymbol{x}_j).$$

In other words, $\mathbf{Y} \in \mathbb{R}^{n \times d}$ with rows $\mathbf{y}_1, \dots, \mathbf{y}_n$ is obtained by taking the first d columns of $\tilde{\mathbf{X}}U$.

4.1.2 Correlation structure

Suppose that U is the matrix of principal directions, that is, $\Sigma = U\Lambda U^\top$. Assume that $X \in \mathbb{R}^p$ is centered. Let $Z = U^\top X \in \mathbb{R}^p$ then $\mathrm{var}(Z) = U^\top \Sigma U = \Lambda$.

The covariance between X and Z is

$$\mathrm{cov}(X, Z) = \mathbb{E} X Z^\top \;=\; \mathbb{E}(XX^\top U) \;=\; \Sigma U \;=\; U\Lambda.$$

Consequently, the covariance between X_i and Z_j is $\lambda_j U_{ij}$. Now X_i and Z_j have variances σ_{ii} and λ_j, respectively, and so their correlation is ρ_{ij} with

$$\rho_{ij} \;=\; \frac{\lambda_j U_{ij}}{\sqrt{\Sigma_{ii}\lambda_j}} \;=\; U_{ij}\sqrt{\frac{\lambda_j}{\Sigma_{ii}}}.$$

We may say that the proportion of variation of X_i "explained" by Z_j is ρ_{ij}^2. More precisely, suppose $\Sigma_{ii} = 1$ for all i. Then,

$$\sum_{j=1}^{m} \rho_{ij}^2 \;=\; \sum_{j=1}^{m} \lambda_j u_{ij}^2, \qquad \sum_{i=1}^{m} \rho_{ij}^2 \;=\; \lambda_j \sum_{i=1}^{m} u_{ij}^2 \;=\; \lambda_j$$

of the variation in X_i. The denominator of this expression represents the variation explained in X_i that is to be explained, and the numerator gives the amount explained by the set G.

4.2 Simple graphics for PCA

4.2.1 Scree plot and fraction of variance exlpained

Suppose $S_n = U\Lambda U^\top$ is the spectral decomposition. The diagonal entries S_{jj} of S_n are the sample variances of each of the variables. On the other hand the diagonal entries λ_j of Λ are the variances of the principal components. Note that

$$\mathrm{tr}(S_n) \;=\; \mathrm{tr}(U\Lambda U^\top) \;=\; \mathrm{tr}(\Lambda U^\top U) \;=\; \mathrm{tr}(\Lambda).$$

It follows that

Note that $S_{ii} = \mathrm{var}(\boldsymbol{e}_i^\top X)$ and so $\lambda_1 \geq \max_i(S_{ii})$.

$$S_{11} + \cdots + S_{pp} \;=\; \lambda_1 + \cdots + \lambda_m$$

and so the total variance is preserved by PCA.

Often, most of the variance is explained by a small number of principal components. The following facts should also be useful.

If we regress Y on $Z_1, \ldots, Z_k$ then the corresponding coefficient of determination (aka R^2) is $R^2 = 1 - \frac{\|\mathbf{y}-\hat{\mathbf{y}}\|^2}{\|\mathbf{y}-\bar{\mathbf{y}}\|^2}$, where $\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$ is the vector of predicted values and $\bar{\mathbf{y}} = \frac{1}{n}\mathbf{y}^\top \mathbf{1}_n$.

Proposition 4.2.1. *Suppose we regress each variable X_j on* PC_1 *and compute the R_j^2 for that regression. If X_j are standardized to have variance equal to one, then $\sum_{j=1}^p R_j^2 \;=\; \lambda_1$. If we regress on the first k principal components, then $\sum_j R_j^2 = \sum_{j=1}^k \lambda_j$.*

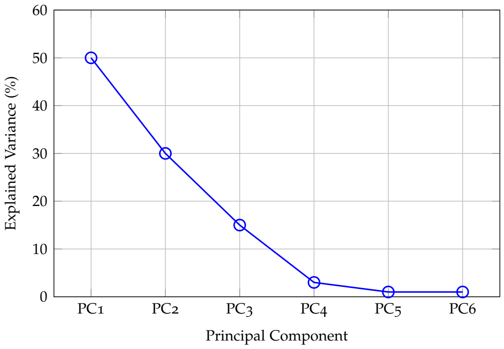


Figure 4.1: An example of the Scree plot. Here the first three components capture most of the variance. If $\Sigma_{ii} = 1$ for all i, then $\frac{\lambda_i}{m}100\%$ is the percentage of the variance explained by the i-th component.

In R simply run `princomp(data,cor=TRUE)`

4.2.2 Biplot

In the slides we presented a number of biplots. What is this? A PCA biplot is a graphical representation that shows both the observations (data points) and the principal components' directions (loadings) in a single plot. It provides insight into how the original variables contribute to the principal components and how the observations relate to each other in the space of the principal components.

The idea is simple. We compute the first two principal directions $\boldsymbol{u}_1, \boldsymbol{u}_2 \in \mathbb{R}^p$ and each observation in the dataset is projected onto the principal components. These projections are called scores. They are computed as $(\boldsymbol{u}_1^\top \mathbf{x}_i, \boldsymbol{u}_2^\top \mathbf{x}_i)$ for $i = 1, \ldots, n$. This is our 2-dimensional dataset.

The loadings represent the contribution of each original variable to the principal components. Loadings are the coefficients of the linear combinations that define each principal component. Concretely, these are obtained from the first two rows of the loading matrix U. In a biplot, these are typically shown as arrows or vectors, indicating the direction and strength of each variable's influence on the principal components.

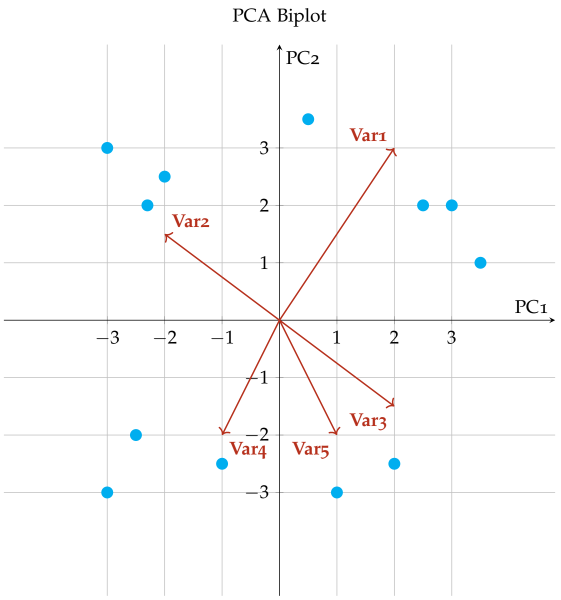


Figure 4.2: An example of a biplot with scores as blue dots and loadings as brick-red arrows.

4.3 *PCA and affine approximating subspaces*

As before let $\mathbf{X} \in \mathbb{R}^{n\times p}$ be the data matrix representing n points $\mathbf{x}_i$ in $\mathbb{R}^p$. Every r dimensional affine subspace of $\mathbb{R}^p$ can be written as $\mu + \mathrm{range}(W)$ for some $\mu \in \mathbb{R}^p$ and $W \in \mathbb{R}^{p\times r}$ with full column rank.

Thus, every point in this subspace can be written as $\mu + W\boldsymbol{z}$ for some $\boldsymbol{z} \in \mathbb{R}^r$. The goal is to find an affine subspace of dimension $\leq r$ that minimizes the sum of squared distances between the points $\mathbf{x}_i$ and their projections onto this subspace. That is, we aim to solve the following optimization problem[3]:

$$\text{minimize} \sum_{i=1}^{n} \|\mathbf{x}_i - (\mu + W\boldsymbol{z}_i)\|^2 \qquad \text{subject to } \mu \in \mathbb{R}^p, W \in \mathbb{R}^{p\times r}, \boldsymbol{z}_i \in \mathbb{R}^r.$$

Note that without loss of generality we can also assume that $\sum_{i=1}^n \boldsymbol{z}_i = \mathbf{0}$. This corresponds to changing (μ, W) to $(\mu - W(\sum_i \boldsymbol{z}_i), W)$, which defines the same affine subspace.

4.3.1 *Reformulating the problem*

Note that each summand in the objective function can be written as

$$\|\boldsymbol{x}_i - \mu\|^2 - 2\langle \boldsymbol{x}_i - \mu, W\boldsymbol{z}_i\rangle + \|W\boldsymbol{z}_i\|^2$$

It is then straightforward to first optimize over μ getting

Here we used the fact that $\sum_i z_i = \mathbf{0}$.

$$\hat{\mu} \;=\; \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - W\boldsymbol{z}_i) \;=\; \bar{\boldsymbol{x}}.$$

[3] To show this reduction, you may first optimize over μ

Define the centered data matrix $\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}_n \bar{\mathbf{x}}^\top$, where $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$ is the sample mean, and $\mathbf{1}_n$ is a vector of ones of length n. After centering, the problem becomes:

$$\text{minimize} \sum_{i=1}^n \|\tilde{\mathbf{x}}_i - W\boldsymbol{z}_i\|^2 \qquad \text{subject to } W \in \mathbb{R}^{p\times r}, \boldsymbol{z}_i \in \mathbb{R}^r.$$

This can be written in matrix form as:

$$\text{minimize}\|\tilde{\mathbf{X}} - \mathbf{Z}W^\top\|_F^2 \qquad \text{subject to } W \in \mathbb{R}^{p\times r}, \mathbf{Z} \in \mathbb{R}^{n\times r}, \qquad (4.1)$$

where $\tilde{\mathbf{X}} \in \mathbb{R}^{n\times p}$ is the centered data matrix, and $\|\cdot\|_F$ denotes the Frobenius norm[4].

4.3.2 Singular value decomposition (SVD) approach

Since the matrices $\mathbf{Z}$ and W in (4.1) are unrestricted, equivalently $\mathbf{Z}W^\top \in \mathbb{R}^{n\times p}$ has rank $\leq r$. The solution to (4.1) can be obtained via the singular value decomposition (SVD) of the data matrix. Recall that the SVD of $\tilde{\mathbf{X}}$ is:

$$\tilde{\mathbf{X}} = U\Lambda V^\top,$$

where $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{p\times p}$ are orthogonal matrices, and $\Lambda \in \mathbb{R}^{n\times p}$ is a diagonal matrix containing the singular values.

Theorem 4.3.1 (Eckart-Young Theorem). *Given $\mathbf{X} \in \mathbb{R}^{n\times p}$ with SVD $\mathbf{X} = UDV^\top$ and the problem: minimize $\|\mathbf{X} - M\|_F$ subject to* $\operatorname{rank}(M) \leq r$*, the solution is given by $\hat{M} = UD_rV^\top$, where D_r is obtained from D by removing all but the first r diagonal entries.*

For a proof and a more detailed discussion, see Section I.9 in [5].

In consequence, the best rank-r approximation to $\tilde{\mathbf{X}}$ is given by truncating the SVD to the first r singular values. That is, the optimal affine subspace is spanned by the first r columns of V, and the corresponding projected points are obtained by multiplying these directions by the singular values:

$$\tilde{\mathbf{X}}_r = U\Lambda_r V^\top.$$

4.3.3 Connection to PCA

This procedure shows that PCA is equivalent to finding the best affine subspace that approximates the data in the least squares sense. The principal components V_r represent the directions of maximum variance in the data, and the singular values Λ_r represent the amount of variance explained by each principal component.

In other words, PCA projects the data onto a lower-dimensional affine subspace in such a way that the variance of the projected data

[4] The space $\mathbb{R}^{n\times p}$ admits a natural inner product $\langle A, B\rangle = \operatorname{tr}(AB^\top)$. The Frobenius norm is the induced norm: $\|A\|_F = \sqrt{\langle A, A\rangle} = \sqrt{\sum_{i,j} A_{ij}^2}$

[5] Gilbert Strang. *Linear Algebra and Learning from Data*. Wellesley-Cambridge Press, Wellesley, MA, 2019. ISBN 9780692196380

is maximized, and the reconstruction error (in terms of squared distances) is minimized. This gives a direct geometric interpretation of PCA as finding the best-fitting affine subspace for the data.

4.4 Principal Component Regression (PCR) and Probabilistic PCA (PPCA)

4.4.1 Principal Component Regression

In some cases, PCA can be used as a preprocessing step in regression to reduce the dimensionality of the data and remove multicollinearity. By using the first few principal components as predictors in the regression model, we can obtain more stable estimates of the regression coefficients and improve the interpretability of the model.

For example, suppose we have a dataset with p variables and we perform PCA to extract the first k principal components. We can then fit a regression model using these k components as predictors:

$$\hat{Y} = \alpha + \beta_1 Z_1 + \beta_2 Z_2 + \cdots + \beta_k Z_k,$$

where $Z_1, Z_2, \ldots, Z_k$ are the first k principal components. This is known as principal component regression (PCR), and it is often used when the number of predictors is large relative to the number of observations.

4.4.2 Probabilistic PCA

There is a related generative model in which we model a random vector X through the linear relation

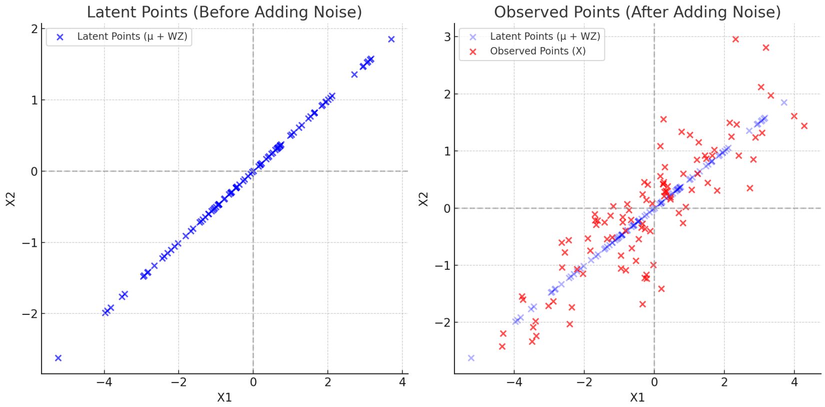
$$X \;=\; \mu + WZ + \varepsilon, \quad Z \sim N_r(0, I_r), \;\; \varepsilon \sim N_m(0, \sigma^2 I_m), \;\; Z \perp\!\!\!\perp \varepsilon.$$

To have a concrete example, suppose $r = 1$, $m = 2$, $\mu = (0, 0)$, and $W = (1, \frac{1}{2})$. Sampling Z from $N(0, 1)$ produces blue points in Figure 4.4.2. Additionally, sampling $\varepsilon \sim N_2(0, I_2)$ and adding to the corresponding blue points, gives the red points, which are the samples from the underlying PPCA model.

It is not hard to see that $X|Z$ is Gaussian and so X is also marginally Gaussian with mean μ and covariance $\sigma^2 I_m + WW^\top$. It turns out that this model admits closed form formula for the MLE, which mimics the PCA solution.

4.5 PCA and matrix completion

Suppose that now $\mathbf{X} \in \mathbb{R}^{n \times p}$ contains some missing data denoted as NA. We would like to be able to:



1. Estimate the missing entries.

2. Compute the PCA in presence of missing data.

We saw in Theorem 4.3.1 that SVD(r) solves

$$\text{minimize}\|\mathbf{X}-M\|_F^2 \qquad \text{subject to } \text{rank}(M)\leq r.$$

Let $\Omega \in \{0,1\}^{n\times p}$ be such that $\Omega_{ij} = 0$ if the corresponding entry of $\mathbf{X}$ is NA and $\Omega_{ij} = 1$ otherwise. Let $\Omega\circ\mathbf{X}$ be the Hadamard (entrywise) product:

$$(\Omega\circ\mathbf{X})_{ij} = \Omega_{ij}\mathbf{X}_{ij} = \begin{cases} 0 & \text{if } \mathbf{X}_{ij} = \text{NA} \\ \mathbf{X}_{ij} & \text{otherwise.}\end{cases}$$

Using this mask, we can frame the optimization problem for estimating missing entries and performing PCA as:

$$\text{minimize } \|\Omega\circ(\mathbf{X}-M)\|_F^2 \qquad \text{subject to } \text{rank}(M)\leq r.$$

This problem is more challenging because the missing entries make it non-trivial to directly solve for M. Unlike the standard PCA formulation, this problem does not have an explicit solution, and we solve it iteratively using algorithms like `hardimpute`.

4.5.1 *Hardimpute algorithm*

The `hardimpute` algorithm iteratively solves the matrix completion problem by alternating between two steps:

1. Filling in missing entries of the matrix using the current estimate M^{j-1}.

2. Low-rank approximation: Updating M^j by projecting onto the space of rank-r matrices.

This is the algorithm:

1. Initialize $M^0 = \mathbf{0}_{n \times p}$.

2. Loop over $j = 1, 2, \ldots$ until convergence

   - $\mathbf{X}^* \;=\; \Omega \circ \mathbf{X} + (\mathbf{1}_{n \times p} - \Omega) \circ M^{j-1}$
   - $M^j \;=\; \arg\min_{M:\mathrm{rk}(M) \leq r} \|\mathbf{X}^* - M\|_F^2$.

It is important to note that, in this algorithm, the objective function $\|\Omega \circ (\mathbf{X} - M)\|_F^2$ decreases monotonically with each iteration because (i) the imputation step does not increase the objective and (ii) the low-rank projection step minimizes the Frobenius norm for a given $\mathbf{X}^*$. Consequently, this algorithm is a descent algorithm and so it converges to a stationary point.

Since the objective is not convex, we have no guarantee that the solution is a global optimum. In practice however it works extremely well; see `softImpute`. Here the name `softimpute` refers to an alternative approach, which uses a nuclear norm penalty (convex relaxation of the rank constraint) instead of the hard rank constraint. `softimpute` solves:

$$\text{minimize} \quad \|\Omega \circ (\mathbf{X} - M)\|_F^2 + \lambda \|M\|_*,$$

where $\|M\|_*$ is the nuclear norm (sum of singular values). This convex formulation guarantees convergence to a global minimum, though it may produce a biased solution for low-rank matrices.

4.6 Appendix: Covariance matrix estimation in high dimensions

In PCA we use the sample covariance matrix S_n as a natural estimator of the population covariance matrix Σ. We note however that this is only a good estimator when n is much larger than m. Otherwise, we may need to use an alternative estimator. In this section we very briefly overview some of the existing techniques.

4.6.1 Motivating alternative estimators

As noted in Section 1.3.3, the sample covariance matrix is a natural estimator due to its small bias, which diminishes rapidly as n increases. Furthermore, it can be shown to be asymptotically Gaussian under the assumption that the underlying dimension p remains fixed. However, this asymptotic analysis becomes inadequate in scenarios where n is not significantly larger than p, as the standard assumptions no longer hold.

For concreteness, suppose that we measure quality of this estimator by analyzing the distance $\|S_n - \Sigma\|$ in the operator norm

$$\|S_n - \Sigma\| \;:=\; \max_{\|\boldsymbol{u}\|=1} \|(S_n - \Sigma)\boldsymbol{u}\|.$$

Under mild moment conditions, an argument based on the classical law of large numbers can be used to show that the difference $\|S_n - \Sigma\|$ converges to zero in probability as $n \to \infty$. Consequently, S_n is a consistent estimator of the population covariance Σ in the classical setting.

What happens when we also allow p to tend to infinity? To make this question a bit more formal, consider sequences of problems (S_n, Σ) indexed by the pair (n, p), and suppose that we allow both n and p grow with their ratio remaining fixed, say $p/n = \gamma \in (0, 1)$. In Figure 4.3 we plot the results of simulations for the case when $\Sigma = I_m$ with each $\mathbf{x}_i \sim N(\mathbf{0}, I_m)$.

Using n samples we compute S_n and compute its eigenvalues. They should be all nonnegative and actually positive if $n \geq p$. We then construct a histogram for these eigenvalues. Since in our simulations n is large (we take $n = 1000$), classical theory suggests, all eigenvalues should be close to 1 as S_n should be close to I_p. But this is not what we observe. It actually seems that the eigenvalues follow a distribution that is well spread out; cf. Figure 4.3.

This situation has been well studied in random matrix theory. The underlying eigenvalue distribution is called the Marchenko-Pastur law. Define $\lambda_{\min} = (1 - \sqrt{\gamma})^2$ and $\lambda_{\max} = (1 + \sqrt{\gamma})^2$ then the law admits density

$$f_{\mathrm{MP}}(\lambda) \;=\; \frac{1}{2\pi\gamma\lambda}\sqrt{(\lambda_{\max} - \lambda)(\lambda - \lambda_{\min})} \qquad \text{for } \lambda_{\min} \leq \lambda \leq \lambda_{\max}.$$

If you want to play around with this, this is R code that produces one of the plots in Figure 4.3.

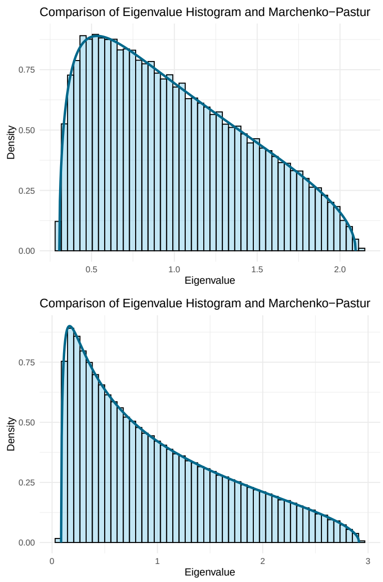


Figure 4.3: The histogram of eigenvalues and the corresponding Marchenko-Pastur law for $n = 1000$ and $p = 200$ (equiv. $\gamma = 0.2$) on the top, $p = 500$ (equiv. $\gamma = 0.5$) in the bottom.

```
 # Load necessary libraries
library(MASS)
library(ggplot2)

# Simulation parameters
n <- 1000          # Number of samples
p <- 200          # Dimensionality
n_sim <- 100    # Number of simulations

# Function to compute eigenvalues of the sample covariance matrix
compute_eigenvalues <- function(n, p) {
  data <- mvrnorm(n = n, mu = rep(0, p), Sigma = diag(p))    sample_cov <-
  ↪  cov(data)  # Compute sample covariance matrix
  eigen(sample_cov)$values  # Return eigenvalues
}

# Perform simulations
```

```
set.seed(42)  # For reproducibility
eigenvalues_list <- replicate(n_sim, compute_eigenvalues(n, p), simplify =
↪  FALSE)
all_eigenvalues <- unlist(eigenvalues_list)

# Parameters for the Marchenko-Pastur density
gamma <- p / n  # Aspect ratio
lambda_min <- (1 - sqrt(gamma))^2  # Lower bound of the MP support
lambda_max <- (1 + sqrt(gamma))^2  # Upper bound of the MP support

# Marchenko-Pastur density function
mp_density <- function(lambda, gamma) {
  if (lambda >= lambda_min && lambda <= lambda_max) {
    return((1 / (2 * pi * gamma)) * sqrt((lambda_max - lambda) * (lambda -
    ↪  lambda_min)) / lambda)
  } else {return(0)}
}

# Generate MP density values
lambda_seq <- seq(lambda_min, lambda_max, length.out = 500)
mp_values <- sapply(lambda_seq, mp_density, gamma = gamma)

# Plot normalized histogram and MP density
ggplot(data = data.frame(eigenvalue = all_eigenvalues), aes(x =
↪  eigenvalue)) +
  geom_histogram(aes(y = ..density..), bins = 50, color = "black", fill =
  ↪  "skyblue", alpha = 0.5) +
  geom_line(data = data.frame(lambda = lambda_seq, density = mp_values),
  ↪  aes(x = lambda, y = density), color = "deepskyblue4", size = 1.2) +
  labs(title =
  ↪  "Comparison of Eigenvalue Histogram and Marchenko-Pastur Density",
    x = "Eigenvalue",y = "Density") +  theme_minimal()
```

The Marchenko-Pastur law is an asymptotic result, albeit with a non-classical flavor, as it allows both the sample size n and the dimension p to diverge simultaneously. Similar to classical asymptotics, it provides an approximation for the distribution of eigenvalues even when n and p are finite. When p is not negligible relative to n, this approximation significantly outperforms the classical one.

4.6.2 *Review of alternative estimators*

If the covariance matrix has some special structure, incorporating this structure in the estimation procedures can greatly help in high-dimensional scenarios. Regularization introduces constraints or penalties to improve the stability and accuracy of covariance estimates. Key approaches include:

Linear shrinkage. The linear shrinkage estimator is given by:

$$\hat{\Sigma}_{\text{ridge}} = (1-\lambda)S_n + \lambda I_m,$$

where $\lambda \in [0,1]$ is the shrinkage parameter. Choosing λ appropriately (e.g., via cross-validation) balances bias and variance, ensuring better performance in high-dimensional settings.

The intuition behind shrinking toward the identity matrix arises from several considerations. First, as we already noted, in high-dimensional settings where $p \approx n$ or $p > n$, the sample covariance matrix S_n is often unstable, highly variable, or even singular (non-invertible). Adding λI_p stabilizes the estimate by ensuring that $\hat{\Sigma}_{\text{ridge}}$ is well-conditioned and invertible. Second, the shrinkage estimator introduces bias by pulling S_n toward I_p, but it reduces variance, which can dominate in high dimensions. A well-chosen λ balances this trade-off, leading to improved estimation accuracy for the covariance matrix.

Shrinking toward I_p corresponds to assuming, in part, that the covariance structure is close to isotropic (equal variance in all directions) unless there is sufficient evidence in the data to suggest otherwise. In cases where the data are strongly anisotropic (variances and correlations differ widely across directions), the assumption of isotropy embedded in I_p may lead to excessive bias. Shrinking toward a more informed matrix may yield better performance.

Graphical Lasso (Sparse Precision Matrix Estimation). For many high-dimensional datasets, it is reasonable to assume that variables are conditionally independent given others. This sparsity assumption is leveraged by estimating the precision matrix $\Theta = \Sigma^{-1}$ using the graphical lasso:

$$\hat{\Theta} = \arg\min_{\Theta \succ 0} \left\{ \operatorname{tr}(\hat{\Sigma}\Theta) - \log\det(\Theta) + \lambda \|\Theta\|_1 \right\},$$

where $\|\Theta\|_1 = \sum_{i,j} |\Theta_{ij}|$ promotes sparsity. The resulting estimate $\hat{\Theta}$ can be inverted to approximate Σ. We discuss this estimator in much more detail in Chapter 7.

Factor Models. In many applications, the data can be modeled as having a low-rank structure plus noise. A covariance matrix in such settings can be decomposed as:

$$\Sigma = LL^\top + \Psi,$$

where $L \in \mathbb{R}^{p \times r}$ $(r \ll p)$ is a low-rank factor matrix, and Ψ is a diagonal matrix capturing idiosyncratic variances. Estimation could proceed by minimizing reconstruction error:

$$\min_{L,\Psi} \|S_n - LL^\top - \Psi\|_F^2.$$

We discuss the closely related Factor Analysis model in Chapter 8.

Thresholding-Based Methods. A simple and effective approach for high-dimensional covariance estimation is to threshold small entries in S_n, which helps reduce noise and enforce sparsity. For a given threshold $\tau > 0$, the thresholded covariance estimate is:

$$\hat{\Sigma}_{\text{thresh}} = \left[(S_n)_{ij} \cdot \mathbb{I}(|(S_n)_{ij}| > \tau) \right]_{i,j}.$$

Thresholding methods rely on the assumption that most entries in the covariance matrix are small or zero, which is common in many applications.

Banding and Tapering for Structured Covariance Matrices

For structured data, such as time series or spatial data, covariance matrices often exhibit banded or tapering structures, where correlations decay with distance. It is them natural to impose this structure in the estimation process.

5
Some Other Dimension Reduction Methods

Dimensionality reduction techniques beyond PCA are often necessary when the underlying data structure is nonlinear or when variance-based methods like PCA do not capture enough of the essential relationships in the data. The goal of this chapter is to show what are the possible ways to handle non-linearity with preserving some of the computational advantages of linear methods. We do it by introducing two state-of-the-art techniques: Uniform Manifold Approximation and Projection (UMAP) and Autoencoders.

This part of the lecture will be presented on slides and will be relatively high-level.

5.1 Multidimensional Scaling (MDS)

Multidimensional Scaling (MDS) is a statistical technique used to represent data points in a lower-dimensional space while preserving their pairwise distances as faithfully as possible.

5.1.1 Problem Setup

Given n objects, let $\Delta = [\delta_{ij}]$ be an $n \times n$ **dissimilarity matrix**. A dissimilarity matrix is simply a symmetric matrix with zero diagonal entries and nonnegative off-diagonal entries. Its entry δ_{ij} represents the dissimilarity between objects i and j. In our setting the objects will be the data points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$ and δ_{ij} will be their mutual distances but this setting is much more general.

The goal of MDS is to find a configuration of points $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$ in a d-dimensional space ($d \ll n$) such that:

$$d_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\| \approx \delta_{ij}.$$

5.1.2 Classical MDS Solution

Classical MDS assumes the dissimilarities δ_{ij} are Euclidean distances, $\delta_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$. Let $D \odot D = [\delta_{ij}^2]$ be the matrix of squared dissimilarities (Hadamard product of D with itself). Note that the entries of $\mathbf{X}\mathbf{X}^\top$ satisfy $(\mathbf{X}\mathbf{X}^\top)_{i,j} = \mathbf{x}_i^\top \mathbf{x}_j$ and $\|\mathbf{x}_i - \mathbf{x}_j\|^2 = (\mathbf{X}\mathbf{X}^\top)_{i,i} + (\mathbf{X}\mathbf{X}^\top)_{j,j} - 2(\mathbf{X}\mathbf{X}^\top)_{i,i}$. Thus,

$$D \odot D \;=\; \operatorname{diag}(\mathbf{X}\mathbf{X}^\top)\mathbf{1}\mathbf{1}^\top + \mathbf{1}\mathbf{1}^\top \operatorname{diag}(\mathbf{X}\mathbf{X}^\top) - 2\mathbf{X}\mathbf{X}^\top.$$

Recall that the centering matrix $H = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ satisfies $H\mathbf{1} = \mathbf{0}$ and so

$$B \;:=\; -\frac{1}{2}H(D \odot D)H \;=\; H\mathbf{X}(H\mathbf{X})^\top \;=\; \tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top.$$

A natural way to get a lower dimensional embedding of the centered data $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n$ in a d dimensional space in a way that approximately preserves the mutual distances is by approximating the matrix B with a d rank matrix $\mathbf{Y}\mathbf{Y}^\top$ with $\mathbf{Y} \in \mathbb{R}^{n \times d}$. We already saw in Theorem 4.3.1 how to solve this problem.

As suggested by Theorem 4.3.1, we first perform eigenvalue decomposition: $B = V\Lambda V^\top$, or equivalently, singular value decomposition of $H\mathbf{X}$. Then, we select the top d eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_d$ (assumed positive) and their corresponding eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_d$. The $n \times d$ configuration matrix is:

$$\mathbf{Y} = U_d \Lambda_d^{1/2},$$

where U_d contains the top d eigenvectors, and $\Lambda_d^{1/2}$ is the diagonal matrix of the square roots of the top d eigenvalues.

5.1.3 Duality between classic MDS and PCA

The solution for MDS was given by d principal eigenvectors of $B = H\mathbf{X}(H\mathbf{X})^\top \in \mathbb{S}^n$. On the other hand, the principal directions in PCA were defined as the d principal eigenvectors of $(H\mathbf{X})^\top H\mathbf{X} = \mathbf{X}^\top H\mathbf{X} = \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} \in \mathbb{S}^p$.

Suppose $n \leq p$. Let $\boldsymbol{u}_i \in \mathbb{R}^p$ denote the i-th principal component loading vector, where $\|\boldsymbol{u}_i\| = 1$. Thus, we have $(H\mathbf{X})^\top H\mathbf{X} = U\Lambda U^\top$, where $U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p]$ and $\Lambda \in \mathbb{S}^p$ is the diagonal matrix with the corresponding eigenvalues. By the singular value decomposition we have, $H\mathbf{X} = V\tilde{\Lambda}^{1/2}U^\top$ for some $V \in O(n)$. Here $\tilde{\Lambda} \in \mathbb{R}^{n \times p}$ is a matrix such that $(\tilde{\Lambda}^{1/2})_{ii} = \sqrt{\lambda_i}$ for $i = 1, \ldots, p$ and all the other entries are zero. Thus,

$$H\mathbf{X}(H\mathbf{X})^\top = V \begin{bmatrix} \Lambda & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} V^\top.$$

Recall from Section 4.1.1 that the scores in PCA after taking the first d principal directions is given by the first d columns of $H\tilde{\mathbf{X}}U = V\tilde{\Lambda}^{1/2}$. In other words, we get the following result.

Theorem 5.1.1. *The classical MDS problem is equivalent to the PCA problem.*

5.1.4 Non-linearity and MDS

In the previous section, we demonstrated that classical MDS yields the same solution as PCA. However, MDS possesses a broader applicability. For instance, the dissimilarities can be calculated using non-Euclidean distances between $\mathbf{x}_i$'s, or they may not necessarily represent distances.

5.2 Laplacian Eigenmaps (Spectral Embedding)

Laplacian Eigenmaps, also known as spectral embedding, is a nonlinear dimensionality reduction technique that constructs low-dimensional representations of data while preserving local geometric structure. It is widely used in manifold learning and graph-based machine learning.

5.2.1 Problem Setup

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ be data points in a high-dimensional space $\mathbb{R}^p$. The goal is to map these points into a low-dimensional space $\mathbb{R}^d$ ($d \ll p$) such that local proximity relationships in the original space are preserved.

We start by constructing a weighted graph $G = (V, E, W)$, where:

- $V = \{1, 2, \ldots, n\}$ represents the data points.
- E represents edges between points based on some notion of proximity (e.g., k-nearest neighbors or ϵ-neighborhoods).
- W is the weight matrix, where W_{ij} represents the similarity between points $\mathbf{x}_i$ and $\mathbf{x}_j$.

Two common methods to define W are:

- **Binary Weights:**

$$W_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_j \text{ is a neighbor of } \mathbf{x}_i, \\ 0, & \text{otherwise.} \end{cases}$$

- **Heat Kernel:**

$$W_{ij} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), & \text{if } \mathbf{x}_j \text{ is a neighbor of } \mathbf{x}_i, \\ 0, & \text{otherwise.} \end{cases}$$

Here, $\sigma > 0$ controls the scale of the similarity.

5.2.2 *Graph Laplacian*

To compute the Laplacian Eigenmaps, we first define the graph Laplacian, which captures the structure of the graph.

The degree matrix D is a diagonal matrix where $D_{ii} = \sum_{j=1}^{n} W_{ij}$ represents the degree of vertex i. The **graph Laplacian** is defined by

← Exercise 5.5.2

$$L \; := \; D - W. \tag{5.1}$$

The **normalized graph Laplacian** is

$$L_{\mathrm{N}} = D^{-1/2} L D^{-1/2}.$$

5.2.3 *Objective Function*

Laplacian Eigenmaps minimize an objective that preserves local distances by penalizing large differences between embeddings of connected points. Let $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^d$ be the embedding coordinates of the points and let $\mathbf{Y} \in \mathbb{R}^{n \times d}$ be the corresponding data matrix. The objective is:

$$\text{minimize } \frac{1}{2} \sum_{i,j} W_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2$$

with respect to $\mathbf{Y} \in \mathbb{R}^{n \times d}$. The interpretation here is clear. If the weight W_{ij} is high, that is when $\mathbf{x}_i$ and $\mathbf{x}_j$ are close, it enforces $\mathbf{y}_i$ and $\mathbf{y}_j$ to be close too.

To efficiently deal with this objective function, we formulate the following result.

Lemma 5.2.1. *We have* $\frac{1}{2} \sum_{i,j} W_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 = \operatorname{tr}(L\mathbf{Y}\mathbf{Y}^\top)$.

Proof. First note that $(\mathbf{Y}\mathbf{Y}^\top)_{i,j} = \mathbf{y}_i^\top \mathbf{y}_j$ and that the diagonal entries of $\mathbf{1}\mathbf{1}^\top W$ and $W\mathbf{1}\mathbf{1}^\top$ are in both cases equal to the diagonal entries of D[1]. Also, directly by definition, $W\mathbf{1} = D\mathbf{1}$. We also easily verify that the matrix

$$E \; = \; \operatorname{diag}(\mathbf{Y}\mathbf{Y}^\top)\mathbf{1}\mathbf{1}^\top + \mathbf{1}\mathbf{1}^\top \operatorname{diag}(\mathbf{Y}\mathbf{Y}^\top) - 2\mathbf{Y}\mathbf{Y}^\top$$

has entries $E_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|^2$ and so

$$\begin{aligned}
\tfrac{1}{2} \sum_{i,j} W_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \quad &= \quad \tfrac{1}{2} \operatorname{tr}(WE) \\
&= \quad \tfrac{1}{2} \operatorname{tr}(W \operatorname{diag}(\mathbf{Y}\mathbf{Y}^\top)\mathbf{1}\mathbf{1}^\top) + \tfrac{1}{2} \operatorname{tr}(W\mathbf{1}\mathbf{1}^\top \operatorname{diag}(\mathbf{Y}\mathbf{Y}^\top)) - \operatorname{tr}(W\mathbf{Y}\mathbf{Y}^\top) \\
&= \quad \tfrac{1}{2} \operatorname{tr}(\mathbf{1}\mathbf{1}^\top W \operatorname{diag}(\mathbf{Y}\mathbf{Y}^\top)) + \tfrac{1}{2} \operatorname{tr}(W\mathbf{1}\mathbf{1}^\top \operatorname{diag}(\mathbf{Y}\mathbf{Y}^\top)) - \operatorname{tr}(W\mathbf{Y}\mathbf{Y}^\top) \\
&= \quad \tfrac{1}{2} \operatorname{tr}(D \operatorname{diag}(\mathbf{Y}\mathbf{Y}^\top)) + \tfrac{1}{2} \operatorname{tr}(D \operatorname{diag}(\mathbf{Y}\mathbf{Y}^\top)) - \operatorname{tr}(W\mathbf{Y}\mathbf{Y}^\top) \\
&= \quad \operatorname{tr}(D \operatorname{diag}(\mathbf{Y}\mathbf{Y}^\top)) - \operatorname{tr}(W\mathbf{Y}\mathbf{Y}^\top) \\
&= \quad \operatorname{tr}(D\mathbf{Y}\mathbf{Y}^\top) - \operatorname{tr}(W\mathbf{Y}\mathbf{Y}^\top) \; = \; \operatorname{tr}(L\mathbf{Y}\mathbf{Y}^\top).
\end{aligned}$$

□

[1] Indeed, $(\mathbf{1}\mathbf{1}^\top W)_{i,i} = \boldsymbol{e}_i^\top \mathbf{1}\mathbf{1}^\top W \boldsymbol{e}_i = \mathbf{1}^\top W \boldsymbol{e}_i$ and $(W\mathbf{1}\mathbf{1}^\top)_{i,i} = \boldsymbol{e}_i^\top W \mathbf{1}\mathbf{1}^\top \boldsymbol{e}_i = \boldsymbol{e}_i^\top W\mathbf{1}$. Both quantities are equal because W is symmetric.

To avoid trivial solutions (e.g., $\mathbf{Y} = \mathbf{0}$), we impose:

- **Orthogonality constraint:** $\mathbf{Y}^\top D\mathbf{Y} = I_d$.
- **Centering constraint (optional):** $\mathbf{Y}^\top D\mathbf{1} = 0$.

The optimization problem becomes:

$$\text{minimize} \quad \text{tr}(\mathbf{Y}^\top L\mathbf{Y}) \quad \text{subject to } \mathbf{Y}^\top D\mathbf{Y} = I.$$

Denoting $\tilde{\mathbf{Y}} = D^{1/2}\mathbf{Y}$ we equivalently get to minimize

$$\text{tr}(\tilde{\mathbf{Y}}D^{-1/2}LD^{-1/2}\tilde{\mathbf{Y}}^\top) \;=\; \text{tr}(\tilde{\mathbf{Y}}L_{\text{N}}\tilde{\mathbf{Y}}^\top)$$

subject to the constraint that the columns of $\tilde{\mathbf{Y}}$ are orthonormal. When discussing PCA we already argued that this problem is equivalent to finding the $d+1$

This is solved by finding the first $d+1$ eigenvectors of the eigenproblem:

$$L_{\text{N}}\mathbf{v} = \lambda D\mathbf{v},$$

where λ are the eigenvalues, and $\mathbf{v}$ are the corresponding eigenvectors. We can stuck these vectors as columns of $\tilde{Y}$. Note that the first column is always $D\mathbf{1}$ and so it can be discarded. The remaining d columns give us the desired embedding.

5.2.4 *Example: Twisted curve*

To have a simple illustration of this method, consider data that align along a twisted curve as depicted in Figure 5.1. The following code takes the corresponding data (blue points on the curve) and feeds them both to PCA and Laplacian eigenmap method.

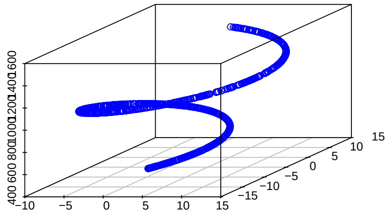


Figure 5.1: Twisted curve in 3D.

```
# Load libraries
library(Rdimtools)
library(ggplot2)
library(scatterplot3d)
library(gridExtra)

# Generate twisted curve data
set.seed(123)
n <- 1000
t <- 1.5 * pi * (1 + 2 * runif(n))  # Angle
x <- t * cos(t)
y <- t * sin(t)
z <- 100*t   # Add height to the roll
twisted <- data.frame(x = x, y = y, z = z)

# Plot 3D twisted curve
scatterplot3d(twisted$x, twisted$y, twisted$z, color = 'blue',
              main = "Twisted curve", xlab = "x", ylab = "y", zlab = "z")

# Convert twisted curve data to matrix
twisted_matrix <- as.matrix(twisted)
```

```
# Apply PCA
pca_result <- prcomp(twisted_matrix, center = TRUE, scale. = TRUE)
pca_2d <- as.data.frame(pca_result$x[, 1:2])  # Select first 2 principal
↪  components
colnames(pca_2d) <- c("PC1", "PC2")

# Apply Laplacian Eigenmaps
lapeig_result <- do.lapeig(twisted_matrix, ndim = 2, type =
↪  c("proportion", 0.1))
lapeig_2d <- as.data.frame(lapeig_result$Y)
colnames(lapeig_2d) <- c("LE1", "LE2")

# Plot PCA result
p1 <- ggplot(pca_2d, aes(x = PC1, y = PC2, color = z)) +
  geom_point() +
  labs(title = "PCA Projection",
       x = "Principal Component 1", y = "Principal Component 2") +
  theme_minimal()

# Plot Laplacian Eigenmaps result
p2 <- ggplot(lapeig_2d, aes(x = LE1, y = LE2, color = z)) +
  geom_point() +
  labs(title = "Eigenmaps Projection",
       x = "Laplacian Dimension 1", y = "Laplacian Dimension 2") +
  theme_minimal()

# Display plots side by side using gridExtra
grid.arrange(p1, p2, ncol = 2)
```

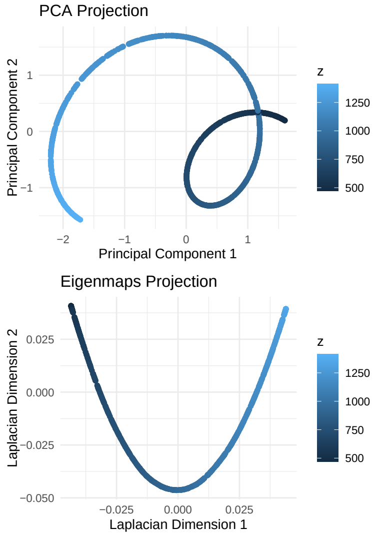


Figure 5.2: PCA and Laplacian eigenmap projections of the twisted curve data. The coloring corresponds to the value of the z-axis.

As observed in Figure 5.2 the principal directions chosen by PCA project the data so that points that are very far away from each other get glued together. On the other hand the nonlinear embedding defined by the eigenmap separated all the points. In other words, the method is able to learn the underlying manifold and propose an embedding that preserves the geometry.

Remark 5.2.2. *I did not provide any explanation behind the Laplacian eigenmap method. This story is however very beautiful and I recommend you to read about this if you have the necessary background. On a very high level: The Laplace operator Δ is a second-order differential operator in $\mathbb{R}^p$. For each twice differentiable function it is defined by*

$$\Delta f = \sum_{i=1}^{p} \frac{\partial^2 f}{\partial x_i^2}.$$

The spectrum of the Laplace operator consists of all eigenvalues $\lambda \in \mathbb{R}$ for which there is a corresponding eigenfunction f with $\Delta f = \lambda f$. This is known as the Helmholtz equation. If Ω is a bounded domain in $\mathbb{R}^p$, then the eigenfunctions of the Laplacian are an orthonormal basis for the Hilbert space $L^2(\Omega)$.

Now, more generally, on a smooth Riemannian manifold $\mathcal{M}$, the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ is a differential operator that generalizes the Laplacian from Euclidean space. Eigenfunctions of $\Delta_{\mathcal{M}}$ (solutions to $\Delta_{\mathcal{M}} f = \lambda f$)

reveal the intrinsic geometry of the manifold. When we do not have explicit access to the manifold $\mathcal{M}$, we work with a discrete approximation using a graph constructed from the data points $\{\mathbf{x}_i\} \subset \mathcal{M}$. The graph Laplacian acts as a discrete version of the Laplace-Beltrami operator. As the number of data points increases and the graph becomes finer, L converges to $\Delta_{\mathcal{M}}$ under certain conditions. In particular, the eigenvectors of L approximate the eigenfunctions of $\Delta_{\mathcal{M}}$.

5.2.5 Properties and Interpretations

- **Connection to Manifold Learning:** Laplacian Eigenmaps approximate the low-dimensional manifold structure by using the eigenfunctions of the graph Laplacian as basis functions.
- **Locality:** The weights W_{ij} ensure that only local relationships are preserved, making Laplacian Eigenmaps suitable for nonlinear structures.
- **Spectral Clustering:** The eigenvectors of the Laplacian are also used in spectral clustering, where they define clusters in the data.
- **Dimensionality Reduction:** Laplacian Eigenmaps is a form of nonlinear dimensionality reduction, complementing methods like PCA that are linear.

In what follows we discuss a refinement of this method and give some examples.

5.3 Uniform Manifold Approximation and Projection (UMAP)

One of the limitations of PCA is that it works well only when the first 2-3 principal components account for most of the variability in the data. UMAP is a powerful dimensionality reduction technique designed to handle nonlinear structures in high-dimensional datasets and preserve more complex relationships than PCA. It is much more flexible and retains some of PCA's speed and scalability. The exposition here is based on the excellent tutorial.[2] The original UMAP paper provides additional details but is mathematically involved (you may need to know some basics of category theory).

5.3.1 Key idea

Consider data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^p$, where $\mathbb{R}^p$ can be very big. Like for eigenmaps, the key idea of UMAP and many similar methods is to embed the data points into a much lower dimensional subspace $\mathbb{R}^d$, say $\mathbf{y}_i = \textsc{umap}(\mathbf{x}_i)$ so that two points $\mathbf{y}_i$ and $\mathbf{y}_j$ are close to each other if and only if the corresponding $\mathbf{x}_i$, $\mathbf{x}_j$ are close to each other.

[2] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Uniform manifold approximation and projection (umap) and its variants: tutorial and survey. *arXiv:2109.02508*, 2021

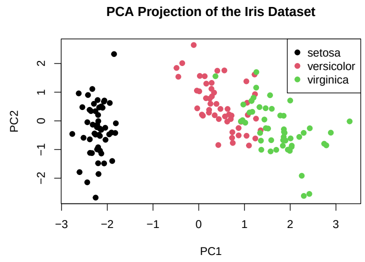
5.3.2 Example: Applying UMAP to the Iris Dataset

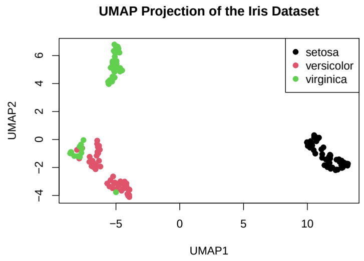
The Iris dataset is a classic example in multivariate statistics. It consists of 150 observations of iris flowers, with four continuous features (sepal length, sepal width, petal length, petal width) and a categorical label (species).

A linear method like PCA can give some separation between species in the first two principal components, but this separation is not always very distinct. UMAP, on the other hand, attempts to preserve both local and global structures of the data in a low-dimensional embedding. As a result, UMAP often produces a more visually separable and intuitively meaningful representation.

The following code compares the performance of both PCA and UMAP on the Iris dataset.

```r
# Load necessary libraries
library(umap)

# Load the Iris data
data(iris)
iris_data <- iris[, 1:4]          # numeric features
iris_labels <- iris$Species       # species labels

set.seed(123)

#### UMAP ####
umap_result <- umap(iris_data)

# UMAP Plot
plot(umap_result$layout, col = as.integer(iris_labels), pch = 19, xlab =
↪  "UMAP1", ylab = "UMAP2", main = "UMAP Projection of the Iris Dataset")
legend("topright", legend = levels(iris_labels), col = 1:3, pch = 19)

#### PCA ####
pca_result <- prcomp(iris_data, scale. = TRUE)
pca_scores <- pca_result$x

# PCA Plot using the first two principal components
plot(pca_scores[, 1], pca_scores[, 2], col = as.integer(iris_labels), pch
↪  = 19, xlab = "PC1", ylab = "PC2", main =
↪  "PCA Projection of the Iris Dataset")
legend("topright", legend = levels(iris_labels), col = 1:3, pch = 19)
```





In the resulting plots, we see that observations from the same species form relatively distinct clusters. Compared to a simple PCA plot, these groups may be more compact and better separated. This suggests that UMAP can capture nonlinear relationships and subtle differences between classes that PCA might not.

5.3.3 High-level description of the algorithm

The UMAP algorithm performs dimensionality reduction in three key steps:

1. construction of a k-nearest neighbor (kNN) graph,
2. initialization using spectral embedding,
3. optimization via stochastic gradient descent (SGD) to minimize a carefully designed cost function.

Below, we provide mathematical details for each step.

5.3.4 *Step 1: Data Graph in the Input Space*

Consider a training dataset $\mathbf{X} \in \mathbb{R}^{n\times p}$, where n is the sample size and p is the dimensionality. We first construct a k-Nearest Neighbors (kNN) graph for this dataset, with $k = 15$ by default. More precisely, we construct a graph such that each node, represented by a data point $\mathbf{x}_i$, is connected to another point $\mathbf{x}_j$ if $\mathbf{x}_j$ is among the k closest neighbors of $\mathbf{x}_i$. Here the distance is arbitrary but we often use the Euclidean distance. For each data point $\mathbf{x}_i$, let $\mathcal{N}_i$ denote its set of k nearest neighbors.

We further refine this procedure as follows. For each pair of points $\mathbf{x}_i$ and its neighbor $\mathbf{x}_j \in \mathcal{N}_i$, we compute the probability $p_{j|i} \in (0, 1)$ that $\mathbf{x}_i$ selects $\mathbf{x}_j$ as a neighbor:

← Exercise 5.5.3

$$p_{j|i} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\| - \rho_i}{\sigma_i}\right), \tag{5.2}$$

where ρ_i is the distance from $\mathbf{x}_i$ to its nearest neighbour[3], and σ_i is a scaling factor computed such that the total similarity to neighbours is normalized:

$$\sum_{j=1}^{n} p_{j|i} = \log_2(k).$$

This is a directional similarity measure, but we symmetrize it to ensure symmetry between points:

← Exercise 5.5.4

$$p_{ij} = p_{j|i} + p_{i|j} - p_{j|i}p_{i|j}. \tag{5.3}$$

5.3.5 *Data Graph in the Embedding Space*

UMAP aims to create a lower-dimensional embedding $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n] \in \mathbb{R}^{n\times d}$ of the original data $\mathbf{X} \in \mathbb{R}^{n\times p}$, where $d \ll p$. The similarity between two points in this embedding space is measured as:

$$q_{ij} = \frac{1}{1 + a\|\mathbf{y}_i - \mathbf{y}_j\|^{2b}}, \tag{5.4}$$

where $a > 0$ and $b > 0$ are hyperparameters typically set to $a \approx 1.929$ and $b \approx 0.7915$.

[3] So that the closest point in $\mathcal{N}_i$ becomes a neighbor with proabbility 1

We still need to explain better how the embeddeding $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ is defined. The goal of UMAP is to make the data graph in the low-dimensional embedding space as similar as possible to the graph in the input space. This is achieved by minimizing the fuzzy cross-entropy cost:

← Exercise 5.5.5
← Exercise 5.5.6

$$c_1 = \sum_{i \neq j} \left(p_{ij} \log \frac{p_{ij}}{q_{ij}} + (1 - p_{ij}) \log \frac{1 - p_{ij}}{1 - q_{ij}} \right). \tag{5.5}$$

This cost function has two terms: the attractive force (first term), which brings neighbors closer in the embedding, and the repulsive force (second term), which pushes non-neighbor points apart. Minimizing this function ensures that points that are neighbors in the input space are also neighbors in the embedding space.

← Exercise 5.5.7

5.3.6 Step 2: Initialization using the spectral embedding

UMAP initializes the embedding $\mathbf{y}_1, \ldots, \mathbf{y}_n$ in a low-dimensional space using a Laplacian eigenmap discussed in Section 5.2. The spectral embedding step ensures that the initial positions of points in the low-dimensional space approximate the global structure of the graph, reducing the computational burden on the subsequent optimization phase.

5.3.7 Step 3: Training Algorithm

To refine the initial embedding, UMAP minimizes a cost function that balances attraction and repulsion between points in the embedding space. The cost function is:

$$c_1 = \sum_{(i,j) \in E} w_{ij} \log \left(\frac{1}{1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2} \right) + (1 - w_{ij}) \log \left(1 - \frac{1}{1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2} \right),$$

where $\|\boldsymbol{y}_i - \boldsymbol{y}_j\|$ is the Euclidean distance in the low-dimensional space.

The first term promotes the attraction of neighbors, while the second term ensures that non-neighbors are repelled, spreading out points in the embedding.

UMAP employs stochastic gradient descent (SGD) to optimize c_1. Unlike traditional gradient descent, which updates all points simultaneously, SGD: 1. Samples pairs of points $(\boldsymbol{y}_i, \boldsymbol{y}_j)$ randomly in mini-batches. 2. Updates $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ incrementally based on the gradient of c_1 with respect to their positions.

SGD is preferred over regular gradient descent due to:

1. Scalability: It handles large datasets efficiently by working on smaller subsets of data at a time.

2. Convergence Properties: The randomness in SGD helps avoid getting stuck in poor local minima, which is critical for high-dimensional, non-convex problems like UMAP.

5.4 *Autoencoders*

An autoencoder is a type of artificial neural network used to learn efficient codings of data. It is an unsupervised learning technique that learns a low-dimensional representation (encoding) for a dataset, typically for the purpose of dimensionality reduction. In the lecture slides, we will provide basic introduction to neural networks and autoencoders.

5.4.1 *Structure of Autoencoders*

Autoencoders consist of two main parts:

- **Encoder:** The encoder is a neural network that maps the input data $\mathbf{X} \in \mathbb{R}^{n \times d}$ to a lower-dimensional representation $\boldsymbol{Z} \in \mathbb{R}^{n \times p}$, where $p < d$.
- **Decoder:** The decoder is another neural network that reconstructs the original data from the encoded representation $\boldsymbol{Z}$.

The goal of training an autoencoder is to minimize the reconstruction error, typically measured as the mean squared error (MSE) between the input and the reconstructed output:

$$L(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2. \tag{5.6}$$

Here, $\hat{\mathbf{X}}$ is the output of the decoder, which attempts to reproduce the input data from the low-dimensional encoding $\boldsymbol{Z}$.

Unlike PCA, which assumes linearity, autoencoders can capture complex, nonlinear relationships in the data due to the use of nonlinear activation functions in the hidden layers. This makes autoencoders a flexible and powerful tool for dimensionality reduction, especially for high-dimensional and complex datasets, such as images or text data.

Autoencoders are widely used in tasks such as anomaly detection, data denoising, and generative modeling (e.g., variational autoencoders).

5.5 *Exercises*

Exercise 5.5.1. *Suppose $C \in \overline{\mathbb{S}}^n$. Show that $C_{ii} + C_{jj} - 2C_{ij} \geq 0$. Show that the distances $d_{ij}^2 = C_{ii} + C_{jj} - 2C_{ij}$ satisfy the triangle inequality.*

Exercise 5.5.2. *Show that the graph Laplacian matrix L in (5.1) is positive definite and it has at least one zero eigenvalue.*

Exercise 5.5.3. *Show that p_{j_i} defined in (5.2) lies in the interval $(0, 1)$.*

Exercise 5.5.4. *Show that p_{ij} defined in (5.3) is nonnegative.*

Exercise 5.5.5. *Let p, q be two probability distributions over some space $\mathcal{X}$. Define the Kullback-Leibler divergence $\mathrm{KL}(p, q) = \mathbb{E}_p \log \frac{p(X)}{q(X)}$. Show that $\mathrm{KL}(p, q) \geq 0$ and it is equal to zero if and only if p, q define the same distribution. Hint: Use the Jensen's inequality.*

Exercise 5.5.6. *Suppose $X_i \sim \mathrm{Bern}(p_i)$ and $Y_j \sim \mathrm{Bern}(q_j)$ are all independent. Show that the Kullback-Leibler divergence between the distribution p of $X = (X_1, \ldots, X_m)$ and the distribution q of $Y = (Y_1, \ldots, Y_m)$ is given by the formula*

$$\mathrm{KL}(p, q) \;=\; \sum_{i=1}^{m} \left(p_i \log \tfrac{p_i}{q_i} + (1 - p_i) \log \tfrac{1-p_i}{1-q_i} \right).$$

Exercise 5.5.7. *Show that the expression in (5.5) is always nonegative and it is equal to zero if and only if $p_{ij} = q_{ij}$ for all $i \neq j$.*

6
Canonical Correlation Analysis

In this chapter, we consider two datasets $\mathbf{X} \in \mathbb{R}^{n\times p}$ and $\mathbf{Y} \in \mathbb{R}^{n\times q}$ that represent observations of vectors $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}^q$, respectively. The goal is to find non-zero linear combinations $\boldsymbol{a}^\top X$ and $\boldsymbol{b}^\top Y$ with the largest possible correlation. Canonical Correlation Analysis (CCA) addresses this problem by finding pairs of linear combinations from each dataset that are maximally correlated.

Example 6.0.1. *Consider two datasets where X contains variables related to physical characteristics (height, weight, etc.), and Y contains variables related to exercise performance (running speed, weightlifting capacity, etc.). CCA can identify combinations of variables in X that are most strongly associated with combinations of variables in Y, such as finding that a linear combination of height and weight is strongly correlated with a linear combination of running speed and weightlifting capacity.*

6.1 Population CCA

6.1.1 Principal Correlation Vectors

Let $\Sigma_{11} = \mathrm{var}(X)$, $\Sigma_{22} = \mathrm{var}(Y)$, and $\Sigma_{12} = \mathrm{cov}(X, Y)$. The correlation between $\boldsymbol{a}^\top X$ and $\boldsymbol{b}^\top Y$ is given by

$$\rho(\boldsymbol{a}, \boldsymbol{b}) \; := \; \frac{\boldsymbol{a}^\top \Sigma_{12} \boldsymbol{b}}{\sqrt{\boldsymbol{a}^\top \Sigma_{11} \boldsymbol{a}\, \boldsymbol{b}^\top \Sigma_{22} \boldsymbol{b}}}.$$

This expression can be challenging to optimize directly. To simplify, we reparametrize the problem using $\boldsymbol{\alpha} = \Sigma_{11}^{1/2}\boldsymbol{a}$ and $\boldsymbol{\beta} = \Sigma_{22}^{1/2}\boldsymbol{b}$. This transforms the correlation expression into

$$\rho(\boldsymbol{\alpha}, \boldsymbol{\beta}) \; = \; \frac{\boldsymbol{\alpha}^\top \Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1/2} \boldsymbol{\beta}}{\|\boldsymbol{\alpha}\| \|\boldsymbol{\beta}\|} \; = \; \frac{\boldsymbol{\alpha}^\top M \boldsymbol{\beta}}{\|\boldsymbol{\alpha}\| \|\boldsymbol{\beta}\|} = \left(\frac{\boldsymbol{\alpha}}{\|\boldsymbol{\alpha}\|}\right)^\top M \left(\frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|}\right),$$

where we define the matrix

$$M \; = \; \Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1/2}.$$

To maximize the correlation, we now solve the following optimization problem:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \boldsymbol{\alpha}^\top M \boldsymbol{\beta} \quad \text{subject to} \quad \boldsymbol{\alpha}^\top \boldsymbol{\alpha} = \boldsymbol{\beta}^\top \boldsymbol{\beta} = 1.$$

The Lagrangian for this constrained optimization is given by:

$$L \;=\; 2\boldsymbol{\alpha}^\top M \boldsymbol{\beta} - \lambda(\boldsymbol{\alpha}^\top \boldsymbol{\alpha} - 1) - \lambda'(\boldsymbol{\beta}^\top \boldsymbol{\beta} - 1).$$

At a stationary point, we get the following system of equations:

$$M\boldsymbol{\beta} = \lambda \boldsymbol{\alpha} \quad \text{and} \quad M^\top \boldsymbol{\alpha} = \lambda' \boldsymbol{\beta}.$$

By computing $\boldsymbol{\alpha}^\top M \boldsymbol{\beta}$ from both expressions, we conclude that $\lambda = \lambda'$. From this, it follows that

$$M^\top M \boldsymbol{\beta} = \lambda^2 \boldsymbol{\beta} \quad \text{and} \quad M M^\top \boldsymbol{\alpha} = \lambda^2 \boldsymbol{\alpha}.$$

Thus, the optimal $\boldsymbol{\alpha}$ is the eigenvector of $MM^\top$ corresponding to the largest eigenvalue, and the optimal $\boldsymbol{\beta}$ is the eigenvector of $M^\top M$ corresponding to the same eigenvalue.

Remark 6.1.1. *If we decompose M using the Singular Value Decomposition (SVD), $M = U\Lambda V^\top$, where U and V are orthogonal matrices and Λ is diagonal, then $MM^\top = U\Lambda^2 U^\top$ and $M^\top M = V\Lambda^2 V^\top$. The eigenvectors of $MM^\top$ are the left singular vectors of M, and the eigenvectors of $M^\top M$ are the right singular vectors of M. The singular values are the canonical correlations.*

6.1.2 First r Principal Correlation Vectors

As in PCA, we can recursively find more canonical correlation directions after obtaining the first one. Let $\boldsymbol{a}_1, \boldsymbol{b}_1$ be the first pair of canonical correlation vectors. To find the next pair $(\boldsymbol{a}_2, \boldsymbol{b}_2)$, we solve the following constrained problem:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \boldsymbol{\alpha}^\top M \boldsymbol{\beta} \quad \text{subject to} \quad \|\boldsymbol{\alpha}\| = \|\boldsymbol{\beta}\| = 1 \quad \text{and} \quad \boldsymbol{\alpha}^\top \boldsymbol{\alpha}_1 = \boldsymbol{\beta}^\top \boldsymbol{\beta}_1 = 0.$$

The solution corresponds to the second largest singular values and their corresponding left and right singular vectors. Repeating this process provides the subsequent pairs $\boldsymbol{a}_3, \boldsymbol{b}_3, \ldots, \boldsymbol{a}_r, \boldsymbol{b}_r$.

6.1.3 Correlations Between Canonical Correlation Variables

Let $\eta_1 = \boldsymbol{a}_1^\top X$ and $\phi_1 = \boldsymbol{b}_1^\top Y$ be the first pair of canonical correlation variables. The covariance between η_1 and ϕ_1 is given by

$$\operatorname{cov}(\eta_1, \phi_1) = \boldsymbol{a}_1^\top \Sigma_{12} \boldsymbol{b}_1 = \boldsymbol{\alpha}_1^\top M \boldsymbol{\beta}_1 = \lambda_1,$$

where λ_1 is the largest canonical correlation.

Theorem 6.1.2. *Let $\eta_i = \boldsymbol{a}_i^\top X$ and $\phi_i = \boldsymbol{b}_i^\top Y$ for $i = 1, \dots, r$ be the first r canonical correlation variables. The covariance matrix of $\boldsymbol{\eta} = (\eta_1, \dots, \eta_r)$ and $\boldsymbol{\phi} = (\phi_1, \dots, \phi_r)$ is block-diagonal:*

$$\operatorname{var}\left(\begin{bmatrix}\boldsymbol{\eta} \\ \boldsymbol{\phi}\end{bmatrix}\right) = \begin{bmatrix} I_r & \Lambda_r \\ \Lambda_r & I_r \end{bmatrix},$$

where $\Lambda_r = diag(\lambda_1, \dots, \lambda_r)$ contains the first r canonical correlations. Hence, the variables $\eta_1, \dots, \eta_r$ and $\phi_1, \dots, \phi_r$ are uncorrelated across different pairs but correlated within each pair.

6.2 Sample CCA

In practice, we do not know the population covariance matrices $\Sigma_{11}, \Sigma_{22}, \Sigma_{12}$. Instead, we estimate them from data. Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ and $\mathbf{Y} \in \mathbb{R}^{n \times q}$ represent the two datasets, and let S_{11}, S_{22}, S_{12} denote the sample covariance matrices of X and Y[1]. As long as S_{11} and S_{22} are invertible, we can apply the same procedure as for the population case, using the sample estimates[2].

The sample CCA maximizes the sample correlation between linear combinations of X and Y:

$$\widehat{\rho}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a}^\top S_{12} \boldsymbol{b}}{\sqrt{\boldsymbol{a}^\top S_{11} \boldsymbol{a}\, \boldsymbol{b}^\top S_{22} \boldsymbol{b}}}.$$

The sample canonical correlation vectors can be found by solving the generalized eigenvalue problem for the matrix

$$\widehat{M} = S_{11}^{-1/2} S_{12} S_{22}^{-1/2}.$$

6.3 Practical Considerations and Applications

6.3.1 Data Preprocessing

Before applying CCA, both datasets **X** and **Y** should be centered to ensure that the covariance matrices reflect the relationships between the variables, not the differences in their means. If the variables have different scales, it may also be necessary to standardize them to have unit variance.

6.3.2 Applications

CCA is widely used in fields where two sets of related data are collected, such as:

- **Neuroscience**: Relating brain activity to behavior or stimuli data, CCA can find the strongest correlations between neural signals (e.g., EEG or fMRI data) and observed behaviors.

[1] Note that in this setting we need joint observations of X and Y. If such joint observations are not available (e.g. these variables come from different domains and so cannot be observed jointly) we need to employ other techniques that assume some latent joint structure.

[2] Recall from Section 4.6 that there may be better ways to estimate the underlying covariance matrix. This is especially important in high-dimensional situations.

- **Genomics**: In genetics, one dataset might contain gene expression levels, and the other might contain phenotypic traits. CCA can help identify which linear combinations of genes are most strongly associated with combinations of traits.

- **Marketing**: CCA can be used to relate consumer survey data (e.g., preferences, opinions) to purchasing behavior data. By identifying canonical correlations, businesses can understand how different groups of opinions are linked to purchasing patterns.

- **Multimodal Data Fusion**: In cases where multiple types of data (e.g., image data and text data) are collected for the same observations, CCA can be used to find the strongest relationships between these different types of data, helping in the integration and interpretation of multimodal datasets.

- **Economics and Finance**: CCA is used to explore the relationships between different economic indicators (such as inflation, unemployment) and financial metrics (such as stock prices, interest rates). This can help in understanding how economic factors jointly influence financial markets.

6.4 Interpretation and Visualization of CCA Results

After obtaining the canonical correlation vectors, it is important to interpret the results and visualize the relationships between the canonical variables. Some common approaches to interpreting and visualizing CCA results include:

6.4.1 Biplots

Biplots can be used to visualize the relationships between the original variables and the canonical variables. In a CCA biplot, the canonical variables $\eta_1, \eta_2, \dots$ are plotted against each other, and the original variables are represented as vectors in the plot. The length and direction of each vector indicate the strength and direction of the relationship between the original variable and the canonical variables.

6.4.2 Canonical Correlation Scores

The canonical correlation scores $\eta_1, \eta_2, \dots$ for X and $\phi_1, \phi_2, \dots$ for Y can be plotted to visualize how the observations are related in the reduced canonical space. Observations that are close to each other in this space are highly correlated with respect to the canonical variables.

6.4.3 *Heatmaps of Canonical Correlations*

Heatmaps can be used to visualize the correlation structure between the canonical variables η_i and ϕ_j. The canonical correlation matrix $\Lambda_r = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r)$ can be plotted as a heatmap, with darker colors representing stronger correlations.

6.5 *Example: Real Data Analysis in R*

To demonstrate Canonical Correlation Analysis, we will use the `mtcars` dataset, which contains various automobile attributes. We will split the dataset into two groups of variables:

- Group 1 (X); Variables related to performance: `mpg` (Miles per gallon), `hp` (Gross horsepower), `drat` (Rear axle ratio).
- Group 2 (Y); Variables related to design: `wt` (Weight (1000 lbs), `qsec` (1/4 mile time), `gear` (Number of forward gears).

Our goal is to identify pairs of canonical variables that capture the strongest relationships between these two groups of variables.

6.5.1 *Step-by-Step Analysis*

Step 1: Load and Preprocess Data

We load the `mtcars` dataset and split it into two groups of variables. All variables are standardized to ensure comparability.

```
#Load necessary library and data
library(CCA)
data(mtcars)

#Group 1: Performance-related variables
X <- mtcars[, c('mpg', 'hp', 'drat')]

#Group 2: Design-related variables
Y <- mtcars[, c('wt', 'qsec', 'gear')]

#Standardize the datasets
X <- scale(X)
Y <- scale(Y)
```

Step 2: Perform Canonical Correlation Analysis

We use the `cancor` function in R to compute the canonical correlations and the corresponding canonical variables.

```
# Perform CCA
cca_result <- cancor(X, Y)

#Print the canonical correlations
print(cca_result$cor)
```

The canonical correlations are the correlations between the canonical variables derived from the two datasets. For this dataset, the values are: `0.94, 0.75, 0.22`, indicating a strong relationship between the first pair of canonical variables.

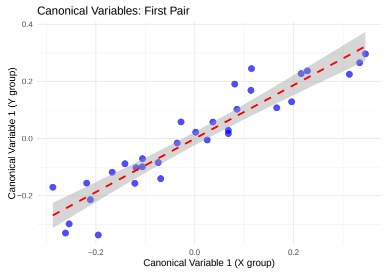
Step 3: Extract Canonical Variables

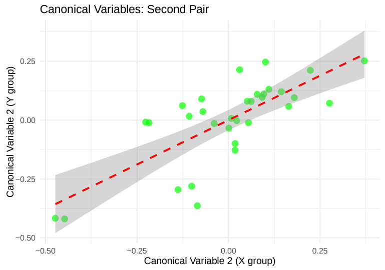
The canonical variables for X and Y can be computed using the canonical weights.

```r
# Extract canonical weights
x_weights <- cca_result$xcoef
y_weights <- cca_result$ycoef

# Create a data frame for weights
weights <- data.frame(
  Variable = c(colnames(X), colnames(Y)),
  Canonical1 = c(x_weights[, 1], y_weights[, 1]),
  Canonical2 = c(x_weights[, 2], y_weights[, 2]),
  Canonical3 = c(x_weights[, 3], y_weights[, 3]))

# Compute canonical scores
x_scores <- as.matrix(X) %*% x_weights
y_scores <- as.matrix(Y) %*% y_weights

# Create a data frame for visualization
scores_df <- data.frame(
  Canonical_X1 = x_scores[, 1],
  Canonical_Y1 = y_scores[, 1],
  Canonical_X2 = x_scores[, 2],
  Canonical_Y2 = y_scores[, 2])
```

Now we can plot scatter plots for the first two canonical variables in both groups.

```r
library(ggplot2)

# Scatterplot for the first pair of canonical variables
ggplot(scores_df, aes(x = Canonical_X1, y = Canonical_Y1)) +
  geom_point(color = "blue", size = 3, alpha = 0.7) +
  geom_smooth(method = "lm", color = "red", linetype = "dashed") +
  labs(
    title = "Canonical Variables: First Pair",
    x = "Canonical Variable 1 (X group)",
    y = "Canonical Variable 1 (Y group)") + theme_minimal()

# Scatterplot for the second pair of canonical variables
ggplot(scores_df, aes(x = Canonical_X2, y = Canonical_Y2)) +
  geom_point(color = "green", size = 3, alpha = 0.7) +
  geom_smooth(method = "lm", color = "red", linetype = "dashed") +
  labs(
    title = "Canonical Variables: Second Pair",
    x = "Canonical Variable 2 (X group)",
```





The plots above shows the relationship between the first pair and the second pair of canonical variables. A strong linear relationship is evident, reflecting the high canonical correlation of `0.94` and `0.75`.

7
Graphical Models

This chapter will be expanded but the outline will be roughly preserved. Exercises will be added.

If you are interested in more details of this beautiful theory, see [1].

7.1 Introduction

Graphical models are a powerful framework for representing and analyzing the structure of dependencies in multivariate data. They combine graph theory and probability to model the conditional independence relationships among a set of random variables. By encoding these relationships in the form of a graph, graphical models provide an intuitive and interpretable representation of complex systems, making them invaluable tools in modern statistics and data science.

Let $X = (X_1, \ldots, X_m)$ be the random vector of interest. A graphical model for X consists of a graph $G = (V, E)$, where $V = \{1, \ldots, m\}$ represents a set of nodes corresponding to random variables, and E represents edges. Here absence of the edge ij indicates a form of conditional independence of X_i and X_j given a subset of variables in the graph. This explicit representation of dependencies facilitates tasks such as prediction, hypothesis testing, and model selection.

Graphical models are ubiquitous in modern statistics, with applications spanning various disciplines:

- **Genetics and Computational Biology**: Graphical models are used to infer gene regulatory networks, model genetic associations, and analyze biological pathways.
- **Social Networks**: They facilitate the understanding of relationships and influences in social structures, enabling the study of community detection, opinion dynamics, and information flow.
- **Finance and Economics**: Graphical models are employed to model dependencies among financial indicators, understand risk propagation, and analyze market behavior.

[1] Steffen L. Lauritzen. *Graphical Models*, volume 17 of *Oxford Statistical Science Series*. Clarendon Press, Oxford, 1996. ISBN 978-0-19-852219-5

- **Machine Learning**: Probabilistic graphical models underpin several machine learning methods, such as hidden Markov models, topic modeling, and deep probabilistic models.

Graphical models can be broadly categorized into three main types:

- **Undirected Graphical Models**: These models, also known as Markov random fields, represent symmetric relationships.
- **Directed Graphical Models**: These models, commonly referred to as Bayesian networks, capture asymmetric dependencies and causal relationships between variables.
- **Mixed Graphs**: These models incorporate both directed and undirected edges, enabling the representation of more complex dependency structures.

This chapter explores the theoretical foundation and practical applications of graphical models. This topic is very vast so, for simplicity, we focus on the special case of undirected graphical models.

7.2 Graphs and Conditional Independence

A graph $G = (V, E)$ consists of a set of nodes V and edges E. Nodes represent random variables, and edges represent direct dependencies or associations between the variables. A simple example is given in Figure 7.2.

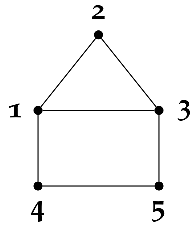


Figure 7.1: A simple graph with five nodes and six edges.

Graphs can be represented in multiple ways. An **adjacency matrix** is A square matrix A where $A_{ij} = 1$ if there is an edge between nodes i and j, and $A_{ij} = 0$ otherwise. For example, for the graph in Figure 7.2, the adjacency matrix is

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Another way to represent a graph is by listing for each node the set of all its neighbors. In our running example: $\mathcal{N}_1 = \{2, 3, 4\}$, $\mathcal{N}_2 = \{1, 3\}$, $\mathcal{N}_3 = \{1, 2, 5\}$, $\mathcal{N}_4 = \{1, 5\}$, and $\mathcal{N}_5 = \{3, 4\}$.

A path is a sequence of nodes connected by edges. A cycle is a path that starts and ends at the same node. A **clique** is a subset of nodes in an undirected graph where every pair of nodes is connected by an edge. A maximal clique is a clique that is maximal with respect to inclusion (no other clique strictly containing it). In the running example the maximal cliques are $\{1, 2, 3\}$, $\{1, 4\}$, $\{3, 5\}$, $\{4, 5\}$.

Separation on graphs is a crucial concept for us. We say that two subsets $A, B \subseteq V$ are **separated** from each other by a subset $C \subseteq V$ is every path between any node $i \in A$ and $j \in B$ necessarily contains a node in C; we write $A \perp_G B|C$. For example, in Figure 7.2, $A = \{2\}$ is separated from $B = \{4, 5\}$ by $C = \{1, 3\}$. However, $C' = \{1\}$ does not separate A from B.

7.2.1 *Graph factorization*

An important defining concept in graphical models is that of **factorization with respect to** G. Let G be an undirected graph with nodes $\{1, ..., m\}$ and maximal cliques $C_1, \ldots, C_k$. For any $C \subseteq \{1, \ldots, m\}$, $\mathbf{x}_C$ denotes a subvector of $\mathbf{x}$, $\mathbf{x}_C = (x_i)_{i \in C}$.

We say that density $f(x)$ of X factorizes according to G if

$$f(\mathbf{x}) \;=\; \prod_{C \in \mathcal{C}} \phi_C(\mathbf{x}_C) \quad \text{for all } \mathbf{x} \in \mathcal{X}, \tag{7.1}$$

where $\phi_C(\mathbf{x}_C) \geq 0$.[2] For the graph in Figure 7.2,

$$f(x_1, x_2, x_3, x_4, x_5) = \phi_{123}(x_1, x_2, x_3)\phi_{14}(x_1, x_4)\phi_{35}(x_3, x_5)\phi_{45}(x_4, x_5).$$

We define the graphical model of G as the set of all distributions on $\mathcal{X}$ that factorize with respect to G.

7.2.2 *Conditional Independence and Graph Separation*

Conditional independence and separation are the core ideas linking graphs to probabilistic models. They formalize the notion that certain subsets of variables are independent, given others.

In undirected graphs, conditional independence is determined by the concept of graphical separation called **Global Markov Property**: If a set of nodes C separates A from B in the graph, then $X_A \perp\!\!\!\perp X_B | X_C$. In the graph in Figure 7.2, $X_2 \perp\!\!\!\perp \{X_4, X_5\} | \{X_1, X_2\}$.

The most fundamental result of graphical models that links Global Markov property with factorization is the Hammersley-Clifford theorem.

Theorem 7.2.1 (Hammersley-Clifford theorem). *Let $f > 0$ be a dentity function for $X = (X_1, ..., X_m)$. Then the following are equivalent:*

(F) f factorizes according to G.

(G) $X_A \perp\!\!\!\perp X_B | X_C$ whenever C separates A and B in G.

(P) $X_i \perp\!\!\!\perp X_j | X_{V \setminus \{i,j\}}$ whenever i, j not connected by an edge in G.

By the Hammersley-Clifford theorem graphical models could be equivalently defined by specific conditional independences.[3]

[2] This is a notion of simplicity, which states that the distribution can be glued in a specific way from simpler pieces.

[3] For simplicity, in this lecture we work only with positive distributions for which the Hammersley-Clifford theorem applies.

In what follows, we discuss two most important versions of undirected graphical models. One for the Gaussian distributions and the other for finite discrete distributions.

7.3 Gaussian Graphical Models

Gaussian Graphical Models (GGMs) describe the conditional independence structure of multivariate normal distributions. They are widely used in statistics, machine learning, and computational biology due to their interpretability and mathematical tractability.

7.3.1 Definition and Properties

We already learned about the multivariate normal distributions and many of its properties will become essential here in this chapter. In particular, recall Proposition 2.3.2, which states that $K_{ij} = 0$ if and only if $X_i \perp\!\!\!\perp X_j | X_{V \setminus \{i,j\}}$. By the Hammersley-Clifford theorem, this equivalent to the fact that the density factorizes with respect to the graph that represents the support of the inverse covariance matrix $K = \Sigma^{-1}$, namely the graph G given by

$$(i,j) \notin E \qquad \Longleftrightarrow \qquad K_{ij} = 0.$$

Moreover, by the Global Markov Property we get a way of reading from the graph other more complicated conditional independences. Slightly abusing notation we write

$$M(G) \;=\; \{\Sigma \in \mathbb{S}^m : \; \Sigma^{-1}_{ij} = 0 \text{ if } ij \notin E\}.$$

7.3.2 Estimation with fixed G

Given n i.i.d. samples $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^m$ consider the sample covariance matrix S_n as defined in (1.4). Throughout we assume that the sample comes from $N(\mu, \Sigma)$, $\Sigma \in M(G)$. A natural way to estimate parameters in a Gaussian graphical model $M(G)$ is by maximizing the log-likelihood function. This results in a consistent and asymptotically normal estimator by standard results.

Note that the mean μ is unrestricted in this model and so, by the same argument as in Section 2.4.1, the maximum likelihood estimator of μ is equal to the sample average; $\hat{\mu} = \bar{\mathbf{x}}_n$. Thus, expressed in terms of $K = \Sigma^{-1}$, the log-likelihood function takes a simpler form (as in (2.6)):

$$\ell(\bar{\mathbf{x}}, K) \;=\; \text{const} + \tfrac{n}{2}(\log\det(K) - \text{tr}(S_n K)).$$

Denote $f(K) = -\log\det(K) + \text{tr}(S_n K)$ so that maximizing $\ell(\bar{\mathbf{x}}, K)$ is equivalent to minimizing $f(K)$. We noted already in Section 2.4.1 that

f is a strictly convex function of K. Define the linear subspace

$$\mathcal{V}(G) \;=\; \{K \in \mathbb{S}^m :\; K_{ij} = 0 \text{ if } ij \notin E\}.$$

With this notation optimizing the log-likelihood for a Gaussian graphical model results in equivalent to minimizing $f(K)$ a convex optimization problem; $\min f(K)$ subject $K \in \mathcal{V}(G) \cap \mathbb{S}^m_+$.[4] This problem does not have a closed form solution in general[5] so we solve it using numerical methods.

Some of the most efficient methods use the dual formulation of the maximum likelihood problem. We formulate this only sketching the proof.

Proposition 7.3.1. *The maximum likelihood estimator $\widehat{\Sigma}$, if exists, is the unique maximizer of* $\log\det\Sigma$ *subject to $\Sigma_{ii} \;=\; (S_n)_{ii}$ for all $i \;\in\; V$ and $\Sigma_{ij} = (S_n)_{ij}$ for all $ij \in E$.*

The idea of the following proof sketch applies to a wide range of related problems in multivariate statistics. Thus, it is beneficial to follow it closely.

Proof Sketch. We can extend f to whole $\mathbb{S}^m$ by setting $f(K) \;=\; +\infty$ if $K \;\notin\; \mathbb{S}^m_+$. Minimizing this extended f over $\mathbb{S}^m$ is equivalent to minimizing f over $\mathbb{S}^m_+$. Consider the orthogonal complement [6]

$$\mathcal{W}(G) \;:=\; \mathcal{V}(G)^{\perp} \;=\; \{K \in \mathbb{S}^m : K_{ii} = 0 \text{ for all } i \text{ and } K_{ij} = 0 \text{ for } ij \in E\}.$$

Observe that

$$\sup_{\Lambda \in \mathcal{W}(G)} f(K) + \operatorname{tr}(\Lambda K) \;=\; \begin{cases} f(K) & \text{if } K \in \mathcal{V}(G), \\ +\infty & \text{otherwise.} \end{cases}$$

Indeed, if $K \in \mathcal{V}(G)$ then $\operatorname{tr}(\Lambda K) = 0$ and so $f(K) + \operatorname{tr}(\Lambda K) = f(K)$. On the other hand, if $K \;\notin\; \mathcal{V}(G)$, there exists $\Lambda \;\in\; \mathcal{W}(G)$ such that $\operatorname{tr}(\Lambda K) \;>\; 0$. Taking $t\Lambda$ with $t \;\to\; \infty$ shows that $\sup_{\Lambda \in \mathcal{W}(G)} f(K) + \operatorname{tr}(\Lambda K) = +\infty$. This all implies that

$$\inf_{K \in \mathbb{S}^m} f(K) \;=\; \inf_{K \in \mathbb{S}^m} \sup_{\Lambda \in \mathcal{W}(G)} f(K) + \operatorname{tr}(\Lambda K).$$

By the minimax theorem from convex analysis[7], which we are not going to prove here, we have

$$\inf_{K \in \mathbb{S}^m} \sup_{\Lambda \in \mathcal{W}(G)} f(K) + \operatorname{tr}(\Lambda K) \;=\; \sup_{\Lambda \in \mathcal{W}(G)} \inf_{K \in \mathbb{S}^m} f(K) + \operatorname{tr}(\Lambda K).$$

The latter expression is easy to handle since

$$\inf_{K \in \mathbb{S}^m} f(K) + \operatorname{tr}(\Lambda K) \;=\; \inf_{K \in \mathbb{S}^m} -\log\det(K) + \operatorname{tr}((S_n + \Lambda)K) \;=\; -\log\det((S_n + \Lambda)^{-1}) + m,$$

[4] Intersection of two convex set is convex. Moreover, $\mathbb{S}^m_+$ is convex directly by definition.

[5] It does if and only if G is **decomposable**. See Lauritzen 1996 for details.

[6] If a finite dimensional linear subspace is defined by vanishing of some of the coordinates then its complement is defined by vanishing of the remaining coordinates.

[7] This is the only informal step of this proof sketch.

where the last equality comes from the fact that minimizing the middle expression is equivalent to maximizing the (unrestricted) Gaussian log-likelihood with sample covariance matrix $S_n + \Lambda$. We get that

$$\inf_{K \in \mathbb{S}^m} f(K) \;=\; \sup_{\Lambda \in \mathcal{W}(G)} \log\det(S_n + \Lambda) + m,$$

which gives the desired conclusion. □

7.3.3 Matrix completion

Proposition 7.3.1 implies that the MLE $\widehat{\Sigma}$ has the same diagonal as the sample covariance matrix S_n. The same applies to all entries corresponding to the edges of G. Thus, it remains to fill-up the remaining entries so that the determinant is maximal possible.

Note that by fixing the i-th row/column, we can write

$$\det(\Sigma) \;=\; \det(\Sigma_{\setminus i,\setminus i})(\Sigma_{ii} - \Sigma_{i,\setminus i}\Sigma^{-1}_{\setminus i,\setminus i}\Sigma_{\setminus i,i}).$$

Suppose we want to optimize the determinant with respect to $y := \Sigma_{\setminus i,i}$ only. Since $\det(\Sigma_{\setminus i,\setminus i}) > 0$, equivalently, we minimize $y^\top \Sigma^{-1}_{\setminus i,\setminus i} y$. Since $\Sigma_{\setminus i,\setminus i}$ is positive definite[8], the global optimum of this quadratic form is $y = \mathbf{0}$. But in our case $y = \Sigma_{\setminus i,i}$ must satisfy $\Sigma_{ji} = (S_n)_{ji}$ for $j - i \in E$. Nevertheless, this is a simple quadratic problem with linear constraints for which there are very efficient algorithms and implementations.

We can now imagine an iterative procedure, where we update Σ row by row. This is an example of block coordinate descent. Although we are not going to prove this formally, since the original function $\log\det\Sigma$ is strictly concave, this is guaranteed to converge to the global optimum.

7.3.4 Challenges in High-Dimensional Settings

As we noted, the maximum likelihood is a consistent estimator. However, we noted already in Section 4.6 that if the underlying dimension is not negligible with respect to n the classical asymptotic results may not be very accurate. For this reason, in modern applications we use estimators that rely on some form of regularization. We will discuss in Section 7.3.6 a form of regularization that allows us to learn the underlying graph G.

7.3.5 Structure learning (aka model selection)

In practice, the graph G is rarely known. A natural approach to test G_0 against its supergraph G is to use the chi-squared test using the

[8] Every principal submatrix of a positive definite matrix is positive definite.

likelihood ratio statistics

$$2\log \mathrm{LR} \;=\; 2(\log(\text{maximized likelihood for } M(G)) - \log(\text{maximized likelihood for } M(G_0))).$$

Proposition 7.3.2. *If G_0 is a subgraph of G and G has ν edges more then*

$$2\log \mathrm{LR} \xrightarrow{d} \chi^2_d.$$

Alternatively, Akaike's Information Criterion (AIC) can be used to choose a model. The AIC is defined by

$$\text{AIC} \;=\; -2\log(\text{maximized likelihood}) + 2\times(\text{number of parameters fitted})$$

and the model with the smallest AIC is the selected model.

For larger m we may apply some stepwise procedure. For example, starting from the full graph we consider $\binom{m}{2}$ models obtained by removing one of the edges. When the optimal removal is picked, we compare it with the complete model. If its AIC is smaller, we remove the edge and check the resulting $\binom{m}{2}-1$ submodels. We stop when the bigger model has smaller AIC then all the proposed smaller models. This procedure has, of course, no guarantee of getting the globally optimal model.

7.3.6 Graphical LASSO

In high-dimensional scenarios, stepwise procedures described above are impractical and we often use other approaches based on regularization techniques.

The graphical LASSO estimates sparse precision matrices by solving the optimization problem:

$$\widehat{K} = \arg\max_{K\in\mathbb{S}^m_+} \left\{\log\det(K) - \operatorname{tr}(S_n K) - \lambda\|K\|_{1,\text{off}}\right\},$$

where $\lambda > 0$ is a fix penalty parameter and

$$\|K\|_{1,\text{off}} \;:=\; \sum_{i\neq j}|K_{ij}|.$$

In other words the graphical LASSO estimator is obtained by penalizing the log-likelihood function with a term that penalizes the off-diagonal entries to be too large effectively pushing some of them to be zero. This is in the same spirit as the LASSO regression, hence the name.

An efficient algorithm to find $\widehat{K}$ follows exactly the same lines as the block coordinate descent algorithm for maximizing the log-likelihood function. First we formulate a variational form of $\|K\|_{1,\text{off}}$. Noting that $|x| = \sup_{-1\leq y\leq 1} xy$ we get

$$\lambda\|K\|_{1,\text{off}} \;=\; \lambda\sum_{i\neq j} sup_{-1\leq\Lambda_{ij}\leq 1}\Lambda_{ij}K_{ij} \;=\; \sup_{\Lambda}\operatorname{tr}(\Lambda K),$$

where the supremum is taken over all $\Lambda \in \mathbb{S}^m$ with zeros on the diagonal and off-diagonal entries in $[-\lambda, \lambda]$.

This allows to rewrite the main function as

$$\sup_{\Lambda} \log\det(K) - \mathrm{tr}((S_n + \Lambda)K).$$

Proceeding as in the proof of Proposition 7.3.1, we get that the dual problem is to maximize $\log\det\Sigma$ subject to $\Sigma_{ii} = (S_n)_{ii}$ for $i = 1, \dots, m$ and $|\Sigma_{ij} - (S_n)_{ij}| \leq \lambda$.

We can again solve this problem using the same block coordinate descent approach.

7.3.7 *Practical Example and R Code*

Here is an example of applying the graphical LASSO in R using the `glasso` package:

```r
library(glasso)

# Simulated data
set.seed(123)
n <- 100
p <- 10
X <- matrix(rnorm(n * p), nrow = n, ncol = p)

# Sample covariance matrix
S <- cov(X)

# Apply graphical LASSO
lambda <- 0.1
fit <- glasso(S, rho = lambda)

# Extract precision matrix
precision_matrix <- fit$wi
print(precision_matrix)
```

7.4 *Log-linear Graphical Models*

In these lecture notes we largely ignored discrete variables. We try to partly fix it in this section by discussing discrete graphical models.

Log-linear graphical models describe the dependence structure of discrete random variables using a graph-based factorization of the joint probability distribution. They are widely used in statistics, machine learning, and social network analysis for modeling categorical data.

7.4.1 *Notation*

Let $X = (X_1, \dots, X_m)$ be a vector of discrete random variables, where each X_i takes values in a finite set $\mathcal{X}_i$. Without loss of generality let

$\mathcal{X} = \{0, 1, \dots, r_i - 1\}$. The joint observations of X is often represented by a **contingency table**, which records the frequencies of all possible configurations of X. This is a $r_1 \times \cdots \times r_m$ table with $(x_1, \dots, x_m)$-th entry $n_{x_1 \cdots x_m}$.

For example, if $m = 2$ and X_1, X_2 are binary variables with values in $\{0, 1\}$, the contingency table might look like this:

	$X_2 = 0$	$X_2 = 1$
$X_1 = 0$	n_{00}	n_{01}
$X_1 = 1$	n_{10}	n_{11}

Sufficient statistics for log-linear models are derived from these counts. Let $\boldsymbol{n} = (n_x)_{x \in \mathcal{X}}$ be the vector of observed cell counts.

7.4.2 Discrete graphical models through factorization

A log-linear model can be represented using an undirected graph $G = (V, E)$, where each node in V corresponds to a variable X_i, and an edge between X_i and X_j indicates a direct interaction between them.

Let $\mathcal{C}$ be the set of (maximal) cliques of the graph G. For each $C \in \mathcal{C}$ consider the set $\mathcal{X}_C = \prod_{i \in C} \mathcal{X}_i$ and let $\theta_C : \mathcal{X}_C \to \mathbb{R}$.[9] The joint probability mass function $f(x) = P(X = x)$ of a log-linear model is expressed as:

$$f(x) \;=\; \frac{1}{Z} \exp\left(\sum_{C \in \mathcal{C}} \theta_C(x_C)\right),$$

where $Z = \sum_x \exp\left(\sum_{C \in \mathcal{C}} \theta_C(x_C)\right)$ is the normalizing constant.

The model assumes that interactions between variables occur only within cliques. It is not hard to see that this definition is equivalent with f factorizing with respect to G, as defined in (7.1). By the Hammersley-Clifford, the corresponding distributions satisfy the global and pairwise Markov properties.

Remark 7.4.1. *Log-linear models belong to the exponential family of distributions, where the natural parameters are θ_C, and the sufficient statistics are $\phi_C(x)$. This connection provides a theoretical foundation for model fitting and inference. Not to overload notation, we will not explain this link carefully.*

7.4.3 Fitting Log-linear Models: Iterative Proportional Fitting (IPF)

The parameters θ_C can be estimated using iterative proportional fitting (IPF), which adjusts the cell counts in the contingency table iteratively to match the sufficient statistics of the model. The algorithm ensures that the fitted probabilities $\hat{P}(X = x)$ satisfy the constraints implied by the model.

[9] Since $d_C := |\mathcal{X}_C|$ is finite, we can represent θ_C as a vector in $\mathbb{R}^{d_C}$.

7.4.4 *The Ising Model*

The dimensionality of the log-linear model does not scale well with the number of the variables and their number of values. For this reason it is important to consider simpler models for discrete data.

Definition of the Ising Model The Ising model is a special case of a log-linear graphical model for binary variables $X_i \in \{-1, 1\}$. The joint distribution is:

$$P(X = x) \propto \exp\left(\sum_{i \in V} \theta_i x_i + \sum_{(i,j) \in E} \theta_{ij} x_i x_j\right),$$

where θ_i are node potentials and θ_{ij} are edge potentials.

Applications The Ising model is widely used in:

- **Network Analysis**: Modeling social networks where nodes represent individuals, and edges represent interactions.

- **Spatial Statistics**: Modeling dependencies in spatial data, such as image segmentation.

Connections to Log-linear Graphical Models The Ising model is a log-linear model where the cliques are individual nodes and edges in the graph. The sufficient statistics are x_i for nodes and $x_i x_j$ for edges.

Statistical Estimation and Inference Parameters θ_i and θ_{ij} can be estimated using methods such as:

- **Maximum Likelihood Estimation (MLE)**: Solve optimization problems involving the log-likelihood function.

- **Pseudo-Likelihood Methods**: Approximate the likelihood to reduce computational complexity.

Inference tasks, such as computing marginal probabilities, are computationally challenging due to the normalization constant Z. Approximation methods like Gibbs sampling and variational inference are often employed.

8
Factor analysis

9
Introduction to Tensor Methods