POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Robust model-based clustering for high-dimensional data via covariance matrices regularization

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Authors: **Luca Panzeri**

Davide Zaltieri

Students ID: 970831, 975994
Advisor: Prof. Andrea Cappozzo
Academic Year: 2022-23

Abstract

Robust clustering for high-dimensional data poses a significant challenge, as existing robust clustering methods suffer from the curse of dimensionality when p is large, while existing approaches for high-dimensional data are, in general, not robust. This thesis proposes a solution to that challenge, by incorporating high-dimensional covariance matrices estimators into a fast and efficient algorithm for robust constrained clustering, the TCLUST methodology, which has been extensively shown to perform well on contaminated low-dimensional data. The key idea is to exploit three different scatter matrices estimators, the Minimum Regularized Covariance Determinant estimator, the linear shrinkage estimator of Ledoit-Wolf, which is a special case of the previous one, and the sparse CovGlasso estimator, to capture the relationships and dependencies among variables in different ways, allowing the algorithm to effectively handle the complexity and variability of high-dimensional data, whilst being protected against the harmful effect of outliers. This thesis aims to provide a robust clustering solution that is applicable to real-world situations, in which it may be necessary to deal with both a large number of features and data contamination. The problem addressed in this study is the recognition of handwritten digits, which is very challenging due to the high dimensionality of the data, the limited separation between classes and the potential presence of outlying units.

Keywords: High-dimensional data; robust clustering; covariance matrices estimators; handwritten digits recognition.

Abstract in lingua italiana

La clusterizzazione robusta di dati ad alta dimensionalità rappresenta una sfida significativa, poiché i metodi esistenti di clusterizzazione robusta subiscono la curse of dimensionality quando p è grande, mentre gli approcci esistenti per dati ad alta dimensionalità non sono, generalmente, robusti. Questa tesi propone una soluzione a tale sfida, incorporando stimatori di matrici di covarianza ad alta dimensionalità in un algoritmo rapido ed efficiente per la clusterizzazione robusta, il TCLUST, che si è ampiamente dimostrato avere buone prestazioni su dati contaminati a bassa dimensionalità. L'idea chiave è sfruttare tre diversi stimatori di matrici di dispersione, il Minimum Regularized Covariance Determinant estimator, il linear shrinkage estimator di Ledoit-Wolf, che è un caso particolare del precedente, e lo sparse CovGlasso estimator, per catturare relazioni e dipendenze tra variabili in modi diversi, consentendo all'algoritmo di gestire in modo efficace la complessità e la variabilità dei dati ad alta dimensionalità, mentre resta protetto dall'effetto dannoso degli outliers. Questa tesi mira a fornire una soluzione di clusterizzazione robusta che sia applicabile a situazioni reali, in cui potrebbe essere necessario affrontare sia la questione di un elevato numero di variabili, sia quella della contaminazione dei dati. Il problema affrontato in questo studio è il riconoscimento di cifre scritte a mano, che risulta molto impegnativo a causa dell'alta dimensionalità dei dati, della limitata separazione tra le classi e della possibile presenza di outliers.

Parole chiave: Dati ad alta dimensionalità; clustering robusto; stimatori di matrici di covarianza; riconoscimento di cifre scritte a mano.

Contents

Introduction

In recent years there has been a rapid growth in data collection and storage capabilities, leading to the prevalence of high-dimensional datasets in many fields, such as image processing, bioinformatics, and more [6]. Despite the benefits of such data availability, this surge in dimensionality poses a significant challenge for traditional clustering methods, resulting in degraded performance and unreliable outcomes, a phenomenon commonly known as the curse of dimensionality. Furthermore, this situation is even exacerbated by the potential presence of anomalous units, which can further confound the clustering process and hinder the accurate identification of meaningful patterns within the data.

We start by considering the TCLUST [9], a fast and efficient algorithm for robust constrained clustering, as our initial methodology. It has been extensively shown to perform well on contaminated low-dimensional data, but it breaks down, ceasing to work, when the number of variables p is large. Consequently, it becomes imperative to devise a new methodology or, rather, modify the existing one, to ensure its efficiency remains intact even in high-dimensional scenarios.

In detail, when p is large, the TCLUST algorithm faces two significant challenges.

The first one is the initialization procedure, where TCLUST randomly selects only $k \times (p + 1)$ observations to ensure outliers-free initialization and maximize stability. The issue arises when p becomes large, leading this initial subset of $k \times (p+1)$ observations to exceed the total number of observations n. Furthermore, with higher values of p, a greater number of initial observations must be taken into account, consequently increasing the risk of including outliers during the initialization phase. Therefore, we need to modify the initialization procedure, in order to make TCLUST applicable even in high-dimensional settings.

The second issue is more problematic, as, with high-dimensional data, the number of parameters in covariance matrices increases rapidly, causing them to become singular or ill-conditioned. This means they will have determinant equal to zero, making them non-invertible and impossible to estimate using traditional methods that rely on matrix inversion. To address this issue, regularized estimation methods are necessary.

Regularization involves adding a penalty term to the estimated covariance matrix, aiming to reduce sensitivity to errors caused by sparse data or excessive complexity. This penalty term is controlled by a parameter which allows balancing the strength of regularization against the data-driven estimation. By introducing regularization, the estimated covariance matrix becomes more stable and less susceptible to singularity, enabling more robust and reliable estimates even in scenarios with a large number of variables compared to available observations. It is widely used in high-dimensional data analysis, such as in machine learning, financial data analysis, and other applications where accurate estimation of covariance matrices is crucial for making informed decisions and obtaining consistent results.

We propose three distinct regularization methods, capturing the relationships and dependencies among variables in different ways: the Minimum Regularized Covariance Determinant estimator [3], the linear shrinkage estimator of Ledoit-Wolf [15] (a special case of the former), and the sparse CovGlasso estimator [1]. By incorporating these scatter matrix estimators into the TCLUST methodology, our algorithm gains the ability to effectively navigate the complexities and intrinsic variations of high-dimensional data, providing a robust and reliable clustering solution suitable for real-world scenarios.

To demonstrate the practicality and effectiveness of our proposal, we focus our investigation on an interesting problem: handwritten digit recognition [4]. It presents a significant challenge due to the high dimensionality of the data, the limited separation between classes and the potential presence of outlying units. Our analysis centers on a dataset of handwritten digits, where each variable represents the pixel count of a square of the grid that divides the original images. In this study, we will begin with an initial data preparation process before applying the extended TCLUST algorithm, augmented with high-dimensional covariance matrix estimators we previously introduced. The goal is to identify the true labels of the handwritten digits and accurately recognize any digits classified as outliers. Through this process, we aim to showcase the effectiveness of our approach in handling the complexities of high-dimensional datasets, contributing to the advancement of robust clustering methodologies.

The rest of the thesis is organized as follows: Chapter 1 and Chapter 2 will provide an in-depth review of the related literature, exploring the TCLUST methodology in the first chapter and the high-dimensional covariance matrix estimators in the second one. Chapter 3 will present the main methodological contribution of the present manuscript, including all the additions, modifications and adjustments made to the existing TCLUST method. In Chapter 4 we will describe the experimental setup, with dataset descriptions and evaluation metrics used to assess the performance of our approach. We will also

present the results and discuss our final outcomes. Finally, Chapter 5 will conclude the thesis, highlighting the contributions and implications of our research, along with potential future directions in this continuously evolving field.

1 | TCLUST: a robust constrained clustering algorithm

Let $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ be a dataset of observations in $\mathbb{R}^p$, where $\boldsymbol{x}_i = (x_{i1}, ..., x_{ip})^T$ denotes the i-th observation, with $i = 1, ..., n$, and $\phi(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the probability density function of a p-variate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

We consider the following robust constrained clustering problem for a fixed trimming level α: divide the set of indexes $\{1, \ldots, n\}$ into a partition $R_0, R_1, \ldots, R_k$, where R_0 is the set of trimmed observations, thus $\#R_0 = \lceil n\alpha \rceil$, and each cluster $R_1, \ldots, R_k$ is characterized by a center $\boldsymbol{m}_1, ..., \boldsymbol{m}_k$ in $\mathbb{R}^p$, a symmetric positive semidefinite $p \times p$ scatter matrix $\boldsymbol{S}_1, ..., \boldsymbol{S}_k$ and a weight $p_1, ..., p_k$, with $p_j \in [0, 1]$ and $\sum_{j=1}^{k} p_j = 1$. We want the partition $R_0, R_1, \ldots, R_k$ to be the one maximizing the following expression with respect to θ, where $\theta = (p_1, \ldots, p_k, \boldsymbol{m}_1, \ldots, \boldsymbol{m}_k, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_k)$:

$$\sum_{j=1}^{k} \sum_{i \in \mathbb{R}_j} \log \left(p_j \phi(\boldsymbol{x}_i; \boldsymbol{m}_j, \boldsymbol{S}_j) \right). \tag{1.1}$$

Notice that observations in R_0 are not taken into account when computing the expression (1.1), and this allows to avoid the adverse impact of outliers, up to a contamination level α.

The direct maximization of (1.1) without any constraint on the scatter matrices is not a well-defined problem; indeed, a cluster j consisting of just an observation $\boldsymbol{x}_i$ would cause, for instance, (1.1) to tend to infinity by taking $\boldsymbol{m}_j = \boldsymbol{x}_i$ and $\boldsymbol{S}_j$ with $\det(\boldsymbol{S}_j) \to 0$.

To address this issue, Garcìa-Escudero et al. [12] introduced an eigenvalue ratio constraint

on the scatter matrices $\boldsymbol{S}_1, \ldots, \boldsymbol{S}_k$:

$$\frac{\max\limits_{j,l} \lambda_l\left(\boldsymbol{S}_j\right)}{\min\limits_{j,l} \lambda_l\left(\boldsymbol{S}_j\right)} \leq c. \tag{1.2}$$

Here, $\lambda_l\left(\boldsymbol{S}_j\right)$ for $l = 1, \ldots, p$ represents the set of eigenvalues of the scatter matrix $\boldsymbol{S}_j$, $j = 1, \ldots, k$, and $c \geq 1$ is a constant controlling the strength of the constraint (1.2), where the smaller the value of c is, the stronger the restriction imposed on the solution.
The maximization (1.1) under the eigenvalue constraint (1.2) leads to the TCLUST methodology, a fast and efficient algorithm for robust constrained clustering, which shows good performance on low-dimensional data.

The algorithm is as follows:

1. **Initialization:**
   The procedure is initialized `nstart` times by selecting different θ^0, where $\theta^0 = (p_1^0, \ldots, p_k^0, \boldsymbol{m}_1^0, \ldots, \boldsymbol{m}_k^0, \boldsymbol{S}_1^0, \ldots, \boldsymbol{S}_k^0)$. For this purpose, $k \times (p+1)$ observations are randomly selected, followed by the computation of k cluster centers $\boldsymbol{m}_j^0$ and k scatter matrices $\boldsymbol{S}_j^0$, according to the chosen data points. Notice that only $k \times (p+1)$ observations are taken into consideration, in order to maximize safety, aiming to obtain an initialized set of parameters that is outliers-free. Weights $p_1^0, \ldots, p_k^0$ in the interval $(0, 1)$ and summing up to 1 are also randomly chosen.

2. **Concentration step:**
   The following steps are executed until convergence, i.e. $\theta^{t+1} = \theta^t$, or until a maximum number of iterations `iter.max` is reached.

   2.1. **Trimming and cluster assignments (E and C-steps):**
   Based on the current parameters $\theta^t = (p_1^t, \ldots, p_k^t, \boldsymbol{m}_1^t, \ldots, \boldsymbol{m}_k^t, \boldsymbol{S}_1^t, \ldots, \boldsymbol{S}_k^t)$, for each observation $\boldsymbol{x}_i$, the quantities

$$D_j\left(\boldsymbol{x}_i; \theta^t\right) = p_j \phi\left(\boldsymbol{x}_i; \boldsymbol{m}_j^t, \boldsymbol{S}_j^t\right) \quad \text{for } j = 1, \ldots, k, \tag{1.3}$$

   are computed. The $\lceil n\alpha \rceil$ observations with the smallest values of

$$D\left(\boldsymbol{x}_i; \theta^t\right) = \max\left\{D_1\left(\boldsymbol{x}_i; \theta^t\right), \ldots, D_k\left(\boldsymbol{x}_i; \theta^t\right)\right\} \tag{1.4}$$

   are discarded. Each remaining observation $\boldsymbol{x}_i$ is then assigned to a cluster

j such that $D_j(\boldsymbol{x}_i;\theta^t) = D(\boldsymbol{x}_i;\theta^t)$. This yields a partition $R_0, R_1, \ldots, R_k$ of $\{1,\ldots,n\}$ holding the indexes of the trimmed observations in R_0, and the indexes of the observations belonging to cluster j in R_j, for $j = 1,\ldots,k$.

2.2. **Update parameters (M-step):**
For $j = 1,\ldots,k$, given $n_j = \#R_j$, the weights are updated by

$$p_j^{t+1} = n_j/[n(1-\alpha)],$$

and the centers by the sample means

$$\boldsymbol{m}_j^{t+1} = \frac{1}{n_j}\sum_{i\in R_j}\boldsymbol{x}_i.$$

Updating the scatter estimates is more difficult, as the sample covariance matrices

$$\boldsymbol{S}_j^{t+1} = \frac{1}{n_j}\sum_{i\in R_j}\left(\boldsymbol{x}_i - \boldsymbol{m}_j^{t+1}\right)\left(\boldsymbol{x}_i - \boldsymbol{m}_j^{t+1}\right)^T$$

may not satisfy the specified eigenvalue ratio constraint.
In this case, the spectral decomposition of $\boldsymbol{S}_j^{t+1} = \mathbf{U}_j\mathbf{D}_j\mathbf{U}_j^T$ is considered, where $\mathbf{D}_j = \operatorname{diag}(d_{j1}, d_{j2}, \ldots, d_{jp})$ is the diagonal matrix holding the eigenvalues of $\boldsymbol{S}_j^{t+1}$ and $\mathbf{U}_j$ is the orthogonal matrix holding the corresponding eigenvectors. Let us consider truncated eigenvalues defined as

$$d_{jl}^m = \begin{cases} d_{jl} & \text{if } d_{jl} \in [m, cm] \\ m & \text{if } d_{jl} < m \\ cm & \text{if } d_{jl} > cm \end{cases}, \tag{1.5}$$

with m as some threshold value. The scatter matrices are updated as

$$\boldsymbol{S}_j^{t+1} = \mathbf{U}_j\mathbf{D}_j^*\mathbf{U}_j^T,$$

with $\mathbf{D}_j^* = \operatorname{diag}\left(d_{j1}^{m_{\text{opt}}}, d_{j2}^{m_{\text{opt}}}, \ldots, d_{jp}^{m_{\text{opt}}}\right)$, and m_{opt} minimizing

$$m \mapsto \sum_{j=1}^{k} n_j \sum_{l=1}^{p}\left(\log\left(d_{jl}^m\right) + \frac{d_{jl}}{d_{jl}^m}\right). \tag{1.6}$$

As it will be shown in Proposition 1.2, this expression has to be evaluated only $2kp+1$ times to exactly find the minimum.

3. **Target function evaluation:**
   Once the concentration steps are completed, the target function value (1.1) is calculated. The parameters that yield the highest value of this target function are then selected and returned as the output of the algorithm.

The choice of `nstart`, the number of random starts, and `iter.max`, the maximum number of constrained concentration steps, depends on the complexity of the considered dataset. Setting high values for these parameters increases the computational effort but also enhances the objective function of the algorithm converging close to the global optimum.

The following proposition is used to justify the M-step of the previous algorithm:

Proposition 1.1.
If the sets $R_j, j = 1, \ldots, k$, are kept fixed, the maximum of (1.1) under constraint (1.2) can be obtained through the following steps:

i. The best choice of p_j is $p_j = n_j/\lfloor n(1-\alpha)\rfloor$, with $n_j = \#R_j$.

ii. Fixed p_j as given in (i), the best choice for $\boldsymbol{m}_j$ is $\boldsymbol{m}_j = \sum_{i\in R_j} \boldsymbol{x}_i/n_j$.

iii. Fixed the eigenvalues for the matrix $\boldsymbol{S}_j$ and the optimum values given in (i) and (ii) for p_j and $\boldsymbol{m}_j$, the best choice for the set of unitary eigenvectors is the set of unitary eigenvectors of the sample covariance matrix of the observations in R_j.

iv. With the optimal selections from (i), (ii) and (iii), if d_{jl} are the eigenvalues of the sample covariance matrix, the best choice for the truncated eigenvalues $d_{jl}^{m_{\text{opt}}}$ is as in (1.5) with m_{opt} minimizing function (1.6). Then, the best choice for the scatter matrix $\boldsymbol{S}_j$ is obtained with the eigenvectors of the sample covariance matrix of the observations in R_j and with the optimally truncated eigenvalues.

By evaluating the function (1.6) a total of $2pk + 1$ times, it is possible to obtain m_{opt} through the following closed form expression:

Proposition 1.2.
Let us consider $e_1 \leq e_2 \leq \cdots \leq e_{2kp}$ obtained by ordering the following $2pk$ values:

$$d_{11}, d_{12}, \ldots, d_{jl}, \ldots, d_{kp}, d_{11}/c, d_{12}/c, \ldots, d_{jl}/c, \ldots, d_{kp}/c$$

and, $f_1, \ldots, f_{2kp+1}$ any values satisfying:

$$f_1 < e_1 \leq f_2 \leq e_2 \leq \cdots \leq f_{2kp} \leq e_{2kp} < f_{2kp+1}.$$

We can choose m_{opt} as the value of:

$$m_i = \frac{\sum_{j=1}^{k} n_j \left(\sum_{l=1}^{p} d_{jl} \left(d_{jl} < f_i \right) + \frac{1}{c} \sum_{l=1}^{p} d_{jl} \left(d_{jl} > c f_i \right) \right)}{\sum_{j=1}^{k} n_j \left(\sum_{l=1}^{p} \left((d_{jl} < f_i) + (d_{jl} > c f_i) \right) \right)},$$

$i = 1, \ldots, 2kp + 1$, yielding the minimum value of (1.6).

2 | Well-conditioned estimators for high-dimensional covariance matrices

In this chapter we will introduce a series of metodologies to address the challenge of estimating high-dimensional covariance matrices within the TCLUST. This issue arises when dealing with data where the number of variables p exceeds that of observations n, leading to potential problems with traditional covariance matrix estimation methods, primarily due to the singularity issue. To overcome this challenge, we will explore regularization techniques aimed at preventing singularity and enhancing the stability of covariance matrix estimation.

2.1. Minimum Regularized Covariance Determinant estimator

Using the same notation introduced in the previous chapter, we assume that most of the observations $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ come from an elliptical distribution, with location $\boldsymbol{\mu}$ and scatter matrix $\boldsymbol{\Sigma}$, and that remaining observations can be arbitrary outliers, without knowing beforehand which ones they are. The problem is to estimate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ despite the outliers.

The Minimum Regularized Covariance Determinant (MRCD) approach searches for an h-subset of the data (where $n/2 \leq h < n$) whose regularized covariance matrix, that will be defined later, has the lowest possible determinant. Efficiency and robustness of the estimator are both influenced by h, the size of the subset. In terms of robustness, it is recommended to have at most $n - h$ outliers; therefore, in situations where there is a

possibility of encountering many outliers, setting $h = \lceil 0.5n \rceil$ or $h = \lceil 0.75n \rceil$ could be a good choice.

Let H denote a set of h indices reflecting the observations included in the subset, and $\mathcal{H}_h$ the collection of all such sets. For a given H in $\mathcal{H}_h$, the mean and the sample covariance matrix of the subset of data corresponding to the indices in H are computed as follows:

$$\boldsymbol{m}_X(H) = \frac{1}{h} \sum_{i \in H} \boldsymbol{x}_i$$
$$\boldsymbol{S}_X(H) = \frac{1}{h-1} \sum_{i \in H} (\boldsymbol{x}_i - \boldsymbol{m}_X(H)) (\boldsymbol{x}_i - \boldsymbol{m}_X(H))^T . \tag{2.1}$$

First of all, the p variables are standardized to guarantee that the MRCD scatter estimator is location invariant and scale equivariant, so that it is robust to linear transformations of the data. The standardization needs to use a robust univariate location and scale estimate. To achieve this, the medians of all variables are computed and stored in a location vector $\boldsymbol{\nu}_X$; similarly, the scales are estimated using the Qn estimator of Rousseeuw and Croux [19], and they are put in a diagonal matrix $\boldsymbol{D}_X$. The standardized observations are then

$$\boldsymbol{u}_i = \boldsymbol{D}_X^{-1}(\boldsymbol{x}_i - \boldsymbol{\nu}_X).$$

In a second step, two quantities are introduced to define our regularized covariance matrix: a predetermined well-conditioned symmetric and positive definite target matrix $\mathbf{T}$, and a scalar weight coefficient ρ, where $0 \leq \rho \leq 1$, called regularization parameter.
Regarding the target matrix $\mathbf{T}$, we have two choices: we can take the identity matrix with robustly estimated univariate scales on the diagonal, or we can take a non diagonal target matrix, setting $\mathbf{T}$ equal to

$$\mathbf{R}_\xi = \xi \boldsymbol{J}_p + (1 - \xi) \boldsymbol{I}_p, \tag{2.2}$$

with $\boldsymbol{J}_p$ the $p \times p$ matrix of ones, $\boldsymbol{I}_p$ the identity matrix, and $-1/(p-1) < \xi < 1$ to ensure positive definiteness. The parameter ξ in the equicorrelation matrix (2.2) can be estimated by averaging robust correlation estimates over all pairs of variables, under the constraint that the determinant of $\mathbf{R}_\xi$ is above a minimum threshold value.

The regularized covariance matrix of an h-subset of the standardized data is defined as:

$$\mathbf{K}(H) = \rho \mathbf{T} + (1 - \rho) c_\alpha \boldsymbol{S}_U(H), \tag{2.3}$$

where $\boldsymbol{S}_U(H)$ is as defined in (2.1) but for the standardized data instead of the original data, and c_α is a consistency factor that depends on the trimming percentage $\alpha = (n - h)/n$. Notice that, when ρ equals zero, $\mathbf{K}(H)$ becomes the sample covariance $\boldsymbol{S}_U(H)$ weighted by c_α, while when ρ equals one, $\mathbf{K}(H)$ becomes the target matrix $\mathbf{T}$.

It would be convenient to use the spectral decomposition $\mathbf{T} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$ when $\mathbf{T}$ is not diagonal, where $\boldsymbol{\Lambda}$ is the diagonal matrix holding the eigenvalues of $\mathbf{T}$ and $\mathbf{Q}$ is the orthogonal matrix holding the corresponding eigenvectors. The regularized covariance matrix $\mathbf{K}(H)$ could then be rewritten as

$$\mathbf{K}(H) = \mathbf{Q}\boldsymbol{\Lambda}^{1/2}[\rho\boldsymbol{I} + (1-\rho)c_\alpha \boldsymbol{S}_W(H)]\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^T,$$

where $\boldsymbol{S}_W(H)$ is as the previous ones, but for the transformed standardized observations $\boldsymbol{w}_i = \boldsymbol{\Lambda}^{-1/2}\mathbf{Q}^T\boldsymbol{u}_i$.

We introduce the notion of condition number of a matrix, as the ratio between the largest and the smallest eigenvalue of that matrix, to measures its numerical stability: a matrix is well-conditioned if its condition number is moderate, whereas it is ill-conditioned if its condition number is high. Therefore, to guarantee that $\mathbf{K}(H)$ is well-conditioned, it is sufficient to bound the condition number of $\rho\boldsymbol{I} + (1-\rho)c_\alpha\boldsymbol{S}_W(H)$. Since the eigenvalues of $\rho\boldsymbol{I} + (1-\rho)c_\alpha\boldsymbol{S}_W(H)$ equal $\rho + (1-\rho)\lambda$, the corresponding condition number is

$$CN(\rho) = \frac{\rho + (1-\rho)max\{\lambda\}}{\rho + (1-\rho)min\{\lambda\}}.$$

In practice, regarding the choice of ρ, it is suggested a data-driven approach that determines the value of ρ based on the lowest non-negative value for which the condition number of $\rho\boldsymbol{I} + (1-\rho)c_\alpha\boldsymbol{S}_W(H)$ is at most κ, where a good choice could be $\kappa = 50$, that is also the default value in the `CovMrcd` implementation in the R package `rrcov` [21].

The MRCD subset H_{MRCD} is defined by minimizing the determinant of the regularized covariance matrix $\mathbf{K}(H)$ in (2.3):

$$H_{MRCD} = \underset{H\in\mathcal{H}_h}{\operatorname{argmin}} \left(\det(\mathbf{K}(H))^{1/p}\right). \tag{2.4}$$

Since $\mathbf{T}, \mathbf{Q}$ and $\boldsymbol{\Lambda}$ are fixed, H_{MRCD} can also be written as

$$H_{MRCD} = \underset{H\in\mathcal{H}_h}{\operatorname{argmin}} \left(\det\left(\rho\boldsymbol{I} + (1-\rho)c_\alpha\boldsymbol{S}_W(H)\right)^{1/p}\right).$$

Once H_{MRCD} is determined, the MRCD location and scatter estimates of the original data matrix are computed as

$$\boldsymbol{m}_{MRCD} = \boldsymbol{\nu}_X + \boldsymbol{D}_X \boldsymbol{m}_U(H_{MRCD})$$
$$\mathbf{K}_{MRCD} = \boldsymbol{D}_X \mathbf{Q} \boldsymbol{\Lambda}^{1/2} [\rho \boldsymbol{I} + (1-\rho) c_\alpha \boldsymbol{S}_W(H_{MRCD})] \boldsymbol{\Lambda}^{1/2} \mathbf{Q}^T \boldsymbol{D}_X. \tag{2.5}$$

Finally, the MRCD precision matrix, that is the inverse of the scatter matrix (2.5), can be obtained as

$$\mathbf{K}_{MRCD}^{-1} = \boldsymbol{D}_X^{-1} \mathbf{Q}^T \boldsymbol{\Lambda}^{-1/2} [\rho \boldsymbol{I}_p + (1-\rho) c_\alpha \boldsymbol{S}_W(H_{MRCD})]^{-1} \boldsymbol{\Lambda}^{-1/2} \mathbf{Q} \boldsymbol{D}_X^{-1}.$$

The ongoing solution for the MRCD approach consists of iteratively applying the so-called C-step, improving an h-subset H_1 by computing its mean and covariance matrix, and then selecting the h observations with the smallest Mahalanobis distance to form a new subset H_2. This theorem generalizes the C-step theorem by Rousseeuw and Van Driessen [18] for the Minimum Covariance Determinant (MCD) methodology to the context of regularized covariance matrices in the MRCD estimation. The MCD [17] forms the foundational framework upon which the MRCD is constructed. While both techniques share the common principle of minimizing the determinant of the covariance matrix, the MRCD improves itself by its capability to manage high-dimensional data through the incorporation of regularization in covariance matrix estimation.

Theorem 2.1
Let $\mathbf{X}$ be a dataset of n points in p dimensions, and take any $n/2 < h < n$ and $0 < \rho < 1$. Starting from an h-subset H_1, one can compute $\boldsymbol{m}_1 = \frac{1}{h}\sum_{i \in H_1} \boldsymbol{x}_i$ and $\boldsymbol{S}_1 = \frac{1}{h}\sum_{i \in H_1} (\boldsymbol{x}_i - \boldsymbol{m}_1)(\boldsymbol{x}_i - \boldsymbol{m}_1)^T$. The matrix

$$\mathbf{K}_1 = \rho \mathbf{T} + (1-\rho) \boldsymbol{S}_1$$

is positive definite hence invertible, so we can compute

$$d_1(i) = (\boldsymbol{x}_i - \boldsymbol{m}_1)^T \mathbf{K}_1^{-1} (\boldsymbol{x}_i - \boldsymbol{m}_1)$$

for $i = 1, \ldots, n$. Let H_2 be an h-subset for which

$$\sum_{i \in H_2} d_1(i) \leq \sum_{i \in H_1} d_1(i) \tag{2.6}$$

and compute $\boldsymbol{m}_2 = \frac{1}{h}\sum_{i\in H_2}\boldsymbol{x}_i$, $\boldsymbol{S}_2 = \frac{1}{h}\sum_{i\in H_2}(\boldsymbol{x}_i - \boldsymbol{m}_2)(\boldsymbol{x}_i - \boldsymbol{m}_2)^T$ and $\mathbf{K}_2 = \rho\mathbf{T} + (1-\rho)\boldsymbol{S}_2$. Then

$$\det(\mathbf{K}_2) \leq \det(\mathbf{K}_1),$$

with equality if and only if $\boldsymbol{m}_2 = \boldsymbol{m}_1$ and $\mathbf{K}_2 = \mathbf{K}_1$.

The MRCD algorithm, which makes use of the just presented generalized C-step, can be found in [3].

2.2. Linear Shrinkage estimator of Ledoit-Wolf

As in the previous section, let us assume that our observations come from an elliptical distribution, with location $\boldsymbol{\mu}$ (in this case, equal to $\mathbf{0}$) and covariance matrix $\boldsymbol{\Sigma}$. Our goal is to find the well-conditioned estimator for $\boldsymbol{\Sigma}$ as the linear combination $\boldsymbol{\Sigma}^* = \rho_1\boldsymbol{I} + \rho_2\boldsymbol{S}$ that minimizes the expected quadratic loss $E\left[\|\boldsymbol{\Sigma}^* - \boldsymbol{\Sigma}\|^2\right]$.

To achieve this, we need four scalar functions of $\boldsymbol{\Sigma}$:

- $\mu = \langle\boldsymbol{\Sigma}, \boldsymbol{I}\rangle$
- $\alpha^2 = \|\boldsymbol{\Sigma} - \mu\boldsymbol{I}\|^2$
- $\beta^2 = E\left[\|\boldsymbol{S} - \boldsymbol{\Sigma}\|^2\right]$
- $\delta^2 = E\left[\|\boldsymbol{S} - \mu\boldsymbol{I}\|^2\right]$.

Here, we are considering the Frobenius norm $\|\boldsymbol{A}\| = \sqrt{\mathrm{tr}\left(\boldsymbol{A}\boldsymbol{A}^T\right)/p}$, whose associated inner product is $\langle\boldsymbol{A_1}, \boldsymbol{A_2}\rangle = \mathrm{tr}\left(\boldsymbol{A_1}\boldsymbol{A_2}^T\right)/p$. It is important to note that we need to assume our variables have finite fourth moments, so that β^2 and δ^2 are finite.

Having all these elements, it can be stated that the optimal linear combination $\boldsymbol{\Sigma}^* = \rho_1\boldsymbol{I} + \rho_2\boldsymbol{S}$ of the identity matrix $\boldsymbol{I}$ and the sample covariance matrix $\boldsymbol{S}$ is the standard solution to a simple quadratic programming problem under linear equality constraint, as we will see in the following theorem:

Theorem 2.2
Consider the optimization problem:

$$\min_{\rho_1,\rho_2} E\left[\|\boldsymbol{\Sigma}^* - \boldsymbol{\Sigma}\|^2\right] \quad \text{s.t.} \quad \boldsymbol{\Sigma}^* = \rho_1 \boldsymbol{I} + \rho_2 \boldsymbol{S}, \tag{2.7}$$

where the coefficients ρ_1 and ρ_2 are nonrandom. Its solution verifies:

$$\boldsymbol{\Sigma}^* = \frac{\beta^2}{\delta^2}\mu\boldsymbol{I} + \frac{\alpha^2}{\delta^2}\boldsymbol{S},$$

$$E\left[\|\boldsymbol{\Sigma}^* - \boldsymbol{\Sigma}\|^2\right] = \frac{\alpha^2\beta^2}{\delta^2}.$$

The estimator $\boldsymbol{\Sigma}^*$ has a fundamental drawback: it requires hindsight knowledge of the four scalar functions of the true, unobservable, covariance matrix $\boldsymbol{\Sigma}$. To address this issue, Ledoit and Wolf [15] decided to explore an asymptotic framework and search, asymptotically, for consistent estimators for μ, α^2, β^2 and δ^2.
They chose general asymptotics, rather than standard asymptotics, because they wanted the number of variables p to go to infinity at the same speed as the number of observations n, being in situations where p is of the same order as n or even larger. In this framework, the optimal shrinkage intensity, generally, does not vanish asymptotically, as it happens under standard asymptotics instead, but rather it tends to a limiting constant that we will be able to estimate consistently. The idea, then, is to use the estimated shrinkage intensity in order to arrive at an efficient unbiased estimator.

Let $n = 1, 2, \ldots$ index a sequence of statistical models. For every $n, \mathbf{X}_n$ is a $p_n \times n$ matrix of n iid observations on a system of p_n random variables with zero mean and covariance matrix $\boldsymbol{\Sigma}_n$. By considering the spectral decomposition $\boldsymbol{\Sigma}_n = \boldsymbol{\Gamma}_n \boldsymbol{\Lambda}_n \boldsymbol{\Gamma}_n^T$, where $\boldsymbol{\Lambda}_n$ is the diagonal matrix holding the eigenvalues of $\boldsymbol{\Sigma}_n$ and $\boldsymbol{\Gamma}_n$ is the orthogonal matrix holding the corresponding eigenvectors, we define $\boldsymbol{Y}_n = \boldsymbol{\Gamma}_n^T \mathbf{X}_n$ as a $p_n \times n$ matrix of n iid observations on a system of p_n uncorrelated random variables that spans the same space as the original system.

Firstly, we assume that there exists a constant K, independent of n, such that $p_n/n \leqslant K$, hence, the number of variables p_n can change, and even go to infinity, with the number of observations n, but not too fast (standard asymptotics are included as a particular case). Moreover, we impose restrictions on the higher moments of $\boldsymbol{Y}_n$. Specifically, we assume that the eighth moment of $\boldsymbol{Y}_n$ is bounded, on average, and that, products of uncorrelated random variables are themselves uncorrelated, on average, in the limit.

By defining the sample covariance matrix $\boldsymbol{S}_n = \mathbf{X}_n\mathbf{X}_n^T/n$, we have:

- $\mu_n = \langle \boldsymbol{\Sigma}_n, \boldsymbol{I}_n \rangle_n$
- $\alpha_n^2 = \|\boldsymbol{\Sigma}_n - \mu_n \boldsymbol{I}_n\|_n^2$
- $\beta_n^2 = E\left[\|\boldsymbol{S}_n - \boldsymbol{\Sigma}_n\|_n^2\right]$
- $\delta_n^2 = E\left[\|\boldsymbol{S}_n - \mu_n \boldsymbol{I}_n\|_n^2\right]$.

These four scalars are well behaved asymptotically, indeed they remain bounded as $n \to \infty$.

If we consider $\boldsymbol{\Sigma}_n^* = \frac{\beta_n^2}{\delta_n^2}\mu_n \boldsymbol{I}_n + \frac{\alpha_n^2}{\delta_n^2}\boldsymbol{S}_n$, the solution of Theorem 2.2.1 under general asymptotics, it is not an efficient unbiased estimator, because it depends on four scalar function of the true, unobservable, covariance matrix $\boldsymbol{\Sigma}_n$: μ_n, α_n^2, β_n^2 and δ_n^2. Therefore, we proceed by estimating them consistently.

Finally, we obtain:

- $m_n = \langle \boldsymbol{S}_n, \boldsymbol{I}_n \rangle_n$
- $d_n^2 = \|\boldsymbol{S}_n - m_n \boldsymbol{I}_n\|_n^2$
- $b_n^2 = \min\left(\bar{b}_n^2, d_n^2\right)$, where $\bar{b}_n^2 = \frac{1}{n^2}\sum_{k=1}^n \left\|\boldsymbol{x}_k^n \left(\boldsymbol{x}_k^n\right)^T - \boldsymbol{S}_n\right\|_n^2$, with $\boldsymbol{x}_k^n$ denoting the k-th column of $\mathbf{X}_n$
- $a_n^2 = d_n^2 - b_n^2$

as consistent estimators for μ_n, δ_n^2, β_n^2, and α_n^2, respectively. All the technical details and proofs about the aforementioned quantities can be found in the paper of Ledoit and Wolf [15].

By replacing the unobservable scalars in the formula defining $\boldsymbol{\Sigma}_n^*$ with their consistent estimators, we obtain our efficient unbiased estimator of the covariance matrix:

$$\boldsymbol{S}_n^* = \frac{b_n^2}{d_n^2} m_n \boldsymbol{I}_n + \frac{a_n^2}{d_n^2}\boldsymbol{S}_n. \tag{2.8}$$

The following theorem will show that $\boldsymbol{S}_n^*$ has the same asymptotic properties as $\boldsymbol{\Sigma}_n^*$. Thus, we can neglect the error that we introduce when we replace the unobservable parameters $\mu_n, \alpha_n^2, \beta_n^2$ and δ_n^2 by their estimators.

Theorem 2.3
$\boldsymbol{S}_n^*$ is a consistent estimator of $\boldsymbol{\Sigma}_n^*$, i.e. $\|\boldsymbol{S}_n^* - \boldsymbol{\Sigma}_n^*\|_n \xrightarrow{\text{q.m.}} 0$. As a consequence, $\boldsymbol{S}_n^*$ has the same asymptotic expected loss as $\boldsymbol{\Sigma}_n^*$, i.e. $E\left[\|\boldsymbol{S}_n^* - \boldsymbol{\Sigma}_n\|_n^2\right] - E\left[\|\boldsymbol{\Sigma}_n^* - \boldsymbol{\Sigma}_n\|_n^2\right] \to 0$.

The final step consists in demonstrating that $\boldsymbol{S}_n^*$, that we obtained as a consistent estimator for $\boldsymbol{\Sigma}_n^*$, possesses an important optimality property. We already know that $\boldsymbol{\Sigma}_n^*$, hence $\boldsymbol{S}_n^*$ in the limit, is optimal among the linear combinations of the identity and the sample covariance matrix with nonrandom coefficients. Now, we will show that $\boldsymbol{S}_n^*$ is still optimal within a bigger class: the linear combinations of $\boldsymbol{I}_n$ and $\boldsymbol{S}_n$ with random coefficients. This class includes both the linear combinations that represent efficient unbiased estimators, and those with coefficients that require hindsight knowledge of the true, therefore unobservable, covariance matrix $\boldsymbol{\Sigma}_n$.

Let $\boldsymbol{\Sigma}_n^{**}$ denote the linear combination of $\boldsymbol{I}_n$ and $\boldsymbol{S}_n$ with minimum quadratic loss. It solves:

$$\min_{\rho_1,\rho_2} ||\boldsymbol{\Sigma}_n^{**} - \boldsymbol{\Sigma}_n\|_n^2 \quad \text{s.t.} \quad \boldsymbol{\Sigma}_n^{**} = \rho_1 \boldsymbol{I}_n + \rho_2 \boldsymbol{S}_n.$$

In contrast to the optimization problem in Theorem 2.2 with solution $\boldsymbol{\Sigma}_n^*$, here we minimize the loss instead of the expected loss, and we allow the coefficients ρ_1 and ρ_2 to be random.
$\boldsymbol{\Sigma}_n^{**}$ is not an an efficient unbiased estimator, since it turns out that it is a function of $\boldsymbol{\Sigma}_n$. However, we can show that $\boldsymbol{S}_n^*$ is a consistent estimator of $\boldsymbol{\Sigma}_n^{**}$, thereby implying that $\boldsymbol{S}_n^*$ has the same asymptotic expected loss as $\boldsymbol{\Sigma}_n^{**}$. Both $\boldsymbol{\Sigma}_n^*$ and $\boldsymbol{\Sigma}_n^{**}$ have the same asymptotic properties as $\boldsymbol{S}_n^*$, therefore, they also have the same asymptotic properties as each other.

The most important result is the following: the efficient unbiased estimator $\boldsymbol{S}_n^*$ defined in (2.8) has uniformly minimum quadratic risk asymptotically among all the linear combinations of the identity with the sample covariance matrix, including those that are efficient unbiased estimators, and even those that use hindsight knowledge of the true covariance matrix. Thus, it is legitimate to say that $\boldsymbol{S}_n^*$ is an asymptotically optimal linear shrinkage estimator of the covariance matrix $\boldsymbol{\Sigma}$ with respect to quadratic loss under general asymptotics.

Notice that the linear shrinkage estimator of Ledoit-Wolf can be considered as a particular case of the MRCD estimator, when the subset H of the equation (2.3) corresponds to the entire sample [3]. However, this particular Ledoit-Wolf estimator lacks robustness to outliers, as they are included in its computation. As an extension, the MRCD

approach can be viewed as a robust enhancement of the Ledoit-Wolf estimator, incorporating the minimum covariance determinant principle and thereby improving its resilience against outliers. At any rate, since the Ledoit-Wolf estimator will be employed within the TCLUST algorithm, it is sensible to make use of it being robustness already enforced by the TCLUST procedure. On the other hand, MRCD will provide a doubly-robust extension, as outliers are taken care of both in clustering procedure as well as within the modified M-step that makes use of MRCD to compute the regularized covariance matrices.

2.3. Sparse CovGlasso estimator

Suppose that our observations come from a p-variate Gaussian distribution with zero mean and covariance matrix $\boldsymbol{\Sigma}$. The log-likelihood is

$$\ell(\boldsymbol{\Sigma}) = -\frac{np}{2}\log(2\pi) - \frac{n}{2}\log(\det(\boldsymbol{\Sigma})) - \frac{n}{2}\operatorname{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}),$$

where $\boldsymbol{S}$ is still the sample covariance matrix. Since we want a lasso estimator, we add a lasso penalty of the form $\lambda\|\boldsymbol{P} * \boldsymbol{\Sigma}\|_1$ to the likelihood, where λ is the lasso regularization parameter, $\boldsymbol{P}$ is an arbitrary matrix with non-negative elements and $*$ denotes the elementwise multiplication. Our goal is to find $\boldsymbol{\Sigma}$ positive definite that minimizes

$$\log(\det(\boldsymbol{\Sigma})) + \operatorname{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}) + \lambda\|\boldsymbol{P} * \boldsymbol{\Sigma}\|_1, \tag{2.9}$$

where, for a matrix $\boldsymbol{A}$, we define $\|\boldsymbol{A}\|_1 = \sum_{ij}|A_{ij}|$.

Two common choices for $\boldsymbol{P}$ would be the matrix of all ones, or this same matrix, but with zeros on the diagonal, to avoid shrinking diagonal elements of $\boldsymbol{\Sigma}$. The objective function (2.9) is not convex, imposing computational challenges for minimizing it.
Initially, Bien and Tibshirani [1] proposed a majorize-minimize approach to approximately minimize (2.9), but two years later Wang [23] suggested a new algorithm, showing it has several advantages with respect to the previous one, including simplicity, computational speed and numerical stability. The minimization of the objective function (2.9) using the coordinate descent method involves updating the covariance matrix $\boldsymbol{\Sigma}$ one column and row at a time while keeping the rest fixed. In particular, $\boldsymbol{\Sigma}$ and $\boldsymbol{S}$ are partitioned as

follows:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\sigma}_{12} \\ \boldsymbol{\sigma}_{12}^T & \sigma_{22} \end{pmatrix}$$
$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{S}_{11} & \boldsymbol{S}_{12} \\ \boldsymbol{S}_{12}^T & s_{22} \end{pmatrix},$$

where:

- $\boldsymbol{\Sigma}_{11}$ and $\boldsymbol{S}_{11}$ are the covariance matrix and the sample covariance matrix of the first $(p-1)$ variables, respectively.
- $\boldsymbol{\sigma}_{12}$ and $\boldsymbol{S}_{12}$ are the covariances and the sample covariances between the first $(p-1)$ variables and the last variable, respectively.
- σ_{22} and s_{22} are the variance and the sample variance of the last variable, respectively.

Let us define $\boldsymbol{\beta} = \boldsymbol{\sigma}_{12}$ and $\gamma = \sigma_{22} - \boldsymbol{\sigma}_{12}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\sigma}_{12}$. Using block matrix inversion, the inverse of $\boldsymbol{\Sigma}$ is given by:

$$\boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \boldsymbol{\Sigma}_{11}^{-1} + \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta} \boldsymbol{\beta}^T \boldsymbol{\Sigma}_{11}^{-1} \gamma^{-1} & -\boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta} \gamma^{-1} \\ -\boldsymbol{\beta}^T \boldsymbol{\Sigma}_{11}^{-1} \gamma^{-1} & \gamma^{-1} \end{pmatrix}.$$

Therefore, the three terms in (2.9) can be expressed as functions of $\boldsymbol{\beta}$ and γ as follows:

- $\log(\det(\boldsymbol{\Sigma})) = \log(\gamma) + c_1$
- $\operatorname{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}) = \boldsymbol{\beta}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{S}_{11} \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta} \gamma^{-1} - 2\boldsymbol{S}_{12}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta} \gamma^{-1} + s_{22} \gamma^{-1} + c_2$
- $\lambda \|\boldsymbol{\Sigma}\|_1 = 2\lambda \|\boldsymbol{\beta}\|_1 + \lambda \left(\boldsymbol{\beta}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta} + \gamma \right) + c_3$, where we are considering $\boldsymbol{P}$ as the matrix of all ones.

The objective function with respect to $\boldsymbol{\beta}$ and γ is:

$$\min_{\boldsymbol{\beta}, \gamma} \left\{ \log(\gamma) + \boldsymbol{\beta}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{S}_{11} \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta} \gamma^{-1} - 2\boldsymbol{S}_{12}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta} \gamma^{-1} + s_{22} \gamma^{-1} + 2\lambda \|\boldsymbol{\beta}\|_1 + \lambda \boldsymbol{\beta}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta} + \lambda \gamma \right\}. \tag{2.10}$$

For γ, removing terms in (2.10) that do not depend on γ gives

$$\min_{\gamma} \left\{ \log(\gamma) + a\gamma^{-1} + \lambda \gamma \right\}, \tag{2.11}$$

where $a = \boldsymbol{\beta}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{S}_{11} \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta} - 2\boldsymbol{S}_{12} \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta} + s_{22}$. The problem (2.11) is solved by:

$$\hat{\gamma} = \begin{cases} a & \text{if } \lambda = 0 \\ (-1 + \sqrt{1 + 4a\lambda})/(2\lambda) & \text{if } \lambda \neq 0 \end{cases} .$$

For $\boldsymbol{\beta}$, removing terms in (2.10) that do not depend on $\boldsymbol{\beta}$ gives

$$\min_{\boldsymbol{\beta}} \left\{ \boldsymbol{\beta}^T \mathbf{V} \boldsymbol{\beta} - 2\mathbf{u}^T \boldsymbol{\beta} + 2\lambda \|\boldsymbol{\beta}\|_1 \right\}, \tag{2.12}$$

where $\mathbf{V} = (v_{ij}) = \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{S}_{11} \boldsymbol{\Sigma}_{11}^{-1} \gamma^{-1} + \lambda \boldsymbol{\Sigma}_{11}^{-1}$ and $\mathbf{u} = \Sigma_{11}^{-1} \boldsymbol{S}_{12} \gamma^{-1}$. The problem in (2.12) is a lasso problem and can be efficiently solved by coordinate descent algorithms [8, 24]. Specifically, for $j \in \{1, \ldots, p-1\}$, the minimum point of (2.12) along the coordinate direction in which β_j varies is:

$$\hat{\beta}_j = \mathcal{S}\left(u_j - \sum_{k \neq j} v_{kj} \hat{\beta}_k, \lambda \right) / v_{jj}, \tag{2.13}$$

where $\mathcal{S}$ is the soft-threshold operator:

$$\mathcal{S}(x, t) = \text{sign}(x) \max(|x| - t, 0).$$

The update (2.13), for $j = 1, \ldots, p-1$, is iterated until convergence, and finally columns are updated as $(\boldsymbol{\sigma}_{12} = \boldsymbol{\beta}, \sigma_{22} = \ \gamma + \boldsymbol{\beta}^T \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\beta})$ followed by cycling through all columns until convergence. This algorithm can be viewed as a block coordinate descent method with p blocks of $\boldsymbol{\beta}$'s and other p blocks of γ's, and it can be found in [23]. Its convergence to a stationary point is supported by the theoretical results of Tseng [22].

This procedure is implemented within the `covglasso` function of the `covglasso` package itself [7], which will be employed to integrate the CovGlasso estimator into the TCLUST methodology in the upcoming chapter.

3 | Work development

3.1. Initial proposal: incorporating the MRCD estimator within TCLUST

The initial proposal is to incorporate the Minimum Regularized Covariance Determinant estimator, previously introduced in Section 2.1, into the TCLUST methodology, in order to address the challenge of robust clustering for high-dimensional data. In high dimensions, one of the main problems of the TCLUST algorithm is the covariance matrices estimation. Indeed, their size grows quadratically with p, causing them to become singular or ill-conditioned.

The covariance matrix estimator used in the TCLUST algorithm is based on the standard sample covariance matrix, enhanced by the inclusion of the eigenvalue ratio constraint (1.2). This constraint is implemented to improve the robustness and to protect against spurious solutions. It is an approach that limits the variation among the eigenvalues, trying to prevent a single dimension from overly influencing the covariance matrix estimate. Through the application of constraint (1.2), the gap between the largest and smallest eigenvalues is limited, thereby contributing to the maintenance of a certain level of stability in the estimation. However, this method fails when dealing with high-dimensional data, and the sample covariance matrix, even if protected by the constraint (1.2), may not be a good estimator due to several reasons:

- **Low observation-dimension ratio:**
  In high-dimensional environments, there is often a high number of variables compared to the available number of observations. This can lead to an unstable and unreliable estimation of covariances between variables, as the information from a limited number of observations might be insufficient to capture the true relationships between variables.

- **Noise sensitivity:**
  In high-dimensional contexts, data noise can have a significant impact on covariance estimates. Since noise is often randomic and unpredictable, it can distort covariance

estimates and make the sample covariance matrix estimator less accurate.

- **High correlation:**
  When highly correlated or redundant variables are present, the sample covariance matrix estimator might overestimate or underestimate covariances due to redundant information. This can lead to a loss of precision in covariance estimates.

- **Numerical instability:**
  In high-dimensional settings, numerical computations associated with covariance calculation may become unstable due to approximation issues or rounding errors. This can adversely affect the quality of estimates.

In high dimensionality, it is often necessary to apply regularization techniques to manage problem complexity. The sample covariance matrix estimator does not inherently incorporate regularization, which may result in suboptimal and less stable estimates. Therefore, we attempt to replace it with the Minimum Regularized Covariance Determinant estimator, which, being a regularized estimation method, allows the covariance matrix to achieve greater stability and reduced susceptibility to singularity. This would result in obtaining more robust and reliable estimates.

We need to develop a new algorithm, building upon the existing TCLUST approach.
Our initial focus is to identify the limitations of the TCLUST algorithm in handling high-dimensional data. For this purpose, we will conduct an in-depth analysis of the `.tclust.R` function within the `tclust` package [10]. This function is composed by various subfunctions, which we carefully examine one by one.

An important issue arises immediately with the `.InitClusters` subfunction, used for calculating the initial cluster assignments and parameters. Specifically, cycling over the number of clusters k, it generates a random sample of $p+1$ integers ranging from 1 to n without replacement, and it extracts the rows of the observation matrix corresponding to the integers previously generated, retaining all columns. Finally, these submatrices are used to compute the initial means and covariance matrices, through the sample mean estimator and the sample covariance matrix estimator, respectively. Therefore, during the initialization procedure, TCLUST randomly selects a subset of $k \times (p+1)$ observations, which is smaller than the entire sample size n in low-dimensional settings, in order to ensure an initialization free from outliers and to maximize stability. The issue arises when p exceeds n, causing this initial subset to become significantly larger than the total number of observations, necessitating a change in the initialization procedure. Furthermore, we previously observed that the sample covariance matrix estimator is not a good estimator

for covariance matrices in high-dimensional settings. Hence, even that estimator needs to be replaced with a new one.

In the following, we will provide a detailed explanation of the initialization procedure we implement:

1. **Generation of initial cluster assignments:**
   We start by generating a preliminary assignment of observations to clusters, where each observation in our dataset is randomly assigned to one of the available clusters.

2. **Designation of some observation as outliers:**
   Subsequently, we want to designate a portion of observations as outliers, specifically equal to the number of trimmed observations $n\alpha$, where α is the trimming percentage passed as input to the algorithm. To do so, we randomly select this portion of observations among all the n's, and we assign them to a special cluster labeled as cluster 0. This process involves nullifying a fraction of the random cluster assignments, effectively categorizing certain observations as outliers.

3. **Computation of cluster weights:**
   Each cluster will now have an associated weight. This weight is computed by dividing the number of elements in the cluster by a normalization constant, which is the number of non-trimmed observations $n(1 - \alpha)$. In practice, we are trying to quantify the significance of each cluster relative to the total number of observations.

4. **Extraction of final cluster assignments:**
   Lastly, we extract the final cluster assignments, including only those observations that have not been trimmed during Step 2.

At this point, we generate a collection of matrices, each representing a distinct cluster. These matrices are essentially stores for data associated with their respective clusters. As we iterate through these matrices, we apply the new Minimum Regularized Covariance Determinant estimator, replacing the previously employed sample covariance matrix estimator.

The MRCD estimator is computed using the `CovMrcd` function of the `rrcov` package [21]. This function takes as input a matrix of observations specific to each cluster, along with other parameters including subset size, maximum concentration steps, regularization parameter, target matrix structure, and maximum allowed condition number.
The `CovMrcd` function computes the Minimum Regularized Covariance Determinant estimator for location and scatter, returning an S4 object of class `CovMrcd-class` [20]. This

class contains MRCD estimates for multivariate location and scatter computed by the algorithm, as well as the number of observations used for estimation and the best subset used for computation. Other elements in this class include the estimated regularization parameter and the inverse of the estimated covariance matrix.

Returning to the discussion of TCLUST steps, `.InitClusters` finally stores the estimated mean vector and covariance matrix of each cluster in the `center` matrix and `sigma` array of the `iter` object, respectively. The subfunction then returns this object as final step. Notice that this object will be presented in detail later.
In summary, `.InitClusters` performs a sequence of essential tasks, including the generation of initial cluster assignments, with some observation designated as outliers, the computation of initial cluster weights, the creation of initial cluster matrices and the estimation of crucial parameters for each cluster. It is important to note that, just as in the TCLUST algorithm, it is necessary to have multiple initializations here to ensure the EM starts from a good starting point.

An other important subfunction is `.findClustAssig`, which finds current cluster assignments based on given location and scatter matrix. It works as follows:

1. It computes the likelihood for each observation in each cluster, using data matrix, cluster means and cluster covariance matrices to initially compute the multivariate normal densities, and then multiplying them by the cluster weights, as in (1.3).
2. It updates the assignment of observations to clusters based on the computed likelihoods, assigning each observation to the cluster that maximizes its likelihood.
3. It applies a trimming procedure to exclude observations with the lowest likelihoods from assignment. In particular, the $\lceil n\alpha \rceil$ observations with the smallest values of (1.4) are discarded as outliers.
4. It checks for convergence by comparing the current and previous assignments, updating a dedicated parameter to TRUE if the convergence is achieved.
5. It computes the sizes of the updated clusters.
6. It constructs a matrix indicating the membership of each observation to each cluster. In particular, its (i,j)-th element is equal to 1 if observation i belongs to cluster j, and it is equal to 0 otherwise.
7. It updates the cluster weights, finally returning the updated `iter` object.

In the previous, we needed to modify the original `.findClustAssig` subfunction because

it was constructing fuzzy membership matrices, whereas our objective is to create binary membership matrices. A fuzzy membership matrix is a generalization of a standard binary membership matrix, where each (i,j)-th element of the matrix is a number ranging from 0 to 1, representing the probability that observation i belongs to cluster j. Differently from a binary one, where each cell contains a binary value that indicates whether an observation belongs to a specific cluster (value equal to 1) or not (value equal to 0), in a fuzzy membership matrix, the values within the cells represent degree of membership of observations to clusters, resulting in values span from 0 to 1, with row elements summing to 1.

Now, `.estimClustPar` is the subfunction that computes the mean vector and the covariance matrix for each of the provided clusters. We follow the same approach as in `.InitClusters` to update the original function. Consequently, we create a set of matrices, each representing a unique cluster, and apply the MRCD estimator, while iterating through these matrices, to compute and store the mean vector and covariance matrix for each cluster. In addition, we introduce a safeguarding condition to ensure the correct execution of the algorithm.

In detail, sometimes it could happen that the clustering algorithm generates empty clusters, meaning clusters that do not contain any data points. This may occur in situations where the algorithm fails to find enough similar data points to assign to a particular cluster, and it results in automatically reducing the total number of clusters. For instance, if you set the algorithm to generate a certain number of clusters, but some of them become empty and are removed, you would end up with fewer clusters than you originally intended. This might cause interpretation problems and compromise the analysis. Furthermore, even clusters with very few members (such as one or two) could be problematic. They may lead to unreliable estimates of cluster parameters as calculations would be significantly affected by such a small number of data points. This negatively impact the accuracy and reliability of the analysis results. To address these issues, we introduce a threshold on the minimum cluster size. This means establishing a rule that prevents clusters from having a membership count below a certain threshold, such as two observations. This way, we avoid empty clusters and prevent them from having very few members, ensuring there is enough data to reliably estimate cluster parameters.

So, if a cluster contains at least three data points, we proceed with the MRCD estimation of means and covariance matrices by applying the `CovMrcd` function, which we already discussed above. Conversely, if a cluster is empty or has less than three observations belonging to it, the MRCD estimation is skipped, and subsequently that initialization will

not be passed for evaluation as the best initialization in the main function.

The final subfunction required is `.calcobj`, which computes the values of objective function (1.1) without modification.

We do not use the eigenvalue ratio constraint because we are already applying regularization. Therefore, we no longer require all the other subfunctions present in the original TCLUST algorithm.

Now, let us turn our attention to the main function, called `.mrcd_in_tclust`. It takes as input:

- Matrix or dataframe of dimension $n \times p$, containing the observations.
- Number of outliers initially searched for.
- Proportion of observations to be trimmed.
- Number of random initialization to be performed.
- Maximum number of concentration steps to be performed. The concentration steps are stopped, whenever two consecutive steps lead to the same data partition.

This function performs the following operations:

1. **Data preparation:**
   It checks the format of the input data and converts it to a matrix if it is a data frame or a vector. In addition, it verifies that the data is numeric, otherwise, it generates an error.
2. **Parameter setup:**
   It initializes a set of parameters within a structure called `pa`. These parameters include the number of observations, the number of variables, the number of clusters to be searched for, the trimming level, the number of observations which are considered as to be outlying and not outlying, and other settings. Notice that these are the parameters that all iterations have in common and will never change.
3. **Variables initialization:**
   Several variables are initialized within a structure named `iter`, which will be used to track the results of subsequent iterations. These variables include the current objective function value, cluster assignments, cluster weights, cluster covariance matrices, cluster centers, and more. Notice that these variables change with each

iteration; in other words, the `iter` object is passed to all the subfunctions, it is modified, and it is returned by them. Furthermore, another object called `best.iter` is initialized, having the same structure as the `iter` object. It will be used to keep track of the best initialization during the optimization process.

4. **Starting of multiple initializations:**
   It performs a series of initializations (`nstart`) to search for the best clustering configuration. For each initialization, an initial set of clusters is initialized, finding their centers and covariance matrices using the `.InitClusters` function.

5. **Iterate for convergence:**
   Within each initialization, a series of iterations are performed until convergence or until the maximum number of available iterations (`iter.max`) is reached. During each iteration, the following actions are performed:

   - Assignment of data points to clusters using the `.findClustAssig` function.
   - Computation of the objective function value using the `.calcobj` function.
   - Estimation of clusters parameters (centers and covariance matrices) using the `.estimClustPar` function.

6. **Evaluation of initializations:**
   At the end of each initialization, it evaluates whether the current initialization has produced a higher objective function than the one of the best initialization so far. If so, the best initialization is updated.

7. **Return of results:**
   At the end of all initializations, it returns the results of the best initialization, including the estimated cluster parameters and other relevant information. In detail, this function returns as output an S3 object of type `tclust` [11], containing the following values:

   - `centers`: matrix of size p x k containing the centers (column-wise) of each cluster.
   - `cov`: array of size p x p x k containing the covariance matrices of each cluster.
   - `cluster`: numerical vector of size n containing the cluster assignment for each observation. Cluster names are represented as integer numbers ranging from 1 to k, while the value 0 is used to indicate trimmed observations.
   - `par`: list containing the parameters the algorithm has been called with.

- `num_init`: best initialization number.
- `loglik_vec`: numerical vector containing objective function values for each iteration of the best initialization.
- `obj`: final objective function value of the best initialization. It corresponds to the last value contained in `loglik_vec`.
- `size`: integer vector of length k, returning the number of observations belonging to each cluster.
- `weights`: numerical vector of length k, containing the weights of each cluster.

The algorithm is now complete and well-functioning. However, there is one final issue to address. The concern is that if we use the MRCD methodology to estimate covariance matrices within the M-step of our algorithm, we are no longer maximizing the objective function of TCLUST. Indeed, this maximization would only occur if we employ the sample covariance matrix to estimate the k covariance matrices, which we are no longer doing. So, if we cannot define the overall objective function, we are essentially proposing the solution to a problem that we have not formally defined. Therefore, our next step should involve creating a new objective function that incorporates the individual objective functions of the MRCD methodology for each cluster. The idea is that, during the M-step, instead of maximizing the function (1.1), we would have k regularized estimates, one for each component, as in (2.4). But this approach makes sense only if the MRCD estimation problem can be reformulated in terms of likelihood. In other words, if we can recast the objective function of the MRCD methodology in a likelihood framework, we would be able to express the objective function of our algorithm as the product of k penalized likelihoods, or even better, as the sum of k penalized log-likelihoods, one for each cluster. This reformulation would result in our methodology becoming likelihood-based.

We will delve deeper into this possibility in the upcoming section.

3.2. Can the MRCD estimator be reformulated in terms of likelihood?

We start by analyzing the main differences between likelihood-based (or model-based) methodologies and heuristic ones.

Likelihood-based methodologies:

1. They are based on the concepts of probability and statistics. Specifically, they are founded on probabilistic models that accurately and formally represent relationships between data and variables.
2. They require clear assumptions to be made about the underlying probabilistic distributions of the data. The assumptions are explicitly specified in the model.
3. The primary goal is to estimate model parameters that maximize the likelihood of the observed data. This approach seeks to find parameters that make the observed data most likely according to the probabilistic model.
4. Results can be interpreted in statistical and probabilistic terms. Confidence intervals, hypothesis tests and assessments of model fit to the data can be computed.
5. They can be computationally more intensive and require a deep understanding of the underlying statistical model.

Heuristic methodologies:

1. They rely on approximate rules and pragmatic strategies to solve complex problems. In particular, they do not necessarily rely on formal models or probabilistic distributions.
2. They often avoid complex assumptions and focus on practical and quick solutions.
3. The primary goal is to find solutions that are "good enough" for the problem, without necessarily ensuring the best possible solution. In other words they tend to provide satisfactory solutions in reasonable time but do not guarantee global optimum search. They may lead to local optimizations rather than global ones.
4. As they focus on efficiently solving problems, they might not provide statistical or probabilistic interpretations of the results.
5. They can be employed when computational complexity is a concern. Indeed, they

are often quicker to implement and can handle large-scale problems without demanding excessive computational resources.

Summing up, likelihood-based methodologies exploit the principles of probability and statistics, using formal probabilistic models to accurately capture the relationships within data. They often involve making clear assumptions about underlying distributions and aim to estimate model parameters that maximize the likelihood of observed data. As a result, they provide interpretable results with potential for statistical inference. On the other hand, heuristic methodologies take a pragmatic approach to tackling complex problems. Rather than relying on formal models or precise distributions, they employ approximate rules and practical strategies. Heuristic approaches prioritize finding "good enough" solutions efficiently, often without guaranteeing optimality, and excel in adaptability and speed, making them well-suited for large-scale problems where exact solutions might be computationally unfeasible.

Now, let us proceed by ascertaining whether the MRCD estimation problem can be reformulated in terms of likelihood.

The likelihood of a d-variate Gaussian distribution is

$$\phi(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\mathrm{MD}^2(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma})/2},$$

where $\boldsymbol{\mu}$ is the mean vector, $\boldsymbol{\Sigma}$ is the covariance matrix, and the Mahalanobis distance is $\mathrm{MD}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$.

For a sample $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$ we put $L(\boldsymbol{x}_i;\boldsymbol{\mu},\boldsymbol{\Sigma}) := -2\ln(\phi(\boldsymbol{x}_i;\boldsymbol{\mu},\boldsymbol{\Sigma}))$, so the maximum likelihood estimator (MLE) of $(\boldsymbol{\mu},\boldsymbol{\Sigma})$ minimizes

$$\sum_{i=1}^{n} L(\boldsymbol{x}_i;\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sum_{i=1}^{n} \left(\ln|\boldsymbol{\Sigma}| + d\ln(2\pi) + \mathrm{MD}^2(\boldsymbol{x}_i;\boldsymbol{\mu},\boldsymbol{\Sigma})\right). \tag{3.1}$$

Let us now look for a subset $H \subset \{1,\ldots,n\}$ with h elements which minimizes (3.1) where the sum is only over i in H. We can also write this with weights w_i that are 0 or 1 in the objective $\sum_{i=1}^{n} w_i L(\boldsymbol{x}_i;\boldsymbol{\mu},\boldsymbol{\Sigma})$, so we minimize

$$\sum_{i=1}^{n} w_i \left(\ln|\boldsymbol{\Sigma}| + d\ln(2\pi) + \mathrm{MD}^2\left(\boldsymbol{x}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)\right)$$
$$\text{under the constraint that} \sum_{i=1}^{n} w_i = h.$$

For the moment, we are proceeding as for the proof that MCD can be reformulated in terms of likelihood from [16]. However, we are not considering the fact that we use a regularized covariance matrix. So, to take it into consideration, we need to replace $\boldsymbol{\Sigma}$ with its regularized version $\mathbf{K} = \rho\mathbf{T} + (1-\rho)c_\alpha\boldsymbol{\Sigma}$, where $\mathbf{T}$ is the target matrix, ρ is the regularization parameter, and c_α is the costincency factor, all already been discussed in section 2.1. Then the previous problem becomes:

$$\begin{aligned}&\sum_{i=1}^{n} w_i \left(\ln|\mathbf{K}| + d\ln(2\pi) + \mathrm{MD}^2\left(\boldsymbol{x}_i; \boldsymbol{\mu}, \mathbf{K}\right)\right)\\ &\text{under the constraint that} \sum_{i=1}^{n} w_i = h.\end{aligned} \tag{3.2}$$

For the minimizing set of weights w_i we know from maximum likelihood that $\widehat{\boldsymbol{\mu}}_{MLE}$ is the mean of the $\boldsymbol{x}_i$ in H, so it is the weighted mean of all $\boldsymbol{x}_i$, and similarly

$$\widehat{\boldsymbol{\Sigma}}_{MLE} = \frac{1}{h}\sum_{i=1}^{n} w_i \left(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{MLE}\right)\left(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{MLE}\right)^T.$$

Therefore, being that

$$\widehat{\mathbf{K}}_{MLE} = \rho\mathbf{T} + (1-\rho)c_\alpha\widehat{\boldsymbol{\Sigma}}_{MLE},$$

we obtain the following expression for $\widehat{\mathbf{K}}_{MLE}$:

$$\widehat{\mathbf{K}}_{MLE} = \rho\mathbf{T} + (1-\rho)c_\alpha\left(\frac{1}{h}\sum_{i=1}^{n} w_i \left(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{MLE}\right)\left(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{MLE}\right)^T\right). \tag{3.3}$$

Since the second term of (3.2) is constant, to prove that minimizing (3.2) is equivalent to minimizing the determinant of (3.3), we need to prove that the third term of (3.2) is also constant.

The third term of (3.2) becomes:

$$\begin{aligned}
&\sum_{i=1}^{n} w_i \,\mathrm{MD}^2\left(\boldsymbol{x}_i; \widehat{\boldsymbol{\mu}}_{MLE}, \widehat{\mathbf{K}}_{MLE}\right) \\
&= \sum_{i=1}^{n} w_i \left(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{MLE}\right)^T \widehat{\mathbf{K}}_{MLE}^{-1} \left(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{MLE}\right) \\
&= \sum_{i=1}^{n} \mathrm{trace}\left(w_i \left(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{MLE}\right)\left(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{MLE}\right)^T \widehat{\mathbf{K}}_{MLE}^{-1}\right) \\
&= \mathrm{trace}\left(\sum_{i=1}^{n} w_i \left(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{MLE}\right)\left(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_{MLE}\right)^T \widehat{\mathbf{K}}_{MLE}^{-1}\right) \\
&= \mathrm{trace}\left(h\widehat{\boldsymbol{\Sigma}}_{MLE}\widehat{\mathbf{K}}_{MLE}^{-1}\right).
\end{aligned}$$

But, remembering that

$$\widehat{\boldsymbol{\Sigma}}_{MLE} = \frac{\widehat{\mathbf{K}}_{MLE} - \rho\mathbf{T}}{(1-\rho)c_\alpha},$$

we get the following:

$$\mathrm{trace}\left(h\widehat{\boldsymbol{\Sigma}}_{MLE}\widehat{\mathbf{K}}_{MLE}^{-1}\right) = h \cdot \mathrm{trace}\left(\frac{\widehat{\mathbf{K}}_{MLE} - \rho\mathbf{T}}{(1-\rho)c_\alpha} \cdot \widehat{\mathbf{K}}_{MLE}^{-1}\right) =$$
$$\frac{h}{(1-\rho)c_\alpha} \cdot \mathrm{trace}\left(\widehat{\mathbf{K}}_{MLE}\widehat{\mathbf{K}}_{MLE}^{-1} - \rho\mathbf{T}\widehat{\mathbf{K}}_{MLE}^{-1}\right).$$

Now, it's easy to notice that

$$\widehat{\mathbf{K}}_{MLE}\widehat{\mathbf{K}}_{MLE}^{-1} = \boldsymbol{I}_d.$$

Therefore, recalling that

$$\mathrm{trace}\left(\boldsymbol{I}_d\right) = d$$

and that the trace of the sum of matrices is equal to the sum of the traces of those matrices, we finally obtain:

$$\sum_{i=1}^{n} w_i \,\mathrm{MD}^2\left(\boldsymbol{x}_i; \widehat{\boldsymbol{\mu}}_{MLE}, \widehat{\mathbf{K}}_{MLE}\right) = \frac{hd}{(1-\rho)c_\alpha} - \frac{h\rho \cdot \mathrm{trace}\left(\mathbf{T}\widehat{\mathbf{K}}_{MLE}^{-1}\right)}{(1-\rho)c_\alpha}. \tag{3.4}$$

The first term (3.4) is constant, whereas the second one varies depending on $\mathbf{T}\widehat{\mathbf{K}}_{MLE}^{-1}$. Consequently, the third term in equation (3.2) is not constant. Therefore, we cannot assert that minimizing (3.2) is equivalent to minimizing the determinant of (3.3). This implies that the MRCD estimation problem cannot be reformulated in terms of likelihood of a Gaussian distribution.

Our proposed algorithm is, thus, not a likelihood-based method; rather, it is a heuristic one. Being such a method, we have the autonomy to select an optimal criterion to assess the best clustering result. In this context, we choose to retain the original criterion of the TCLUST methodology, which involves searching for the clustering configuration that maximizes the objective function (1.1).

We finally end up with a well-performing, but heuristic, method. Actually, the initial goal was to find a robust clustering method that was both well-performing in high-dimensional settings and likelihood-based. For this reason, in the next section, we will introduce another algorithm, once again building upon the TCLUST framework, but this time incorporating a well-conditioned covariance matrix estimator that is model-based (specifically, Gaussian-based), known as CovGlasso.

3.3. A likelihood-based methodology: incorporating the CovGlasso estimator within TCLUST

We now aim to incorporate the CovGlasso estimator, previously introduced in Section 2.3, into the TCLUST algorithm. This effort seeks to address the ongoing challenge of robust clustering for high-dimensional data while simultaneously developing a likelihood-based methodology. Indeed, during the construction of the CovGlasso estimator, Gaussianity is assumed, making it a model-based approach, specifically built upon the Gaussian model. Considering that TCLUST also operates within the Gaussian framework, the methodology we are set to develop, arising from the integration of the CovGlasso estimator into TCLUST, will similarly be rooted in Gaussian framework. More precisely, we can formulate the objective function for our problem as the summation of k minus penalized log-likelihoods, each representing the individual objective function (2.9) of the CovGlasso

methodology for each cluster:

$$\sum_{i=1}^{k} \left(\log \left(\det \left(\widehat{\boldsymbol{\Sigma}}_i \right) \right) + \operatorname{tr} \left(\widehat{\boldsymbol{\Sigma}}_i^{-1} \boldsymbol{S}_i \right) + \lambda \left\| \boldsymbol{P} * \widehat{\boldsymbol{\Sigma}}_i \right\|_1 \right), \tag{3.5}$$

where $\widehat{\boldsymbol{\Sigma}}_i$ and $\boldsymbol{S}_i$ represent the CovGlasso estimator and the sample covariance matrix estimator for the i-th cluster, respectively. The objective function (3.5) of our new methodology needs to be collectively minimized, as it is the sum of k CovGlasso objective functions, each requiring minimization.

Let us start with the `.InitClusters` subfunction. Our initial step involves replacing the MRCD estimator with the new CovGlasso estimator, leaving the rest of the function structured as before.

The CovGlasso estimator is computed using the `covglasso` function of the `covglasso` package itself [7]. It estimates a sparse covariance matrix by minimizing the expression (2.9), specifically using a fast coordinate descent algorithm to solve the covariance graphical lasso. Notice that a sparse covariance matrix is characterized by a significant portion of elements being zero or close to zero. In other words, many of the variables have weak or negligible relationships with each other. The main goal of a sparse estimator is to reduce computational complexity and improve the stability of covariance matrix estimates by avoiding the calculation and consideration of all matrix elements.

There is a problem: in high-dimensional settings, the sample covariance matrix, just like the covariance matrix, will not be full rank, resulting in a degenerate solution. In this case, we need to set $\boldsymbol{S} = \boldsymbol{S} + \epsilon \boldsymbol{I}_p$, for some $\epsilon > 0$, thus regularizing the sample covariance matrix. This procedure has not been directly implemented within `covglasso`; therefore, we need to apply it before recalling the function, allowing it to work without any issue.

In detail, we update `.InitClusters` as follows: while iterating through the cluster matrices, we begin by computing and recording the number of rows and columns for each matrix. Next, we calculate the cluster center using the sample mean estimator. Subsequently, the sample covariance matrix is computed, and regularization is applied if the number of columns equals or exceeds the number of rows. Finally, we recall `covglasso` to perform the CovGlasso estimation, saving the resulting estimated covariance matrix of each cluster in the `sigma` array within the `iter` object, which is then returned as final step.

At this point, we would have the `.InitClusters` subfunction arranged and ready to be recalled within the main function during the initialization phase. However, we decide to

further enhance it by introducing another type of initial cluster assignment, which is more pertinent and precise than the previously employed random assignment.
This is the assignment made through the application of the TCLUST algorithm on a subset of the original variables. Specifically, we generate a vector containing 10 random indices from 1 to p, and then we extract the columns corresponding to those indices from the data matrix, thus obtaining a submatrix of observations composed of 10 randomly selected columns. Subsequently, we apply the `tclust` function to the newly created submatrix. For α, we use the same value provided as input to the main function. We set the constant c controlling the strength of the constraint (1.2) to 50 through trial and error. A lower value caused excessive restriction on the eigenvalues of scatter matrices, resulting in warnings and errors during algorithm execution. Finally, the result of the initial cluster assignments, which already includes the designation of $n\alpha$ trimmed observations as outliers, is stored in the corresponding vector of the `iter` object, and the subsequent computation of cluster weights is done as already explained above. It is important to note that, here as well, it is necessary to repeat the initialization procedure many times and choose the best result in order to avoid falling into local minima or suboptimal solutions.

Using TCLUST on a subset of variables from the data matrix for the initial assignment of observations to clusters, instead of employing a completely random initial assignment, offers some advantages:

1. **Improved initialization:**
   It can yield better initialization as it takes potential data structures into account.
2. **Computational efficiency:**
   It can make our final algorithm less computationally intensive, as it brings to a reduced number of iterations required at each initalization, given that we have a better starting point compared to the random one.
3. **Reduced randomness:**
   Complete random assignment may introduce a degree of randomness in initial cluster assignments, which could impact result stability and repeatability. Using TCLUST on a subset of variables might reduce this randomness.

Regarding the `.estimClustPar` subfunction, responsible for computing the mean vector and the covariance matrix for each provided cluster, the only segment requiring modification is the one internal to the loop that iterates through cluster matrices. We make this update by following the same approach carried out in the previous subfunction for the initial cluster parameter estimation. Furthermore, the constraints on the minimum cluster

size remain unchanged. So, if a cluster contains at least three data points, the CovGlasso estimation of covariance matrices is executed. However, if a cluster is empty or has less than three observations, the CovGlasso estimation is skipped, and subsequently, that initialization will not be passed for evaluation as the best initialization in the main function.

The `.findClustAssig` subfunction remains exactly the same, while the `.calcobj` subfunction, responsible for computing the objective function values for our problem, needs to be entirely rewritten. In particular, our objective function is defined as the summation of k minus penalized log-likelihoods, each representing the individual objective function (2.9) of the CovGlasso methodology for each cluster.

In detail, `.calcobj` works as follows:

1. It initializes the objective function value as zero.
2. It iterates through each cluster and, within each iteration, it checks if the determinant of the covariance matrix for the current cluster is non-zero, indicating that the matrix is invertible and well-defined.
3. If the determinant is non-zero, it computes the logarithm of the determinant, the trace term and the sparsity penalty term, which correspond to the three distinct components in expression (2.9).
4. The computed terms are added (by summation) to the ongoing objective function value.
5. Conversely, if the determinant is equal to zero, the objective function value is set to infinity, the loop breaks prematurely, and subsequently, this initialization will not be passed for evaluation as the best initialization in the main function.
6. The loop continues to iterate through the clusters unless it is stopped prematurely. Once the iteration is completed, the `iter` object, updated with the final objective function value, is returned.

The very last thing to do is to update the main function, called `.covglasso_in_tclust`. It maintains the same structure as the previously developed `.mrcd_in_tclust`, with one significant adjustment.
It concerns modifying the final condition for identifying the best initialization as follows: if the current initialization is the first one or if the current objective function value is lower than the objective function value of the best initialization encountered so far, the structure

of the best initialization is updated with the structure of the current initialization. This modification aims to minimize the objective function of our problem, ensuring that the vector storing the objective function values for each iteration of the best initialization decreases. Since the objective function (2.9) of the CovGlasso estimation problem required minimization, the same principle applies to the objective function of our new methodology, as it is the sum of k minus penalized log-likelihoods in the form of (2.9), which therefore need to be collectively minimized.

It is important to note that, for `.covglasso_in_tclust`, we introduce two additional input parameters compared to those of `.mrcd_in_tclust`. These parameters are λ, which is the lasso regularization parameter, and $\boldsymbol{P}$, which represents the lasso regularization matrix, both seen in expression (2.9). Typically, when applying the CovGlasso estimation, the most suitable λ is often determined through the use of BIC or EBIC within its algorithm execution. However, in our case, due to the iterative nature of the algorithm where CovGlasso is employed repeatedly, λ needs to be set beforehand and consistently maintained across all EM iterations. This ensures the constancy of λ in the objective function value computation throughout the iterative process.

At this point, we have two different approaches for high-dimensional robust clustering: a heuristic methodology and a likelihood-based one. Regarding the first methodology, in the next session we will introduce a final improvement to better address the issue of robustness to outliers.

3.4. An improvement to the heuristic methodology: replacing the MRCD estimator with the Ledoit-Wolf estimator

In the previously developed heuristic methodology, where the MRCD estimator has been incorporated into the TCLUST algorithm, a single issue arises: this approach presents a doubly-robust extension. Outliers are effectively addressed not only within the clustering procedure itself, but also through the modified M-step that makes use of MRCD to compute the regularized covariance matrices, exploiting its robust-based estimation.
Therefore, our intention is to replace the MRCD estimator with the linear shrinkage es-

timator proposed by Ledoit and Wolf. This estimator can be seen as a particular case of the former, where the subset H in equation (2.3) corresponds to the entire sample [3], and the target matrix $\mathbf{T}$ is the identity matrix. Moreover, unlike the MRCD estimation, during the shrinkage estimation of Ledoit-Wolf the data does not need to be initially standardized. The Ledoit-Wolf estimator lacks robustness to outliers, as it incorporates them in its computation. Nonetheless, considering its incorporation into the TCLUST algorithm, it becomes logical to use it, given that the robustness is already enforced by the TCLUST procedure.

We thus proceed by replacing `CovMrcd`, previously used for the MRCD estimation, with `linearShrinkLWEst`, responsible for linear shrinkage Ledoit-Wolf estimation, within the `.InitClusters` and `.estimClustPar` subfunctions.

The `linearShrinkLWEst` function, which is derived from the `cvCovEst` package [2], computes an asymptotically optimal convex combination of the sample covariance matrix and the identity matrix, effectively shrinking the eigenvalues of the sample covariance matrix towards the identity. It takes only the data matrix as input and provides an output matrix corresponding to the Ledoit-Wolf linear shrinkage estimate of the covariance matrix.

The main function, now renamed as `.LedoitWolf_in_tclust`, will retain the same structure as `.mrcd_in_tclust`, with the only change being the improvement of the initialization procedure, as done for CovGlasso in TCLUST.

Finally, we end up with two distinct algorithms for robust clustering on high-dimensional data: a heuristic methodology, which is implemented in `.LedoitWolf_in_tclust`, and a likelihood-based methodology, which is developed in `.covglasso_in_tclust`. We now want to test and compare our two methods, first on simulated data, that are easier to handle, and then on real data, aiming to solve a real-world problem. We will address these issues in the next chapter.

4 | Data analysis

In this chapter we will test and compare the two methodologies we previously developed for robust clustering in high-dimensional data scenarios. To comprehensively evaluate the performance of our algorithms, our initial approach involves their application to a simulated dataset, where the data are generated from multivariate Gaussian distributions. Subsequently, we expand our analysis to real-world datasets, with a specific focus on addressing the challenge of recognizing handwritten digits. Our primary objective is to validate the effectiveness of our algorithms in efficiently grouping digits into meaningful clusters, while also accurately identifying those marked as outliers.

4.1. Simulated data

For the evaluation of our algorithms on a simulated dataset, we generate synthetic data, simulating a three-component mixture distribution, with each component modeled as a Gaussian distribution. Additionally, we introduce outliers by using a separate Gaussian component.

We first define the dimensionality of our data, opting for a high-dimensional setting with 50 features. This choice introduces complexity into the data, making the scenario more similar to a real-world one. We then decide to use 95 non-trimmed observations, and we need to determine how many observations come from each distribution. For this purpose, we exploit the multinomial function, providing as input the desired number of random samples to be generated (95) and a vector of three components (0.4, 0.3 and 0.3), each representing the probability of originating from a specific distribution. This function generates a matrix with 3 rows and 95 columns, where each element is 1 if the i-th observation originates from the j-th distribution, and 0 otherwise, based on the probability vector provided as input. The function that sums by row is then employed to compute the sum of elements in each row of the matrix. It is important to note that

the sum of elements in a row corresponds to the number of observations originating from the distribution associated to that row of the matrix. Therefore, we ultimately obtain a new vector with three components, indicating how many times each distribution is used to generate the observations.

Our approach involves the employment of different mean vectors for each distribution, while maintaining the use of the same covariance matrix across all of them. This strategic choice results in the three distributions sharing the same shape, indicating an identical data dispersion structure. However, due to the difference in mean vectors, they have distinct locations within the feature space. Specifically, we are adopting mean vectors for the three distributions where the values of their first two components are closely situated: 1 and 2 for the first distribution, 3 and 4 for the second, and 5 and 6 for the third. As a shared covariance matrix for all distributions, we are selecting a diagonal matrix with 50 equidistant values ranging from 0.1 to 1. Similarly, to generate the 5 outliers to be added to our data, we employ another Gaussian distribution. This approach enables us to achieve a clear visualization of our simulated data in the first two dimensions of the feature space. Indeed, it results in the generated data points from the distinct distributions being closely situated and distinctly separated from each other, as shown in the scatter plot presented in Figure 4.1.

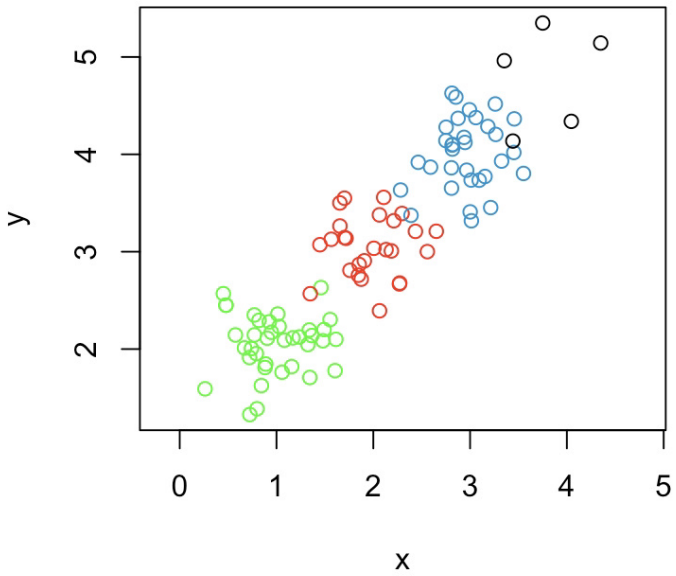


Figure 4.1: Simulated data in the first two dimensions of the feature space.

The scatter plot in Figure 4.1 illustrates the simulated data points in the first two dimensions of the feature space. Each data point is colored according to its group membership:

green for observations originating from the first distribution, red for those from the second distribution, blue for those from the third distribution, and black for outliers.

At this point, we are ready to apply our two robust high-dimensional clustering algorithms to the just generated and visualized simulated data. We accomplish this by invoking the previously implemented functions `.LedoitWolf_in_tclust` and `.covglasso_in_tclust`.

Here are the shared input parameters for both functions:

- Dataset composed of our simulated data, created combining the observations generated from the four distinct Gaussian distributions by rows.
- Number of clusters we are searching for. We therefore set $k = 3$.
- Proportion of trimmed observations. Since we have 5 outliers out of a total of 100 observations, we set $\alpha = 0.05$.
- Number of random initialization to be performed. We initially decide to use 30 random initializations. However, subsequent empirical investigations reveal the computational efficiency of the algorithm, prompting us to increase its value to 50. This decision is indeed justified, as a higher value allows for a more precise exploration of various initial conditions, thereby increasing the probability of identifying the optimal clustering configuration. However, it is important to note that, during the transition to real-world scenarios, computational demand increases, necessitating a reduction in its value to maintain computational feasibility.
- Maximum number of concentration steps to be performed. At first, we choose a relatively high value of 20. This decision is made considering that concentration steps cease if two consecutive steps result in the same data partition, even before reaching this value. However, further empirical investigations demonstrate that the algorithm primarily converges within around ten iterations, rendering a higher value unnecessary. Therefore, we adjust it to 10.

However, `.covglasso_in_tclust` has two additional input parameters:

- Lasso regularization parameter. We select $\lambda = 8$ using trial-and-error experimentation.
- Lasso regularization matrix. We opt for an all-ones matrix, which equalizes the penalty across all elements of the covariance matrix. In other words, by employing this type of regularization matrix, we implicitly apply the same level of regularization to every element of the covariance matrix, thus encouraging a sparser pattern.

In our evaluation, we find out that both algorithms achieve a remarkable 100% accuracy rate in identifying outliers. This demonstrates the robustness of our methodologies in distinguishing anomalous data points from the primary clusters, which will be a crucial aspect in real-world scenarios.
Furthermore, the assignment of data points to their respective clusters is consistently accurate across both algorithms. Indeed, in every test scenario, each data point is precisely assigned to the appropriate cluster without error. These precise cluster assignments underscore the ability of our methodologies to recognize underlying patterns and groupings within the data, even in presence of noise and outlying units.

Finally, we are particularly interested in the resulting clustering configurations.

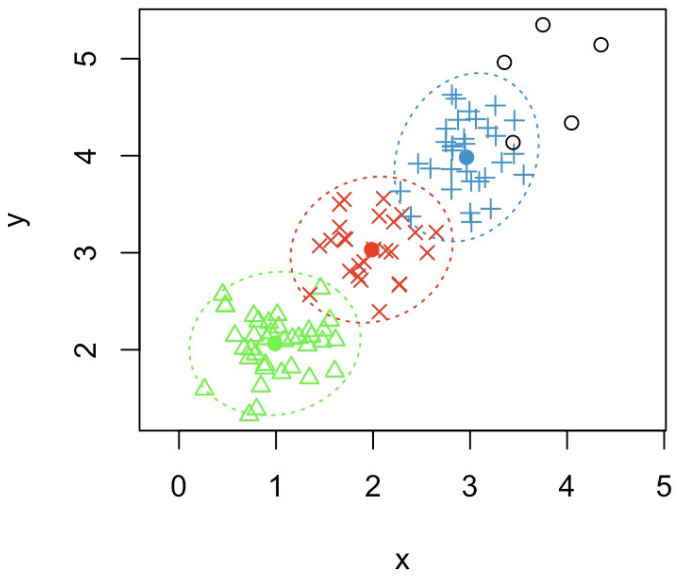


Figure 4.2: Clustering configuration achieved by Ledoit-Wolf in TCLUST.

In Figure 4.2, we provide a visual representation of the clustering configuaration accomplished by Ledoit-Wolf in TCLUST. Each data point is represented and colored based on its cluster membership: green plus signs for observations assigned to the first cluster, red triangles for those assigned to the second cluster, blue crosses for those assigned to the third cluster, and black circles for observations identified as outliers. It is important to note that each cluster assumes an elliptical form.

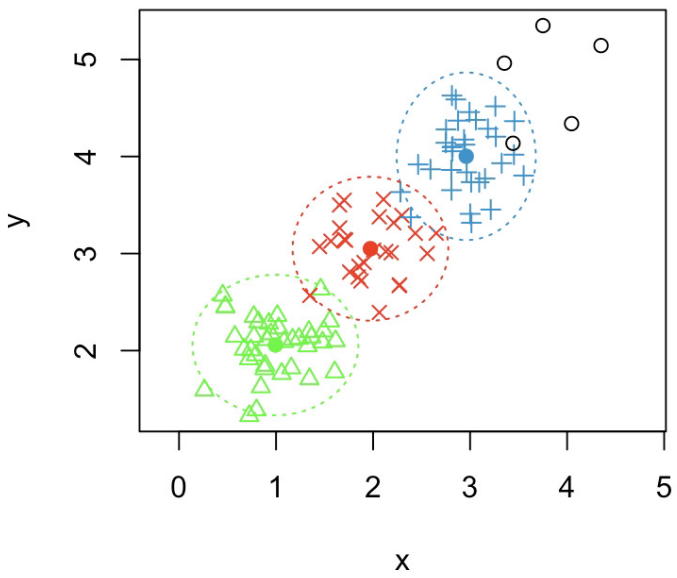


Figure 4.3: Clustering configuration achieved by CovGlasso in TCLUST.

In Figure 4.3, we show the clustering configuration obtained using CovGlasso in TCLUST. In this visualization, each data point is assigned to a specific elliptical-shaped cluster, and is color-coded based on its cluster membership, as previously described.

In summary, the evaluation of our robust clustering algorithms on simulated data showcases their exceptional performance. Each data point is precisely assigned to the cluster corresponding to its original distribution, while anomalous units are accurately identified as outliers, resulting in an outstanding 100% accuracy for both cluster assignments and outliers detection. The visual representations in Figures 4.2 and 4.3 depict the same estimated clusters, with slightly different elliptical shapes, due to the distinct covariance matrix estimation techniques employed by each algorithm.

These outcomes underscore the robustness and effectiveness of our methodologies when dealing with high-dimensional simulated data. Now, our focus shifts to real-world data, which poses greater challenges due to their intricacies. Specifically, in the upcoming section, we will address the problem of recognizing handwritten digits, which is very challenging due to the high dimensionality of the data, the limited separation between classes and the potential presence of outlying units.

4.2. Real-world data: the handwritten digits recognition problem

In this section, we explore the application of our robust clustering algorithms to real-world high-dimensional data. We specifically focus on two datasets designed for recognizing handwritten digits, which are subsets of the well-known USPS dataset available through the UCI Machine Learning Repository. The original one includes 7291 images of handwritten digits categorized into 10 classes representing digits from 0 to 9.

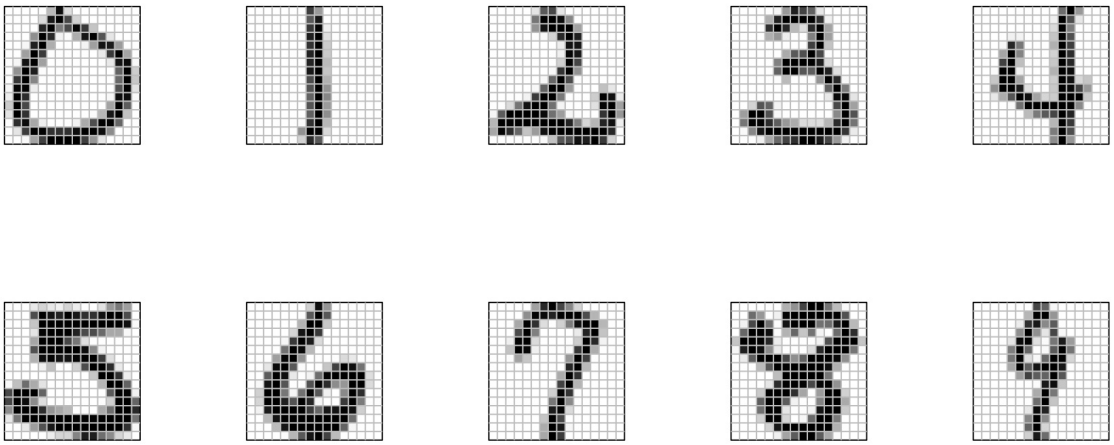


Figure 4.4: Visual representation of digits 0 to 9 from the handwritten digits dataset.

Every image is partitioned by a 16×16 grid, resulting in 256 total squares. Each square then represents a distinct variable in the dataset, capturing the pixel count of that specific portion of the image. In particular, variables assume values spanning from 0 to 1. A value of 0 denotes an entirely white pixel, while a value of 1 signifies a completely black pixel. Intermediate values encompass a spectrum of gray shades, ranging from very pale gray, almost white, for values close to 0, to very deep grey, nearly black, for values near 1.

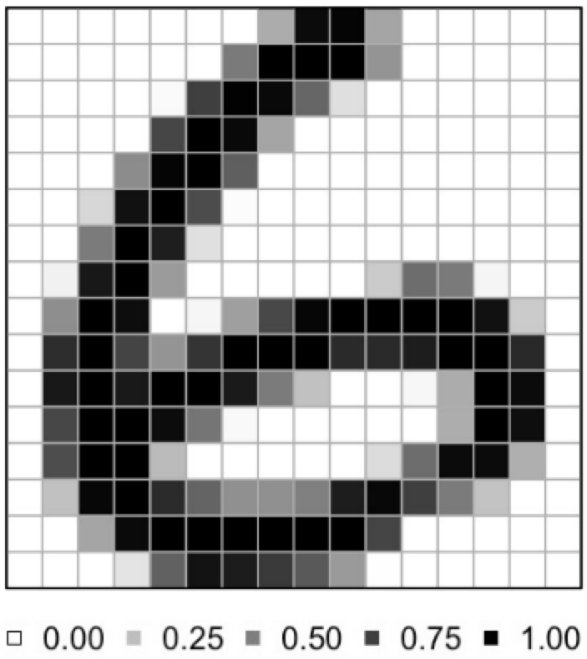


Figure 4.5: Visual representation of a randomly selected handwritten digit labeled as 6.

In Figure 4.5, we can observe a randomly selected handwritten digit, specifically labeled as 6. Each of the 256 squares is filled with its corresponding color, which may be white, black or a distinct shade of gray, as illustrated in the legend.

We start our analysis of the USPS dataset by plotting the multivariate means for all the digits. This initial step aims to identify both the groups of digits exhibiting the highest similarities among themselves and groups of digits that are more easily distinguishable. Our goal is to apply our algorithms to two distinct subsets of the original USPS dataset. One subset comprises groups of digits that are clearly distinguishable from one another, while the other subset consists of groups of digits that are more similar to each other. This approach allows us to assess the robustness and effectiveness of our algorithms for both less and highly complex real-world scenarios. In the more complex one, in addition to the challenges of high-dimensional data and presence of outliers, there arises also the issue of limited separation between classes.

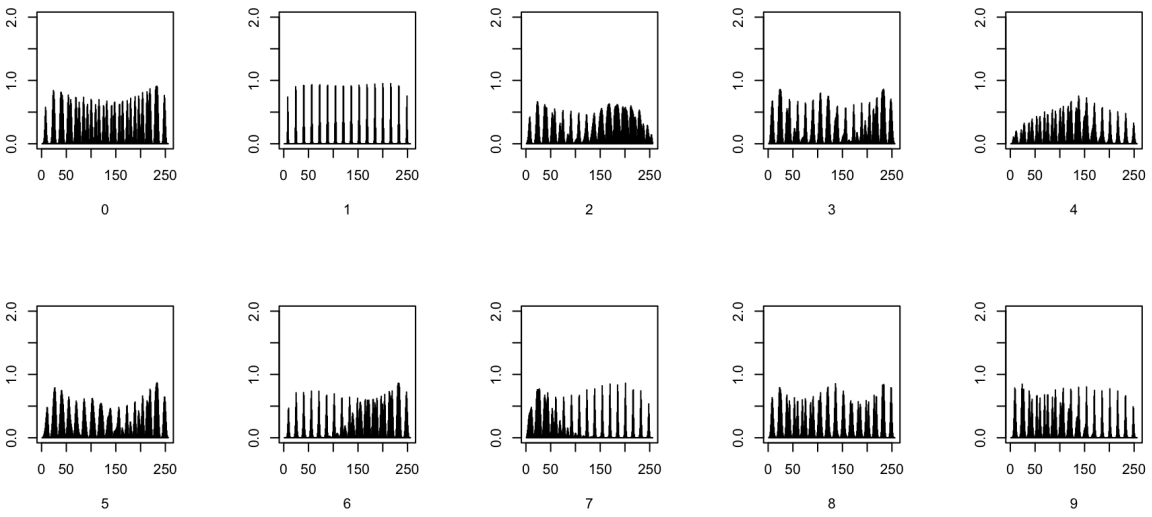


Figure 4.6: Multivariate means for all digits within the USPS dataset.

It is evident from Figure 4.6 that digits 0, 1 and 4 exhibit clearly distinct behaviors, each diverging significantly from the others, while digits 3, 5 and 8 display remarkably similar multivariate means, as also stated in [4].

Firstly, we focus our analysis on digits 0, 1 and 4, which, based on the aforementioned observations, are expected to be less complex for clustering. Subsequently, we investigate digits 3, 5 and 8, which present a more challenging clustering task due to their closely aligned multivariate means. This resemblance arises from the inherent similarity in the handwriting of these three digits, as displayed in Figure 4.7.

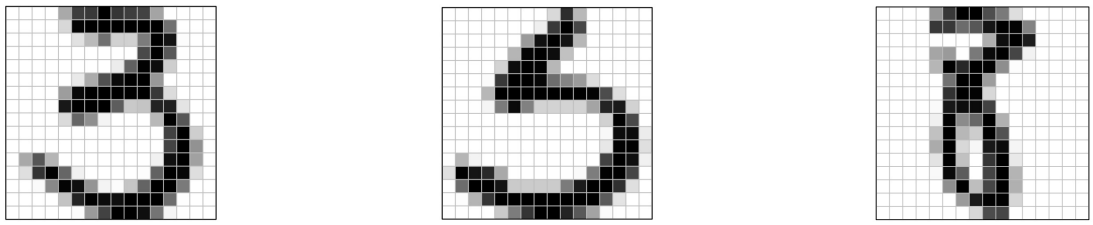


Figure 4.7: Visual representation of similar handwritten digits labeled as 3, 5 and 8.

In detail, our approach involves applying our two methodologies, Ledoit-Wolf in TCLUST and CovGlasso in TCLUST, to two separate datasets of handwritten digits. One dataset includes digits 0, 1 and 4, while the other comprises digits 3, 5 and 8. Our primary objective is to evaluate the robustness and effectiveness of these algorithms in addressing the challenges presented by each dataset. Furthermore, a comparative analysis of the outcomes generated by these two algorithms will be conducted to identify any performance disparities.

We start the analysis with the digits 0, 1 and 4. We create a dataset by randomly selecting 50 data points from each of these digit subsets, while also including 5 outliers from the subset of digits distinct from the previous ones.
Given the intricate high-dimensional nature of the data, it is imperative to carefully manage the feature space our algorithms operate within. To mitigate the potential adverse effects of uninformative features, we perform a variable selection step aimed at reducing the dimensionality of our dataset by removing irrelevant features. This preliminary selection involves identifying variables with variances exceeding a specific threshold. We determine this threshold through a process of trial and error, ultimately setting it at 0.50, which results in retaining around 130 features out of the total 256.
We generate a scatter plot to visualize the two-dimensional representation of the data points, colored according to their respective labels. By experimenting with various pairs of variables, it becomes apparent that representing data in only two dimensions does not yield clear insights due to visual limitations. Indeed, data points of different colors are not distinctly separated as was the case with the simulated data; instead, they appear paired and mixed with each other. It is worth noting that this will be even more evident when considering the digits 3, 5 and 8, due to their high similarity and, consequently, their limited separation.

To evaluate the quality of the results obtained from applying our robust clustering methodologies, we will use two similarity measures between the estimated labels and the true labels of the digits. These measures are:

- **Overall accuracy:** it is computed as the sum of correct matches divided by the total number of observations, which is 155 in our case (we are also including the outliers in the evaluation). The overall accuracy of the clustering model is computed as the sum of correct matches divided by the total number of data points, quantifying the degree of correspondence between the estimated and the true labels.

- **Adjusted Rand Index (ARI):** it is a validity index used to evaluate the similarity between two clustering assignments, such as the labels estimated by a clustering algorithm and the true labels. This index takes into account all pairs of samples and assesses how well the clustering assignments align with the true labels. The ARI ranges from -1 to 1, with the following interpretations:
  - A value of 1 indicates a perfect match between the clustering assignments and the true labels.
  - A value of 0 suggests that the assignments are, on average, no better than random assignments.
  - A value of -1 indicates that the clustering assignments are completely discordant with the true labels.

  The term "adjusted" in ARI reflects its accounting for random similarities that naturally occur in clustering assignments. This attribute makes ARI a more reliable index compared to unadjusted measures, especially in situations with numerous clusters. In summary, the Adjusted Rand Index offers a way to assess how effectively a clustering algorithm has assigned data into clusters compared to the reference labels. A positive value indicates a degree of consistency between the estimated and the true labels, while negative values signify that the clustering assignments are less effective than random assignments.

Below, we provide the two contingency tables, displaying comparisons between the cluster labels estimated by Ledoit-Wolf in TCLUST and the true labels (table on the left), as well as between the cluster labels estimated by CovGlasso in TCLUST and the true labels (table on the right).

group	0	1	4	out
0	45	3	2	0
1	0	50	0	0
4	1	9	40	0
out	0	0	0	5

(a) Ledoit-Wolf in TCLUST

group	0	1	4	out
0	48	0	2	0
1	0	49	1	0
4	0	2	48	0
out	0	0	0	5

(b) CovGlasso in TCLUST

Table 4.1: Contingency tables for comparisons between estimated cluster labels and true labels (digits 0, 1 and 4).

Notably, both methodologies excel in identifying the five anomalous units present in the processed dataset, accurately labeling them as outliers, as shown in Table 4.1. This showcases their robustness in detecting unusual instances within the data.
In terms of performance, both algorithms exhibit highly satisfactory clustering results. Using Ledoit-Wolf in TCLUST yields an overall accuracy of 90.3% and ARI of 0.729. Impressively, the CovGlasso in TCLUST achieves even better performance, boasting an overall accuracy of 96.8% and ARI of 0.905.
In conclusion, these remarkably high metrics values affirm the capability of our algorithms in accurately grouping digits into clusters aligned with the true labels, while also effectively addressing the presence of outliers. This accomplishment underscores their capacity to tackle the challenge of robust clustering in the context of high-dimensional real-world data. Notice that such a result would not have been achievable using TCLUST, due to the high dimensionality of the data.

We will now analyze the outcomes obtained by employing our two robust clustering methodologies in addressing a more intricate high-dimensional real-world data challenge. Specifically, we focus on digits 3, 5 and 8, which, as previously demonstrated, display notably similar behaviors. This similarity poses a heightened challenge to robust clustering due to their limited distinction. We follow the same approach as in the previous scenario, creating a dataset consisting of a total of 155 data points. Out of these, 50 data points are extracted from each subset representing digits 3, 5 and 8. Additionally, 5 data points are randomly sampled from all the other digits, distinct from the aforementioned ones. Subsequently, we establish a variance threshold (consistently set at 0.50), and only retain features exceeding this threshold. As before, this results in approximately 130 features. We then apply our two algorithms to this refined dataset, employing identical input parameter values as in the previous case.

group	3	5	8	out
3	29	9	12	0
5	16	29	5	0
8	9	11	30	0
out	0	0	0	5

(a) Ledoit-Wolf in TCLUST

group	3	5	8	out
3	48	2	0	0
5	29	19	2	0
8	4	10	36	0
out	0	0	0	5

(b) CovGlasso in TCLUST

Table 4.2: Contingency tables for comparisons between estimated cluster labels and true labels (digits 3, 5 and 8).

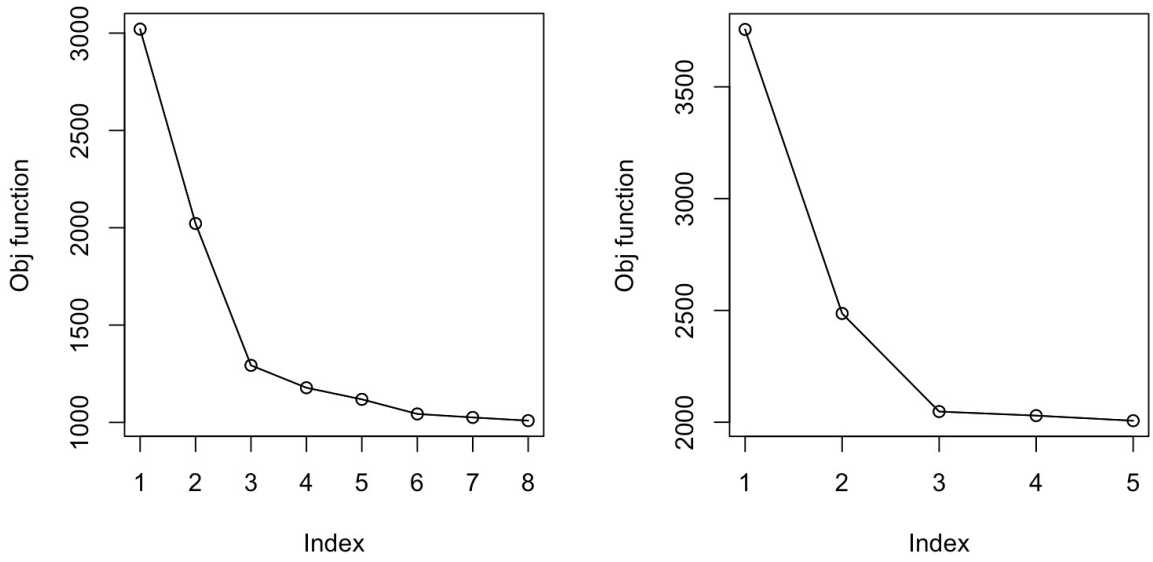
Table 4.2 illustrates two contingency tables that present comparisons between the cluster labels estimated by Ledoit-Wolf in TCLUST and the true labels (left table), as well as between the cluster labels estimated by CovGlasso in TCLUST and the true labels (right table). The first algorithm achieves an overall accuracy of 60.0% (correctly classifying 93 data points out of a total of 155) with ARI of 0.172. Differently, using the second one results in an accuracy of 69.7% (108 data points correctly classified out of 155) and ARI of 0.385. Notably, both algorithms continue to accurately detect all five outlying units, underscoring their robustness in detecting unusual instances within the dataset.

In terms of performance, the results obtained with CovGlasso in TCLUST are better than those achieved with Ledoit-Wolf in TCLUST, as they have achieved higher overall accuracy and ARI. These outcomes align with our initial assumptions, especially on the nature of the algorithms. Specifically, the CovGlasso in TCLUST methodology, which is grounded in a Gaussian framework, was expected to yield better results compared to Ledoit-Wolf in TCLUST, an heuristic methodology not rooted in any statistical framework. This expectation is motivated by the fact that likelihood-based methodologies have the ability to more effectively capture the underlying distributional characteristics and intricacies of the data. This advantage is particularly crucial when dealing with complex, high-dimensional data, such as that representing handwritten digits.

The performance of CovGlasso in TCLUST are satisfactory, given the high complexity of the problem we are tackling. This complexity arises from both the high dimensionality of the data and the presence of anomalous units, as well as the substantial similarity among the three considered classes. Specifically, examining the right outcome in Table 4.2, which represents the matching counts for the CovGlasso in TCLUST methodology, i.e. the better performing one, we initially observe that the true 3's are mostly classified correctly. Indeed, out of 50, 48 are accurately estimated as 3's. Similarly, the true 8's are also well classified, as 36 out of 50 are correctly estimated as 8's. The 5 anomalous units, as previously mentioned, are correctly identified as outliers. However, an issue arises with the 5's. In fact, the true 5's are mostly assigned to the estimated cluster of 3's, with 29 out of 50 being estimated as 3's, while only 19 estimated as 5's. This occurs because, as also observed in Figure 4.7, many observations labeled as 3's and 5's are virtually interchangeable due to their almost identical handwritten forms. Consequently, the algorithm struggle in distinguishing between these two digits for samples showing an absolute similarity. Nevertheless, beyond this, they effectively address the other challenges described, confirming their robustness and efficiency in clustering high-dimensional real-world data with the potential presence of outliers.

To conclude, let us go deeper into the final results obtained with CovGlasso in TCLUST, which is the methodology yielding the best outcomes in both clustering digits 0, 1 and 4, as well as digits 3, 5 and 8. Specifically, we want to verify that the objective function of the best initialization of the algorithm is actually decreasing, according to the definition of our methodology. We aim also to ascertain that the estimated covariance matrices for the three identified clusters are sparse, as expected due to the employment of the sparse CovGlasso estimator.

Objective function decrease:
Following the formulation of our likelihood-based methodology, we expect that, for each initialization of our algorithm, the value of the objective function decreases. This expectation is rooted in the definition of our objective function, where we search for the clustering that minimizes the summation of k minus penalized log-likelihoods, each representing the individual objective function of the CovGlasso methodology for each cluster. To verify this reduction in the objective function throughout iterations of the algorithm, we visualize the values stored in `loglik_vec`, a vector containing the objective function values for each iteration of the best initialization, introduced in Chapter 3. This best initialization is defined as the one that results in the lowest objective function value at the end of the entire algorithmic process, ultimately leading to our final clustering configuration.



(a) Clustering digits 0, 1 and 4

(b) Clustering digits 3, 5 and 8

Figure 4.8: Graphical behavior of the objective function for the best initialization.

From both plots in Figure 4.8, it is evident that there is a clear decreasing trend in the objective function. On the left side (robust clustering of digits 0, 1 and 4), we can observe convergence in 8 iterations, as the `loglik_vec` consists in 8 values. On the right side (robust clustering of digits 3, 5 and 8), convergence is instead achieved in 5 iterations. Particularly, it is important to note the exponential decrease in the values of the objective function for both cases, underscoring the successful performance of our algorithm. Indeed, the significant drops in the initial algorithm iterations, due to its starting point being far from the optimal solution, become smaller as the algorithm approaches convergence.

Covariance matrices sparsity:

Finally, we verify that each cluster obtained using CovGlasso in TCLUST exhibits the expected sparsity in its covariance matrix, as our methodology employs the CovGlasso approach for sparse covariance matrix estimation. Therefore, we create an individual plot for each covariance matrix to visually depict their sparsity patterns, following the approach outlined in [5]. These plots display a checkerboard pattern, representing non-zero elements as black squares and zero elements as white squares.

In the following Figures 4.9 and 4.10, we will present these types of plots, providing a visual representation of the final covariance matrices estimated by our algorithm for the three identified clusters: 0, 1, 4 (Figure 4.9) and 3, 5, 8 (Figure 4.10). These plots clearly demonstrate the sparsity of our covariance matrices across all clusters.

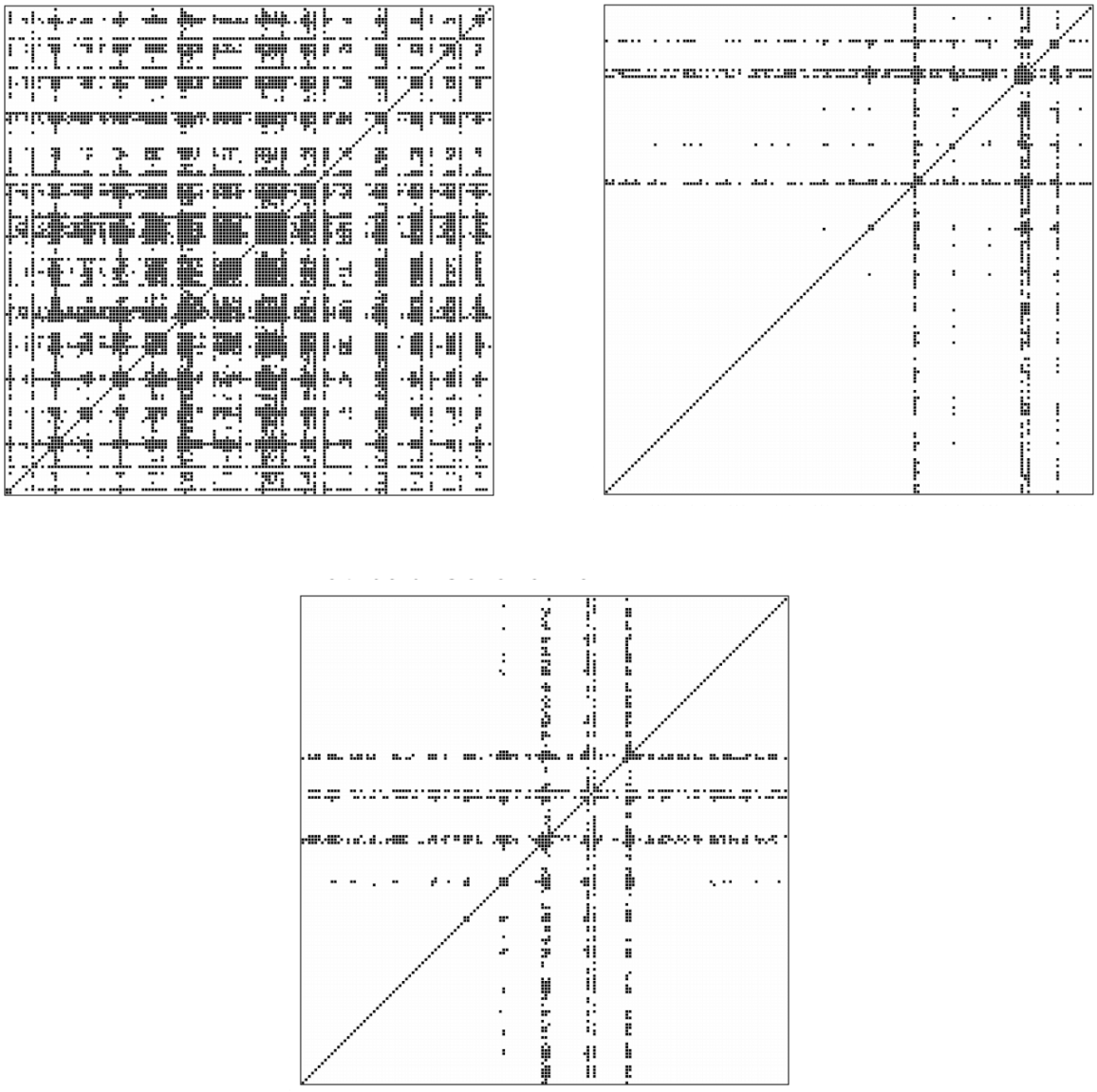


Figure 4.9: Sparsity plot of the estimated covariance matrices. The top left matrix corresponds to the cluster of 0's, the top right to the cluster of 1's, and the bottom to the cluster of 4's.

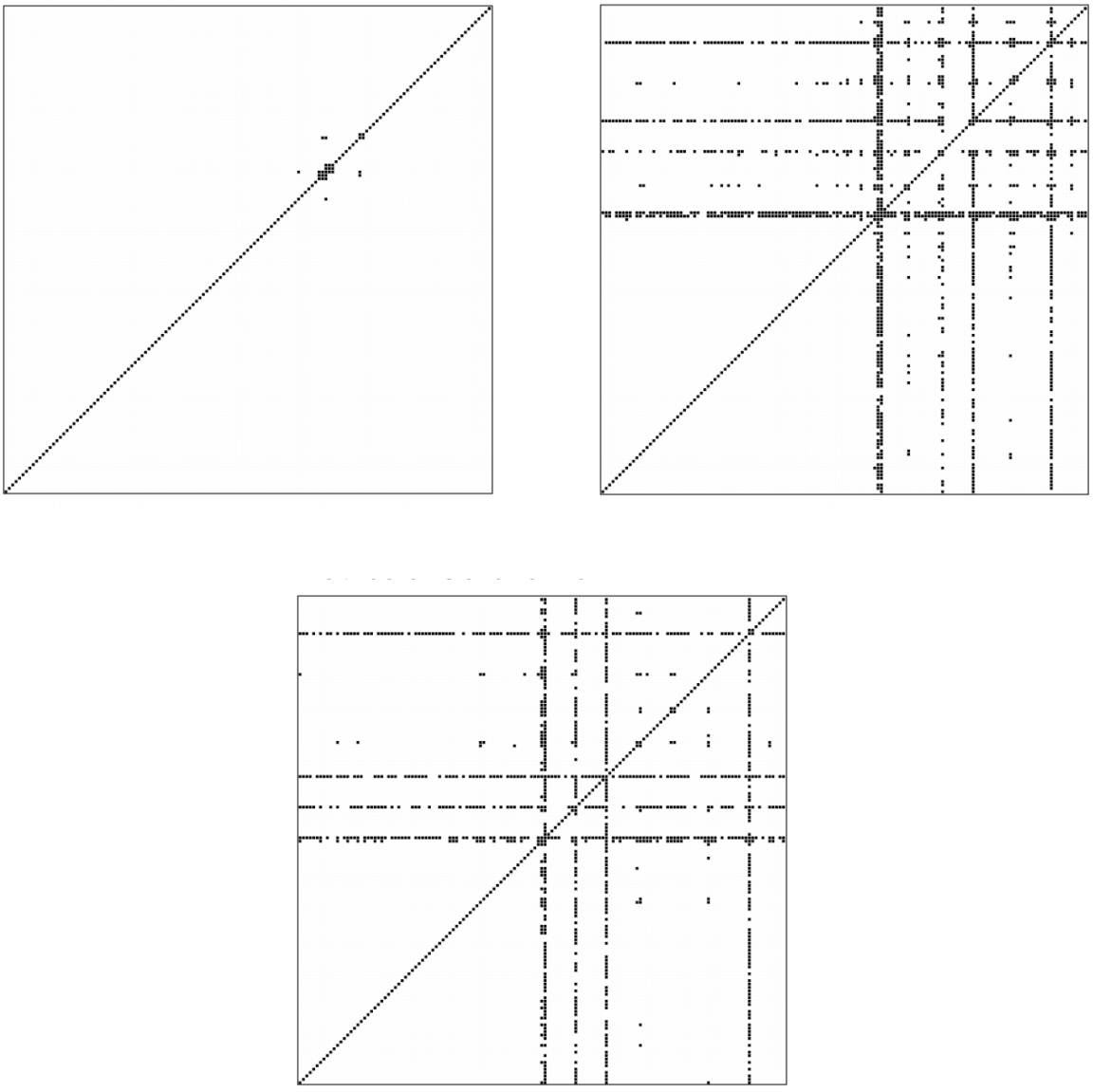


Figure 4.10: Sparsity plot of the estimated covariance matrices. The top left matrix corresponds to the cluster of 3's, the top right to the cluster of 5's, and the bottom to the cluster of 8's.

5 | Conclusions and future developments

This thesis proposes a solution to the very challenging problem of robust clustering for high-dimensional data. In the first two chapters, we explained the TCLUST algorithm, i.e., the one that underlies our new methodology, and introduced three regularized estimators for high-dimensional covariance matrices. In the third chapter, we provided a detailed description of the implementation process for our two methodologies: the heuristic one, derived from the incorporation of the Ledoit-Wolf linear shrinkage estimator into TCLUST, and the likelihood-based one, obtained by employing the sparse CovGlasso estimator into TCLUST. In the fourth chapter, we applied our two methodologies to both simulated and real-world data, to test and evaluate them. In the case of the simulated dataset, both of our algorithms demonstrated exceptional performance by correctly assigning all data points to their respective clusters and identifying all anomalous units within the data. For the dataset of digits 0, 1 and 4, both algorithms performed very well, achieving high overall accuracy and ARI scores. However, on the dataset containing digits 3, 5 and 8, CovGlasso in TCLUST outperformed Ledoit-Wolf in TCLUST, producing satisfactory results despite the inherent challenges of the task, which includes high data dimensionality, the presence of outliers, and significant similarity between classes. In conclusion, CovGlasso in TCLUST emerges as a robust solution for addressing clustering challenges in real-world high-dimensional data.

Now, let us explore potential enhancements and developments for this likelihood-based methodology.

1. **Exploration of additional covariance matrices estimators:**
   We should investigate the effectiveness of additional high-dimensional covariance matrices estimators, in addition to those already integrated into our methodologies. This exploration may involve the examination of emerging approaches or the customization of existing estimators to address the specific challenges associated with high-dimensional data.

2. **Performance evaluation on distinct high-dimensional real-world data:**
   We should expand the evaluation of our algorithm to encompass a broader spectrum of complex real-world data spanning various domains, extending beyond handwritten digit recognition. This expansion may involve applying the algorithm to datasets from various fields such as biomedicine, finance and other relevant domains. By doing so, we can acquire a more comprehensive understanding of its applicability and performance across a multitude of practical scenarios.

3. **Robustness evaluation at high contamination levels:**
   We should investigate how our algorithm performs under higher contamination levels, particularly when outliers are widespread, and develop strategies to effectively address these scenarios. By prioritizing robustness in these extreme conditions, we can enhance the usefulness of the algorithm in practical applications where data quality is a critical factor.

4. **Search for optimal lambda value:**
   We should search for the optimal lambda value to use as an input parameter in our algorithm. We could employ the approach outlined in [5], where a grid of 100 equispaced elements for the penalty term lambda is considered, with lower and upper extremes set as in that paper. We then seek the lambda that yields the best final result, meaning the one that leads to the lowest objective function value at the end of the algorithm execution.

5. **Employment of state-of-the-art visualization techniques:**
   We should employ advanced visualization techniques tailored for high-dimensional contexts. These techniques would play a crucial role in effectively representing clustering results, enhancing the interpretability of outcomes, and facilitating the understanding of complex data structures. By investing in state-of-the-art visualization tools, we can provide valuable insights into the performance and characteristics of our algorithm, making it more accessible and insightful for practitioners.

6. **Exploration of hybrid approaches:**
   We should try to integrate our algorithm with other machine learning or clustering approaches, aiming for a methodology that, while more computationally expensive, is generally more performing. One common hybrid approach is ensemble clustering, where multiple clustering algorithms are applied independently, and their results are combined together in a meaningful way. By exploring the potential for hybridization, we can discover new avenues for improving clustering outcomes and advancing the versatility and effectiveness of our methodology.

Bibliography

[1] J. Bien and R. J. Tibshirani. Sparse estimation of a covariance matrix. *Biometrika*, page 807–820, 2011.

[2] P. Boileau, N. Hejazi, B. Collica, J. Liu, M. van der Laan, and S. Dudoit. Package 'cvcovest'. 6 2023.

[3] K. Boudt, P. J. Rousseeuw, S. Vanduffel, and T. Verdonck. The minimum regularized covariance determinant estimator. *Statistics and Computing*, 30:113–128, 2020.

[4] C. Bouveyron, G. Celeux, B. Murphy, and A. Raftery. *Model-based Clustering and Classification for Data Science, with Applications in R - Chapter 8*, volume 4 of *50*. Cambridge University Press, 6 2019. ISBN 9781108644181.

[5] A. Casa, A. Cappozzo, and M. Fop. Group-wise shrinkage estimation in penalized model-based clustering. *Journal of Classification*, 39:648–674, 10 2022.

[6] D. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64 (4), 2000.

[7] M. Fop. Package 'covglasso'. 10 2022.

[8] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[9] H. Fritz, L. A. Garcìa-Escudero, and A. Mayo-Iscar. A fast algorithm for robust constrained clustering. *Computational Statistics and Data Analysis*, pages 124–136, 11 2012.

[10] H. Fritz, L. A. Garcìa-Escudero, and A. Mayo-Iscar. tclust: An r package for a trimming approach to cluster analysis. 12(47):1–26, 5 2012.

[11] H. Fritz, L. A. Garcìa-Escudero, and A. Mayo-Iscar. Package 'tclust'. 3 2023.

[12] L. A. Garcìa-Escudero, A. Gordaliza, C. Matràn, and A. Mayo-Iscar. A general

trimming approach to robust cluster analysis. *The Annals of Statistics*, 36(3):1324–1345, 6 2008.

[13] R. Grübel. A minimal characterization of the covariance matrix. *Metrika*, 35(1): 49–52, 1988.

[14] O. Ledoit and M. Wolf. Honey, i shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 30(4):110–119, 2004.

[15] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2 2004.

[16] J. Raymaekers and P. J. Rousseeuw. The cellwise minimum covariance determinant estimator. 7 2022.

[17] P. J. Rousseeuw. Multivariate estimation with high breakdown point. *Mathematical Statistics and Applications Vol. B*, pages 283–297, 1 1985.

[18] P. J. Rousseeuw and K. V. Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 6 1999.

[19] P. J. Rouusseeuw and C. Croux. Alternatives to the median absolute deviation. *Journal of American Statistical Association*, 88(424):1273–1283, 12 1992.

[20] V. Todorov. Package 'rrcov'. 6 2023.

[21] V. Todorov and P. Filzmoser. An object-oriented framework for robust multivariate analysis. *Journal of Statistical Software*, 32(3):1–47, 10 2009.

[22] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 6 2001.

[23] H. Wang. Coordinate descent algorithm for covariance graphical lasso. *Statistics and Computing*, 24:521–529, 2013.

[24] T. T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.

A | Appendix

Proof of Proposition 1.1

The main novelty of this algorithm, compared to the one in [12], is how constraints on the eigenvalue ratio are enforced. In particular, the algorithm proposed in [12] constrains eigenvalues by solving the following minimization problem:

$$\left(d_{11}^*, d_{12}^*, \ldots, d_{jl}^*, \ldots, d_{kp}^*\right) \mapsto \sum_{j=1}^{k} n_j \sum_{l=1}^{p} \left(\log\left(d_{jl}^*\right) + \frac{d_{jl}}{d_{jl}^*} \right), \tag{A.1}$$

under the restriction

$$\left(d_{11}^*, d_{12}^*, \ldots, d_{jl}^*, \ldots, d_{kp}^*\right) \in \Lambda,$$

in which Λ is the cone defined by

$$\Lambda = \left\{ d_{jl}^* : d_{jl}^* \leq c \cdot d_{rs}^* \text{ for every } j, r \in \{1, \ldots, k\} \text{ and } l, s \in \{1, \ldots, p\} \right\}. \tag{A.2}$$

This is clearly a more complex problem than minimizing (1.6), as its complexity tremendously increases with the number of clusters k and the dimension p. The problem of minimizing (A.1) in Λ was translated into a quadratic programming problem, which was approximately solved by recursive projections onto cones. These recursive projections must be carried out in each concentration step and, thus, the algorithm becomes extremely slow and even unfeasible for moderately high values of k and/or p. Moreover, there was a mistake in the algorithm proposed in [12], as the term n_j in (A.1) was omitted and, thus, the algorithm proposed there can only be applied to similarly sized clusters.

To prove the proposition, let us consider the spectral decomposition of the sample covariance matrices of observations given by:

$$\mathbf{T}_j = \frac{1}{n_j} \sum_{i \in R_j} (\boldsymbol{x}_i - \boldsymbol{m}_j)(\boldsymbol{x}_i - \boldsymbol{m}_j)^T = \mathbf{U}_j^T \mathbf{D}_j \mathbf{U}_j, \tag{A.3}$$

where $\mathbf{U}_j$ are orthogonal matrices and $\mathbf{D}_j = \text{diag}\,(d_{j1}, d_{j2}, \ldots, d_{jp})$ are diagonal matrices. Let S_j be the optimally constrained scatter matrices maximizing (1.1) under restriction (1.2) when $R_0, R_1, \ldots, R_k$ are known and parameters $\boldsymbol{m}_j$ and p_j are those given by (i) and (ii). Analogously to the previous decomposition of the sample covariance matrices, matrices $\boldsymbol{S}_j$ can be split up into $\boldsymbol{S}_j = \mathbf{V}_j^T \mathbf{D}_j^* \mathbf{V}_j$ with $\mathbf{V}_j$ orthogonal matrices and $\mathbf{D}_j^* = \text{diag}\left(d_{j1}^*, d_{j2}^*, \ldots, d_{jp}^*\right)$ diagonal matrices. Statement (iii) tells us that eigenvectors of the optimal constrained matrices $\boldsymbol{S}_j$ must be exactly the same as the eigenvectors of the unrestricted sample covariance matrices in (A.3) (i.e., we can set $\mathbf{U}_j = \mathbf{V}_j$). We just need to search for the optimal eigenvalues $\left\{d_{j,l}^*\right\}$ to obtain the optimally constrained scatter matrices $\boldsymbol{S}_j = \mathbf{U}_j^T \mathbf{D}_j^* \mathbf{U}_j$. Given the eigenvalues $\{d_{j,l}\}$, the optimal $\left\{d_{j,l}^*\right\}$ are obtained by minimizing expression (A.1) when $\left\{d_{j,l}^*\right\} \in \Lambda$, with Λ as defined in (A.2).
The proof of this claim follows from the proof of Proposition 4 in [12], with the difference that expression (A.1) now contains the cluster sizes n_j, whereas Eq. (A.1) in the mentioned article wrongly did not. Note that Λ can be written as:

$$\Lambda = \bigcup_{m \geq 0} \Lambda_m \quad \text{with } \Lambda_m = \bigcup_{m \geq 0} \left\{ d_{jl}^* : m \leq d_{jl}^* \leq cm \right\}.$$

Thus, for globally minimizing expression (A.1) in Λ, we need to be able to minimize it when $\left\{d_{j,1}^*\right\} \in \Lambda_m$ for every possible value $m > 0$. The minimization (for a fixed value of m) can be significantly simplified by considering truncated eigenvalues $d_{jl}^* = d_{jl}^m$ like those in (1.5).

Possible singularities in $\mathbf{T}_j$ are not a problem, provided that not all values of d_{jl} are 0 at the same time. Under this mild assumption, it is easy to see that $m > 0$ and this prevents that any value of d_{jl}^* drops to 0 (i.e. no singular clusters are obtained after the truncation of the eigenvalues).

Proof of Proposition 1.2

First, let us rewrite the target function (1.6) as

$$f : m \mapsto \sum_{j=1}^{k} n_j \left[\sum_{l=1}^{p} \left(\log(m) + d_{jl}/m \right) \left(d_{jl} < m \right) + \sum_{l=1}^{p} \left(\log\left(d_{jl} \right) + 1 \right) \left(m \leq d_{jl} < cm \right) + \sum_{l=1}^{p} \left(\log(cm) + d_{jl}/cm \right) \left(d_{jl} > cm \right) \right]. \tag{A.4}$$

Since f is a continuously differentiable function, it minimizes in one of its critical values, which satisfies the following fixed point equation:

$$m^* = \frac{\sum_{j=1}^{k} \left(s_j\left(m^*\right) + t_j\left(m^*\right)/c \right)}{\sum_{j=1}^{k} n_j r_j\left(m^*\right)}$$

with

$$r_j(m) = \sum_{l=1}^{p} \left(\left(d_{jl} < m \right) + \left(d_{jl} > cm \right) \right),$$

$$s_j(m) = \sum_{l=1}^{p} d_{jl} \left(d_{jl} < m \right) \text{ and } t_j(m) = \sum_{l=1}^{p} d_{jl} \left(d_{jl} > cm \right).$$

Functions r_j, s_j and t_j take constant values in the intervals $(-\infty, e_1], (e_1, e_2], \ldots, (e_{2k}, \infty)$. Therefore, we only need to evaluate (A.4) at the $2kp + 1$ values $m_1, \ldots, m_{2kp+1}$.

Proof of Theorem 2.1

Generate a p-variate sample $\boldsymbol{Z}$ with $p+1$ points for which $\boldsymbol{\Lambda} = \frac{1}{p+1}\sum_{j=1}^{p+1}(\boldsymbol{z}_i - \bar{z})(\boldsymbol{z}_i - \overline{z})^T$ is nonsingular and $\bar{z} = \frac{1}{p+1}\sum_{j=1}^{p+1}\boldsymbol{z}_i$. Then $\tilde{\boldsymbol{z}}_i = \boldsymbol{\Lambda}^{-1/2}(\boldsymbol{z}_i - \bar{z})$ has mean zero and covariance matrix $\boldsymbol{I}_p$. Now compute $\boldsymbol{y}_i = \mathbf{T}^{1/2}\tilde{\boldsymbol{z}}_i$, hence $\boldsymbol{Y}$ has mean zero and covariance matrix $\mathbf{T}$.
Next, create the artificial dataset

$$\tilde{\mathbf{X}}^1 = \left(w_1\left(\boldsymbol{x}_1^1 - \boldsymbol{m}_1\right), \ldots, w_h\left(\boldsymbol{x}_h^1 - \boldsymbol{m}_1\right), w_{h+1}\boldsymbol{y}_1, \ldots, w_k\boldsymbol{y}_{p+1}\right)$$

with $k = h + p + 1$ points, where $\boldsymbol{x}_1^1, \ldots, \boldsymbol{x}_h^1$ are the members of H_1. The factors w_i are given by

$$w_i = \begin{cases} \sqrt{k(1-\rho)/h} & \text{for } i = 1, \ldots, h \\ \sqrt{k\rho/(p+1)} & \text{for } i = h+1, \ldots, k. \end{cases}$$

The mean and covariance matrix of $\tilde{\mathbf{X}}^1$ are then

$$\frac{1}{k}\sum_{i=1}^{k}\tilde{\boldsymbol{x}}_i^1 = \sqrt{\frac{1-\rho}{kh}}\sum_{i=1}^{h}\left(\boldsymbol{x}_i^1 - \boldsymbol{m}_1\right) + \sqrt{\frac{\rho}{k(p+1)}}\sum_{j=1}^{p+1}\boldsymbol{y}_j = 0$$

and

$$\begin{aligned}\frac{1}{k}\sum_{i=1}^{k}\tilde{\boldsymbol{x}}_i^1\left(\tilde{\boldsymbol{x}}_i^1\right)^T &= \frac{1-\rho}{h}\sum_{i=1}^{h}\left(\boldsymbol{x}_i^1 - \boldsymbol{m}_1\right)\left(\boldsymbol{x}_i^1 - \boldsymbol{m}_1\right)^T + \frac{\rho}{p+1}\sum_{j=1}^{p+1}\boldsymbol{y}_j\boldsymbol{y}_j^T \\ &= (1-\rho)\boldsymbol{S}_1 + \rho\mathbf{T} = \mathbf{K}_1.\end{aligned}$$

The regularized covariance matrix $\mathbf{K}_1$ is thus the actual covariance matrix of the combined data set $\tilde{\mathbf{X}}^1$. Analogously we construct

$$\tilde{\mathbf{X}}^2 = \left(w_1\left(\boldsymbol{x}_1^2 - \boldsymbol{m}_2\right), \ldots, w_h\left(\boldsymbol{x}_h^2 - \boldsymbol{m}_2\right), w_{h+1}\boldsymbol{y}_1, \ldots, w_k\boldsymbol{y}_{p+1}\right)$$

where $\boldsymbol{x}_1^2, \ldots, \boldsymbol{x}_h^2$ are the members of H_2.
$\tilde{\mathbf{X}}_2$ has zero mean and covariance matrix $\mathbf{K}_2 = (1-\rho)\boldsymbol{S}_2 + \rho\mathbf{T}$.

Denote $d_{\mathbf{K}_1}(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{x}}^T (\mathbf{K}_1)^{-1} \tilde{\boldsymbol{x}}$. We can then prove that:

$$\frac{1}{k}\sum_{i=1}^{h} d_{\mathbf{K}_1}\left(\tilde{\boldsymbol{x}}_i^2\right) = \frac{1-\rho}{h}\sum_{i=1}^{h} d_{\mathbf{K}_1}\left(\boldsymbol{x}_i^2 - \boldsymbol{m}_2\right) \tag{A.5}$$

$$\leq \frac{1-\rho}{h}\sum_{i=1}^{h} d_{\mathbf{K}_1}\left(\boldsymbol{x}_i^2 - \boldsymbol{m}_1\right) \tag{A.6}$$

$$\leq \frac{1-\rho}{h}\sum_{i=1}^{h} d_{\mathbf{K}_1}\left(\boldsymbol{x}_i^1 - \boldsymbol{m}_1\right) \tag{A.7}$$

$$= \frac{1}{k}\sum_{i=1}^{h} d_{\mathbf{K}_1}\left(\tilde{\boldsymbol{x}}_i^1\right) \tag{A.8}$$

in which the second inequality (A.7) is the condition (2.6). The first inequality (A.6) can be shown as follows. Put $\boldsymbol{z}_i = (\mathbf{K}_1)^{-1/2}\boldsymbol{x}_i^2$ and $\tilde{\boldsymbol{z}} = (\mathbf{K}_1)^{-1/2}\boldsymbol{m}_1$ and note that $\overline{\boldsymbol{z}} = (\mathbf{K}_1)^{-1/2}\boldsymbol{m}_2$ is the average of the $\boldsymbol{z}_i$. Then (A.6) becomes

$$\sum_{i=1}^{h} \|\boldsymbol{z}_i - \overline{\boldsymbol{z}}\|^2 \leq \sum_{i=1}^{h} \|\boldsymbol{z}_i - \tilde{\boldsymbol{z}}\|^2,$$

which follows from the fact that $\tilde{\boldsymbol{z}}$ is the unique minimizer of the least squares objective $\sum_{i=1}^{k} \|\boldsymbol{z}_i - c\|^2$, so (A.6) becomes an equality if and only if $\tilde{\boldsymbol{z}} = \overline{\boldsymbol{z}}$ which is equivalent to $\boldsymbol{m}_2 = \boldsymbol{m}_1$. It follows that

$$\begin{aligned}\sum_{i=1}^{k} d_{\mathbf{K}_1}\left(\tilde{\boldsymbol{x}}_i^2\right) &= \sum_{i=1}^{h} d_{\mathbf{K}_1}\left(\tilde{\boldsymbol{x}}_i^2\right) + \frac{\rho}{p+1}\sum_{j=1}^{p+1} d_{\mathbf{K}_1}\left(\boldsymbol{y}_j\right) \\ &\leq \sum_{i=1}^{h} d_{\mathbf{K}_1}\left(\tilde{\boldsymbol{x}}_i^1\right) + \frac{\rho}{p+1}\sum_{j=1}^{p+1} d_{\mathbf{K}_1}\left(\boldsymbol{y}_j\right) \\ &= \sum_{i=1}^{k} d_{\mathbf{K}_1}\left(\tilde{\boldsymbol{x}}_i^1\right).\end{aligned}$$

Now put

$$b = \frac{\sum_{i=1}^{k} d_{\mathbf{K}}\left(\hat{\mathbf{x}}_{\mathbf{i}}^2\right)}{\sum_{i=1}^{k} d_{\mathbf{K_1}}\left(\hat{\mathbf{x}}_{\mathbf{i}}^1\right)} \leq 1.$$

If we now compute distances relative to $b\mathbf{K}_1$, we find

$$\begin{aligned}
&\frac{1}{k}\sum_{i=1}^{k} d_{b\mathbf{K}_1}\left(\tilde{\boldsymbol{x}}_i^2\right) = \frac{1}{b}\frac{1}{k}\sum_{i=1}^{k} d_{\mathbf{K}_1}\left(\tilde{\boldsymbol{x}}_i^2\right) = \frac{1}{k}\sum_{i=1}^{k} d_{\mathbf{K}_1}\left(\tilde{\boldsymbol{x}}_i^1\right) = \frac{1}{k}\sum_{i=1}^{k}\left(\tilde{\boldsymbol{x}}_i^1\right)^T\left(\mathbf{K}_1\right)^{-1}\tilde{\boldsymbol{x}}_i^1 \\
&= \frac{1}{k}\sum_{i=1}^{k}\left(\mathbf{K}_1^{-1/2}\tilde{\boldsymbol{x}}_i^1\right)^T\left(\mathbf{K}_1^{-1/2}\tilde{\boldsymbol{x}}_i^1\right) = \operatorname{trace}\left(\frac{1}{k}\sum_{i=1}^{k}\left(\mathbf{K}_1^{-1/2}\tilde{\boldsymbol{x}}_i^1\right)^T\left(\mathbf{K}_1^{-1/2}\tilde{\boldsymbol{x}}_i^1\right)\right) \\
&= \operatorname{trace}\left(\left(\mathbf{K}_1\right)^{-1/2}\left(\frac{1}{k}\sum_{i=1}^{k}\left(\tilde{\boldsymbol{x}}_i^1\right)\left(\tilde{\boldsymbol{x}}_i^1\right)^T\right)\left(\mathbf{K}_1\right)^{-1/2}\right) = \operatorname{trace}\left(\boldsymbol{I}_p\right) = p.
\end{aligned}$$

From the theorem by Grübel [13], it follows that $\mathbf{K}_2$ is the unique minimizer of $\det(\boldsymbol{S})$ among all $\boldsymbol{S}$ for which $\frac{1}{k}\sum_{i=1}^{k} d_{\boldsymbol{S}}\left(\tilde{\boldsymbol{x}}_i^2\right) = p$ (note that the mean of $\tilde{\boldsymbol{x}}_i^2$ is zero). Therefore

$$\det\left(\mathbf{K}_2\right) \leq \det\left(b\mathbf{K}_1\right) \leq \det\left(\mathbf{K}_1\right),$$

and we can only have $\det\left(\mathbf{K}_2\right) = \det\left(\mathbf{K}_1\right)$ if both of these inequalities are equalities. For (A.6), by uniqueness we can only have equality if $\mathbf{K}_2 = b\mathbf{K}_1$. For (A.7), equality holds if and only if $b = 1$. Combining both yields $\mathbf{K}_2 = \mathbf{K}_1$. Moreover, $b = 1$ implies that (A.6) becomes an equality, hence $\boldsymbol{m}_2 = \boldsymbol{m}_1$.
This concludes the proof of Theorem 2.1.

Proof of Theorem 2.2

By a change of variables, problem (2.7) can be rewritten as:

$$\min_{\rho,v} E\left[\|\boldsymbol{\Sigma}^* - \boldsymbol{\Sigma}\|^2\right]$$
$$\text{s.t. } \boldsymbol{\Sigma}^* = \rho v \boldsymbol{I} + (1-\rho)\boldsymbol{S}.$$

With a little algebra, and observing that $E[\boldsymbol{S}] = \boldsymbol{\Sigma}$, we can rewrite the objective as

$$E\left[\|\boldsymbol{\Sigma}^* - \boldsymbol{\Sigma}\|^2\right] = \rho^2\|\boldsymbol{\Sigma} - v\boldsymbol{I}\|^2 + (1-\rho)^2 E\left[\|\boldsymbol{S} - \boldsymbol{\Sigma}\|^2\right]. \tag{A.9}$$

Therefore, the optimal value of v can be obtained as the solution to a reduced problem that does not depend on ρ : $\min_v \|\boldsymbol{\Sigma} - v\boldsymbol{I}\|^2$. Remember that the norm of the identity is one by convention, so the objective of this problem can be rewritten as $\|\boldsymbol{\Sigma} - v\boldsymbol{I}\|^2 = \|\boldsymbol{\Sigma}\|^2 - 2v\langle\boldsymbol{\Sigma}, \boldsymbol{I}\rangle + v^2$. The first-order condition is: $-2\langle\boldsymbol{\Sigma}, \boldsymbol{I}\rangle + 2v = 0$. The solution is: $v = \langle\boldsymbol{\Sigma}, \boldsymbol{I}\rangle = \mu$. Replacing v by its optimal value μ in (A.9), we can rewrite the objective of the original problem as $E\left[\|\boldsymbol{\Sigma}^* - \boldsymbol{\Sigma}\|^2\right] = \rho^2\alpha^2 + (1-\rho)^2\beta^2$.
The first-order condition is: $2\rho\alpha^2 - 2(1-\rho)\beta^2 = 0$. The solution is: $\rho = \beta^2/\left(\alpha^2 + \beta^2\right) = \beta^2/\delta^2$. Note that $1 - \rho = \alpha^2/\delta^2$.
At the optimum, the objective is equal to: $\left(\beta^2/\delta^2\right)^2\alpha^2 + \left(\alpha^2/\delta^2\right)^2\beta^2 = \alpha^2\beta^2/\delta^2$, and this completes the proof.

Note that $\mu\boldsymbol{I}$ can be interpreted as a shrinkage target and the weight β^2/δ^2 placed on $\mu\boldsymbol{I}$ as a shrinkage intensity [15]. The percentage relative improvement in average loss over the sample covariance matrix is equal to

$$\frac{E\left[\|\boldsymbol{S} - \boldsymbol{\Sigma}\|^2\right] - E\left[\|\boldsymbol{\Sigma}^* - \boldsymbol{\Sigma}\|^2\right]}{E\left[\|\boldsymbol{S} - \boldsymbol{\Sigma}\|^2\right]} = \frac{\beta^2}{\delta^2},$$

same as the shrinkage intensity. Therefore, everything is controlled by the ratio β^2/δ^2, which is a properly normalized measure of the error of the sample covariance matrix $\boldsymbol{S}$. Intuitively, if $\boldsymbol{S}$ is relatively accurate, then you should not shrink it too much, and shrinking it will not help you much either; if $\boldsymbol{S}$ is relatively inaccurate, then you should shrink it a lot, and you also stand to gain a lot from shrinking [14].

Proof of Theorem 2.3

The following lemma will be useful in proving Theorem 2.3; if you are interested in the proof of the lemma, please refer to [15].

Lemma A.1
If u^2 is a sequence of non-negative random variables (implicitly indexed by n, as usual) whose expectations converge to zero, and τ_1, τ_2 are two nonrandom scalars, and $\frac{u^2}{d^{t_1}\delta^{t_2}} \leqslant 2\left(d^2+\delta^2\right)$ a.s., then

$$E\left[\frac{u^2}{d^{\tau_1}\delta^{\tau_2}}\right] \to 0$$

Now, we consider the first statement of Theorem 2.3:

$$\begin{aligned}
&\|\boldsymbol{S}^* - \boldsymbol{\Sigma}^*\|^2 = \left\|\frac{\beta^2}{\delta^2}(m-\mu)\boldsymbol{I} + \left(\frac{a^2}{d^2} - \frac{\alpha^2}{\delta^2}\right)(\boldsymbol{S} - m\boldsymbol{I})\right\|^2 = \\
= &\frac{\beta^4}{\delta^4}(m-\mu)^2 + \left(\frac{a^2}{d^2} - \frac{\alpha^2}{\delta^2}\right)^2 \|\boldsymbol{S} - m\boldsymbol{I}\|^2 + 2\frac{\beta^2}{\delta^2}(m-\mu)\left(\frac{a^2}{d^2} - \frac{\alpha^2}{\delta^2}\right)\langle \boldsymbol{S} - m\boldsymbol{I}, \boldsymbol{I}\rangle = \\
= &\frac{\beta^4}{\delta^4}(m-\mu)^2 + \left(\frac{a^2}{d^2} - \frac{\alpha^2}{\delta^2}\right)^2 d^2 \leqslant (m-\mu)^2 + \frac{\left(a^2\delta^2 - \alpha^2 d^2\right)^2}{d^2\delta^4}.
\end{aligned} \tag{A.10}$$

It is sufficient to show that the expectations of both terms on the right-hand side of Eq. (A.10) converge to zero.
Now consider the second term. Since $\alpha^2 \leqslant \delta^2$ and $a^2 \leqslant d^2$, note that

$$\frac{\left(a^2\delta^2 - \alpha^2 d^2\right)^2}{d^2\delta^4} \leqslant d^2 \leqslant 2\left(d^2+\delta^2\right) \quad \text{a.s.}$$

Furthermore, since $a^2-\alpha^2$ and $d^2-\delta^2$ converge to zero in quadratic mean, and since α^2 and δ^2 are bounded, $a^2\delta^2 - \alpha^2 d^2 = (a^2 - \alpha^2)\,\delta^2 - \alpha^2\,(d^2 - \delta^2)$ converges to zero in quadratic mean. Therefore, the assumptions of Lemma A.1 are verified with $u^2 = (a^2\delta^2 - \alpha^2 d^2)^2$, $\tau_1 = 2$, and $\tau_2 = 4$. This implies that

$$E\left[\frac{(a^2\delta^2 - \alpha^2 d^2)^2}{d^2\delta^4}\right] \to 0$$

The expectation of the second term on the right-hand side of Eq. (A.10) converges to zero. Going back, $\|\boldsymbol{S}^* - \boldsymbol{\Sigma}^*\|$ converges to zero in quadratic mean. This completes the proof of the first statement of Theorem 2.3.
Now consider the second statement:

$$\begin{aligned} E\left[\|\|\boldsymbol{S}^* - \boldsymbol{\Sigma}\|^2 - \|\boldsymbol{\Sigma}^* - \boldsymbol{\Sigma}\|^2\,|\right] &= E\left[|\langle \boldsymbol{S}^* - \boldsymbol{\Sigma}^*, \boldsymbol{S}^* + \boldsymbol{\Sigma}^* - 2\boldsymbol{\Sigma}\rangle|\right] \\ &\leqslant \sqrt{E\left[\|\boldsymbol{S}^* - \boldsymbol{\Sigma}^*\|^2\right]}\sqrt{E\left[\|\boldsymbol{S}^* + \boldsymbol{\Sigma}^* - 2\boldsymbol{\Sigma}\|^2\right]} \end{aligned} \tag{A.11}$$

As we have shown above, the first term on the right-hand side of Eq. (A.11) converges to zero. Given that $E\left[\|\boldsymbol{\Sigma}^* - \boldsymbol{\Sigma}\|^2\right]$ is bounded, it also implies that the second term on the right-hand side of Eq. (A.11) is bounded. Therefore, the product of the two terms on the right-hand side of Eq. (A.11) converges to zero.
This completes the proof of the second and final statement.

List of Figures

List of Tables