Estimation of Gaussian Bi-Clusters with General Block-Diagonal Covariance Matrix & Applications to Topic Modelling

Author:
Anastasiia Livochka

Supervisor:
Ryan Browne

Associate Professor,
Dept. of Statistics and
Actuarial Science,
University of Waterloo

Research paper submitted in fulfillment of the requirements for the degree of Master of Mathematics (MMath) in
Computational Mathematics
University of Waterloo

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Bi-clustering is a technique that allows for the simultaneous clustering of observations and features in the dataset. This technique is often used in bioinformatics, text mining, and time series analysis. An important advantage of this method is the ability to uncover multiple "views" (or column groupings) in the data.

In this work, we discuss a bi-clustering version of the Gaussian Mixture Model based on the covariance matrix restriction to be block-diagonal. We show that current implementations of the GMM-based bi-clustering algorithm impose severe limitations on the structure of the covariance matrix. This project proposes multiple heuristics that relax the problem and allow for greater applicability of the model.

We show that the clustering accuracy of the proposed model is comparable to other known techniques. We demonstrate the proposed model's advantages in topic modelling and data exploration applications.

Dedicated to the Armed Forces of Ukraine who have been protecting my friends and family back in Ukraine during my Master's journey.

Table of Contents

Chapter 1

Introduction

1.1 Clustering Problem & Gaussian Mixture Models

Humans are particularly good at classifying objects into natural groups based on some notion of similarity and the goal they want to achieve. However, as the set of objects to organize and the number of features grows, we require faster and more memory-efficient tools to complete the task. Clustering is an important area of unsupervised machine learning that deals with the problem of grouping objects so that those in one cluster are more similar to each other than to objects in other clusters[17]. The main applications of clustering include data exploration and pattern recognition[7].

Clustering has been widely studied for decades, and many clustering algorithms have been developed. Interestingly, the great variety of the available models can be attributed to the fact that precisely defining a cluster presents a problem [39]. Some algorithms (e.g. DBSCAN, HDBSCAN) leverage density assumptions by defining clusters as connected dense regions in the space [31, 37]. Other approaches view clusters as mixtures of certain distributions, with Gaussian Mixture Models (GMM) among the most popular ones [45].

The GMM clustering algorithm also proved effective in the presence of multi-view and incomplete data [55], which is essential for modern applications. Additionally, Gaussian Mixture Models can be easily adapted for semi-supervised settings [25]. Hence, we consider building on top of this algorithm.

1.2 Topic Modelling & Bi-Clustering

Topic Modelling [52] is an area of Natural Language Processing that solves the problem of finding abstract human-interpretable topics in a collection of an arbitrary number of documents. Essentially, the problem of grouping documents into unsupervised topics can be solved by leveraging clustering techniques. However, classical clustering algorithms will require additional data post-processing to provide interpretations of the topics they produce [47].

Additionally, human language is incredibly complex. There are multiple possible groupings one can find in a set of documents, some having a stronger signals-no-noise ratio than others. For example, Tripadvisor's hotel reviews can be grouped by review sentiment (recommend/do not recommend), hotel geographical location or more generalized hotel features (has access to the beach, superb breakfast, polite managers etc.). Each of the clusterings can be desired depending on the application. However, how do we tell the model which type of grouping we aim to obtain?

One of the promising solutions to the above issues is bi-clustering. Bi-clustering is a technique that allows for the simultaneous clustering of observations and features in the dataset[29]. This technique is often used in bioinformatics, text mining, and time series analysis [34, 35, 11]. An important advantage of this method is the ability to uncover multiple "views" (or column groupings) in the data. In addition, bi-clustering is a promising approach for solving topic modelling problems as it can automatically provide column cluster names if used for the documents-words matrix type of text data representation [46].

1.3 Focus of the Work

In this work, we aim to review current bi-clustering algorithms and expand on the GMM-based bi-clustering approach. In chapter 2, we argue that existing implementations for mixture model-based techniques are overly restrictive. In chapter 3, we propose three heuristics to estimate the general block-diagonal covariance matrix for the gaussian multivariate distribution, and we discuss an Expectation Maximization (EM) algorithm that utilizes proposed estimators [14]. In chapter 6, we report on the results of the proposed method's performance on the topic modelling task.

Chapter 2

Literature Review

Topic modelling is an essential tool in extracting latent structure from a large collection of documents. Latent Dirichlet Allocation (LDA) [9] is one of the earliest and most frequently used algorithms that view documents as generated from a mixture of hidden topics. In turn, topics are generated by the mixtures of words. LDA demonstrates significantly elevated performance compared to the earlier algorithms. However, it is sensitive to hyperparameters and therefore requires a careful estimation procedure. Data sparsity is another problem as LDA encounters issues when presented with large vocabulary sizes [50]. However, one of the main limitations of the algorithms based on the bag-of-words is the complete unawareness of contextual and semantic relationships between words and sentences in the data. Some of the recently published deep language representation models (e.g. BERT [15]) proved to be highly effective in generating contextual word and sentence [44] vectors. BERTopic [27] proposes solving the topic modelling problem via HDBSCAN clustering on top of the modern text embeddings. Another major problem is the incredible complexity of human language and the fact that most of the collections of text documents will likely contain more than one set of topics. None of the algorithms mentioned above allow for uncovering multiple "views" or multiple contextual subspaces in the data. Bi-clustering is one potential solution to this problem.

Similar to the cluster, there is no single definition of a bi-cluster. Therefore, a variety of bi-clustering models exist. One of the first applications of bi-clustering to text documents is based on the spectral graph [16] partitioning heuristic. Similarly, [32] shows that the checkerboard structure of the data matrix can be inferred from its eigenvectors. There are many greedy approaches proposed for the co-clustering of gene expressions [42, 12, 6]. For instance, [5] introduces an approach that finds the hidden order-preserving sub-matrices in the data matrix.

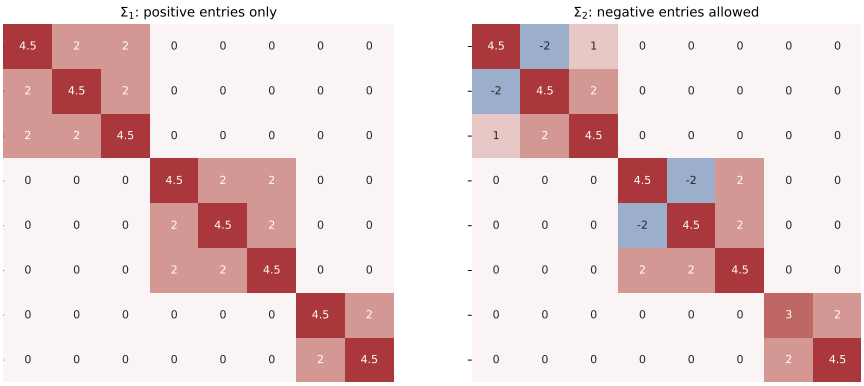

Figure 2.1: Illustration of the block diagonal matrices with off-diagonal elements limited to positive (*left*) and general (*right*) structure

Most model-based approaches assume the observations in the same row cluster and column groups to follow the same distribution. One component of the gaussian mixture model generates one row cluster that follows the same distribution $\mathcal{N}(\mu, \Sigma)$. In turn, column groups can be inferred from the structure of the covariance matrix. For illustration, let us assume the structure of covariance matrix from Fig. 2.1. The distribution mean μ and covariance Σ can be written as:

$$\Sigma = \begin{bmatrix} \Sigma_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Sigma_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Sigma_3 \end{bmatrix}, \quad \mu = \begin{pmatrix} \mu_1 & \mu_2 & \mu_3 \end{pmatrix}$$

Where $(\mu_1,\ \Sigma_1), (\mu_2, \Sigma_2), (\mu_3, \Sigma_3)$ define three distinct column groupings. Note that column permutations may be required to achieve a checkerboard-like structure illustrated in Fig. 2.1. This adds to the non-triviality of the estimation of block-diagonal covariance matrix.

The initial model-based bi-clustering approaches introduced additional latent variables to indicate column clusters [8, 28]. The authors in [24] propose a Gaussian model-based clustering for high-dimensional data. They partition columns of the dataset twice - once by means and once by variances. Others [49, 38] leveraged the bi-clustering framework rooted in the latent factor analyzer approach. The important advantage of the factor-analyzer structure is that it decreases the number of estimated parameters by assuming a latent low-dimensional representation of high-dimensional variables. However, this representation restricts the covariance matrix structure they are able to recover. For example, [36, 53] requires the off-diagonal elements of the covariance matrix to 1. The authors in [49] relaxed

some constraints by characterizing covariance as $B^T T B + D$ where B is binary, T is diagonal matrix with positive entries and D is diagonal. This enforces all of the off-diagonal entries of i-th covariance block to be equal to t_{ii}, where $t_{ii} > 0$ be definition, therefore it is not able to reconstruct the covariance matrix from Fig. 2.1 (*right*). In addition, the parameter estimation utilized in [49] has limitations in application to high-dimensional data due to the increased complexity of finding column groupings.

In this work, we aim to build on the model-based bi-clustering procedure by fixing some of the issues with gaussian mixture models. In particular, we allow the covariance matrix of the underlying mixture components to have a general block-diagonal structure. Then we explore multiple readily available covariance matrix estimators [33, 22, 10]. Neither satisfies our desirable conditions for the estimator, namely (i) enforces the block-diagonal structure on the covariance matrix; and (ii) allows for general covariance matrix structure inside the blocks.

Chapter 3

Estimation of Block-Diagonal Covariance Matrix

This chapter discusses the model structure and parameter estimation procedure for Gaussian Mixture Model-based bi-clustering algorithm. We propose three methods for estimating the block-diagonal covariance matrix (as illustrated on the Fig. 2.1 - which is a key ingredient for the model-based GMM bi-clustering procedure. For simplicity, we derive an estimation for one component of GMM. Chapter 4 discusses extending the techniques for multiple mixtures.

As described in the literature review, many methods exist to estimate a general covariance matrix with a block-diagonal structure. However, there is no guarantee that real-world data will indeed be generated from a model with such covariance. Therefore, we need to impose this restriction artificially for bi-clustering purposes. The main aim of this work is to propose an approach such that: (i) imposes a block-diagonal structure onto the covariance matrix; (ii) does not restrict the covariance matrix structure inside the blocks.

3.1 Imposing Block-Diagonal Structure Onto General Covariance Matrix

Let us assume that we have a normally distributed data $X \sim \mathcal{N}(\mu, \Sigma)$. One way to compute feature clusters, common for biclustering, is by estimating a block-diagonal covariance matrix Θ that is close to the original Σ. In this work, we propose to parametrize Θ by the

following:

$$\Theta = D_1 \Sigma D_1 + \cdots + D_K \Sigma D_K \tag{3.1}$$
$$\text{where } D_1 + \cdots + D_K = I_K$$
$$\forall k \in [1, n] : \; D_k[i, j] = \begin{cases} 0, \text{ if } i \neq j \\ 0 \text{ or } 1, \text{ if } i = j \end{cases}$$

In other words, the D_k are binary diagonal matrices that sum up to identity. This allows for each D_k to define a column cluster k. For example, $D_1 = \text{diag}(\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix})$ for covariance matrices illustrated at Fig. 2.1, where where $\text{diag}(\vec{a})$ creates a diagonal matrix using the element from the $\vec{a}$.

The proposed parametrization imposes a general block-diagonal structure on the covariance matrix. In the next three sections, we discuss three approaches to estimate D_k: greedy, convex relaxation (numerical) and hierarchical.

3.2 Greedy Approach

Motivated by the approach in [49], we decided to adopt a greedy algorithm for estimating D_k in the proposed parametrization of Σ deom eq. (3.1). Let us assume that we have independent identically distributed data generated from one component of the Gaussian Mixture Model:

$$X = \{X_1, X_2, \ldots X_N\}, \quad \text{where } X_i \sim \mathcal{N}(\mu, \Sigma), \quad \mu \in R^m, \Sigma \in R^{m \times m} \tag{3.2}$$

In the first step, we obtain Maximum Likelihood Estimates (MLEs) of the sample mean and sample covariance:

$$\hat{\mu} = \frac{\sum_{i=1}^{N} X_i}{N},$$
$$\hat{\Sigma} = \frac{\sum_{i=1}^{N} (X_i - \hat{\mu})(X_i - \hat{\mu})^T}{N}.$$

For the initialization of D_k, we apply Principal Component Analysis to form a matrix $P \in R^{K \times m}$ of the first K principal components of the scaled $\hat{\Sigma}$ [38, 49]. For every column j we compute $p = argmax(P_j)$, where P_j denotes j-th column of P. Then we put $D_p[j, j] = 1$, which can be interpreted as assigning the j-th column to the p-th column cluster.

In the second step, we iteratively reassign column clusters by selecting such cluster for the column j that maximizes data log-likelihood (Θ computed as in eq. (3.1)):

$$\mathcal{L}(\theta; X) = -\frac{N}{2}\log|\hat{\Theta}| - \frac{1}{2}\sum_{i=1}^{N}(X_i - \hat{\mu})^T\hat{\Theta}^{-1}(X_i - \hat{\mu})$$

Starting with the first column $j = 1$, we choose:

$$p = argmax_{1\leq k\leq K}\mathcal{L}(\hat{\Theta}_k), \text{ with } \hat{\Theta}_k = \sum_{i=1}^{K} D_i\hat{\Sigma}D_i,\ D_{i=k}[j,j] = 1 \wedge D_{i\neq k}[j,j] = 0$$

The cluster assignment for the j-th columns is updated by setting $D_p[j,j] = 1$ and $D_{k\neq p}[j,j] = 0$. Conditioning on the first updated columns, we repeat the procedure for the rest of them.

3.3 Convex Relaxation Approach

We propose estimating D_k by formulating a convex optimization problem. Given that all off-diagonal elements of D_k are known to be zero, we can minimize the number of parameters by introducing:

$$D_k = \text{diag}(\vec{s}_k), \quad \vec{s}_k = \begin{pmatrix} s_{1k} & \dots & s_{mk} \end{pmatrix}^T$$

As assumed in the section above, we have independent identically distributed data generated from one component of the Gaussian Mixture Model (eq. 3.2) and there are K column clusters. This approach is based on the maximization of log-likelihood. Note that the proposed parametrization of the precision or covariance matrix is equivalent because the inverse of a block diagonal matrix is block diagonal matrix. However, using the precision matrix will simply the derivatives. We denote $\hat{\Sigma}$ to be a MLE of the true covariance, and we set:

$$\Theta^{-1} = D_1\hat{\Sigma}^{-1}D_1 + \cdots + D_K\hat{\Sigma}^{-1}D_K \tag{3.3}$$

We want to maximize the log-likelihood function with respect to the block-diagonal structure of the covariance matrix:

$$\mathcal{L}(\theta; X) = -\frac{N}{2}\log|\Theta| - \frac{1}{2}\sum_{i=1}^{N}\text{Tr}(\Theta^{-1}(X_i - \mu)(X_i - \mu)^T)$$

$$\mathcal{L}^{\star}(\theta; X) = \frac{1}{N}\mathcal{L}(\theta; X) = \frac{1}{2}\log|\Theta^{-1}| - \frac{1}{2}\sum_{i=1}^{N}\mathrm{Tr}(\Theta^{-1}R).$$

Where $R = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)(X_i - \mu)^T$. By construction, we have $R = \hat{\Sigma}$. Alternatively, we can rewrite eq. (3.3) as $\Theta^{-1} = \sum_{k=1}^{K} D_k R^{-1} D_k$ and section 3.3 as:

$$\mathcal{L}^{\star}(\theta; X) = \frac{1}{2}\log|\sum_{k=1}^{K} D_k R^{-1} D_k| - \frac{1}{2}\mathrm{Tr}(\sum_{k=1}^{K} D_k R^{-1} D_k R). \tag{3.4}$$

We utilize the properties of the trace and element-wise product operators to obtain

$$\mathrm{Tr}(D_k R^{-1} D_k R) = \vec{s}_k{}^T (R^{-1} \odot R)\vec{s}_k \tag{3.5}$$

Where $\odot$ is element-wise or Hadamard product.

We aim to find $S = \begin{bmatrix}\vec{s}_1 & \vec{s}_2 & \dots & \vec{s}_K\end{bmatrix}$ that maximizes log-likelihood derived above (3.4). Keeping in mind that $\vec{s}_k$ should be binary by definition of eq. (3.1), we obtain a non-linear integer problem, which is known for its complexity [30]. To simplify, we relaxed the $\vec{s}_k$ binary constraints and leveraged quadratic programming to solve the following:

$$\max \mathcal{L}^{*} \quad \text{st.} \quad \forall k: \ \vec{s}_k > \vec{0} \quad \text{and} \quad \sum_{k=1}^{K} s_k = \vec{1} \tag{3.6}$$

Naturally, the optimal solution will be $\forall k: \ \vec{s}_k = \begin{pmatrix}\frac{1}{K} & \frac{1}{K} & \dots & \frac{1}{K}\end{pmatrix}^T$ as it results in $\Theta^{-1} = \hat{\Sigma}^{-1}$. To avoid the trivial solution, we added a penalty on the logarithm of the determinant of $S^T S$. To push $\vec{s}_k$ towards binary values we add the $\sum_{k=1}^{K} \vec{s}_k{}^T \vec{s}_k - m$ term to the objective function:

$$\mathcal{Q} = \mathcal{L}^{*} + \gamma(\sum_{k=1}^{K} \vec{s}_k{}^T \vec{s}_k - m) - \lambda \log|S^T S| \qquad \gamma, \lambda \in R. \tag{3.7}$$

Such optimization problem can be solved numerically utilizing gradient descent. Gradient descent is an iterative, first-order optimization method that finds local minima or maxima of the differentiable function. We aim to estimate the parameters $S = \begin{bmatrix}\vec{s}_1 & \vec{s}_2 & \dots & \vec{s}_K\end{bmatrix}$ by maximizing penalized log-likelihood eq. (3.7) given the observed data $X \sim \mathcal{N}(\mu, \Sigma)$. To simplify the problem, we propose to estimate the columns of S iteratively, starting with $\vec{s}_1$ and conditioning on the values of all other column vectors.

Let's compute the partial derivatives of $\mathcal{Q}$ eq. (3.7) wrt $\vec{s}_k$. Note that $D_k = \mathrm{diag}(\vec{s}_k)$ and $\partial\ \mathrm{diag}(\vec{s}_k) = \mathrm{diag}(\partial\vec{s}_k)$:

$$\frac{\partial}{\partial \vec{s}_k}(\frac{1}{2}\log|\sum_{k=1}^{K} D_k R^{-1} D_k|) = (T \odot R^{-1})\vec{s}_k \tag{3.8}$$

Where $T = (\sum_{k=1}^{K} D_k R^{-1} D_k)^{-1}$. We utilize the term rearrangement suggested in equation 3.5:

$$\frac{\partial}{\partial \vec{s}_k}(-\frac{1}{2}\operatorname{Tr}(\sum_{k=1}^{K} D_k R^{-1} D_k R)) = \frac{\partial}{\partial \vec{s}_k}(-\frac{1}{2}\sum_{k=1}^{K} \vec{s}_k{}^T (R^{-1} \odot R)\vec{s}_k) = -(R^{-1} \odot R)\vec{s}_k \tag{3.9}$$

Now we will take the partial derivatives of the penalty terms:

$$\frac{\partial}{\partial \vec{s}_k}(\gamma(\sum_{k=1}^{K} \vec{s}_k{}^T \vec{s}_k - m)) = 2\gamma\vec{s}_k \tag{3.10}$$

$$\frac{\partial}{\partial \vec{s}_k}(-\lambda \log|S^T S|) = -2\lambda(S(S^T S)^{-1})_k \tag{3.11}$$

where A_k denotes the k-th column of A. Combining all of the above derivations:

$$\frac{\partial \mathcal{Q}}{\partial \vec{s}_k} = (T \odot R^{-1})\vec{s}_k - (R^{-1} \odot R)\vec{s}_k + 2\gamma\vec{s}_k - 2\lambda(S(S^T S)^{-1})_k \tag{3.12}$$

To maximize the $\mathcal{Q}$, at every iteration, we take a step in the direction of a positive gradient. Let us denote $\omega \in R$ to be a learning rate, then the proposed update for $\vec{s}_k$ is defined as:

$$\vec{s}_k{}^{(t+1)} = \vec{s}_k{}^{(t)} + \omega((T^{(t)} \odot R^{-1} - R^{-1} \odot R + 2\gamma)\vec{s}_k{}^{(t)} - 2\lambda(S(S^T S)^{-1})_k^{(t)}) \tag{3.13}$$

3.4 Hierarchical Clustering Approach

Hierarchical clustering refers to a widely utilized family of clustering algorithms [40, 4, 18], that are based on an iterative procedure of either merging or splitting nested clusters. Merging or splitting is also known as bottom-up and top-down approaches correspondingly. Many readily available implementations and generalizations of the algorithm ensure robustness on various input data configurations [3].

We propose to leverage the bottom-up hierarchical approach known as agglomerative clustering to find the desired number of column clusters. However, instead of clustering the transpose of the dataset, we propose to use the absolute values of correlations of the

columns as a (i) similarity/dissimilarity metric between variables or (ii) feature representations to cluster. Let us denote $\hat{R}$ to be an estimate of the correlation matrix of columns in the dataset X (eq.3.2) and $\hat{\Sigma}$ to be an MLE of the covariance matrix, then

$$\hat{R} = \text{diag}(\hat{\Sigma})^{-\frac{1}{2}} \; \hat{\Sigma} \; \text{diag}(\hat{\Sigma})^{-\frac{1}{2}}$$

$$\hat{R}^{\star} = abs(\hat{R}) \tag{3.14}$$

where $abs(A)$ applies the absolute value element-wise to the matrix A and $\text{diag}(A)$ creates a diagonal matrix using the diagonal elements of the matrix A. Assuming correlations as similarity metric (i), we define $D = \mathbf{1} - \hat{R}^{\star}$ to be a distance matrix input to the clustering algorithm. Under this setting, every iteration of agglomerative clustering merges two features in one cluster when they correlate the most with each other. Assuming the second formulation (ii), we treat $\hat{R}^{\star}$ as feature vectors. To be merged in one cluster, two features not only need to be highly correlated, but their absolute values of correlations with other features should also be similar. We compare two variations of the algorithm in appendix A, which concludes that (ii) is more suitable for the topic modelling application because it is more context aware. Next sections utilize the (ii) formulation.

The agglomerative clustering procedure starts by assigning every row of $\hat{R}^{\star}$ into its cluster. On every iteration, it merges two of the most similar clusters, as measured by a selected similarity/dissimilarity metric, which is known as linkage criteria. On every iteration, the proposed configuration of the agglomerative clustering approach will merge the groups of columns that have the most similar correlations to all the columns, which is what we need, given the assumptions of biclustering.

In hierarchical clustering, the linkage criterion is a metric that defines the distance between two clusters A and B. The naming likely comes from the fact that it defines which clusters will be merged or linked together at every iteration of the algorithm. In this section, we discuss three of the most widely used linkage criteria [51], including single linkage, complete linkage and average linkage. We also provide the reasoning behind selecting one of them for the purposes of biclustering. Let $l(A, B)$ denote the distance between two clusters A and B (linkage) and $d(a, b)$ denote a distance (e.g. Euclidean distance) between any of the two cluster elements.
The single-linkage is defined as

$$l(A, B) = \min_{a \in A, b \in B} d(a, b) \tag{3.15}$$

The agglomerative clustering based on a single linkage will merge two clusters with a minimum distance between the closest observations. Therefore, it may not produce an

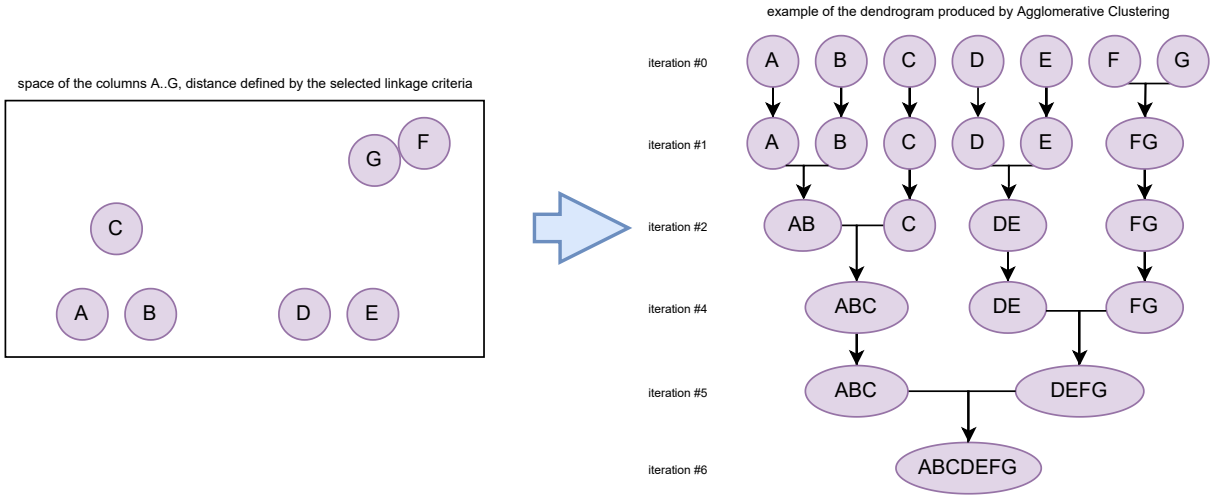

Figure 3.1: Example of the feature space with distance defined by a selected linkage criterion (*left*) and the result of iterations of agglomerative clustering (*right*)

optimal result in the presence of two columns that correlate significantly with each other, but other columns in two different clusters are completely uncorrelated. The complete linkage is defined in the following way

$$l(A, B) = \max_{a \in A, b \in B} d(a, b) \tag{3.16}$$

The procedure based on complete linkage will merge two clusters with a minimum distance between the furthest observations. This may not produce an optimal result when column c correlates with $\forall b \in B$, but it also has a large correlation with one other column $a \in A$ st $|A| = 1$. In this case, the result of the next iteration will be $A^{(t+1)} = \{a, c\}$ and $B^{(t+1)} = B^{(t)}$. The average linkage is defined as

$$l(A, B) = \frac{1}{|A||B|} \sum_{a \in A, b \in B} d(a, b) \tag{3.17}$$

We propose utilizing average linkage criteria for our purposes as this method results in clusters with the highest cohesion and diminishes the disadvantages of the method discussed above [48].

Chapter 4

Model

There is no closed-form solution for estimating the parameters of the Gaussian Mixture Model. Parameters of such models are usually estimated by leveraging the Expectation Maximization (EM) algorithm or its variants. This section extends the proposed parameter estimation procedure for bi-clustering with multiple mixtures of GMM. The biggest difference between the proposed procedure and the factor-analyzer model is in how the covariance matrices are estimated.

Let's denote $X = \{X_1, X_2, \dots X_N\}, X_i \in R^m$ to be a sample of N independent observations from a S-components Gaussian mixture. Let z_{is} for $i = 1, \dots, N$ and $s = 1, \dots, S$ be the latent variable with z_{is} determening the probability of X_i to be generated from s-th component. We start by defining a probability density function for a finite S-component GMM,

$$P(X_i|\theta) = \sum_{s=1}^{S} \pi_s f(X_i|\theta_s) \tag{4.1}$$

EM is an iterative algorithm that finds local maximum likelihood estimates of the model parameters θ. In one iteration, it alternates between the Expectation (E-step) and Maximization (M-step) steps. In the E-step, it computes $Q(\theta|\theta^t)$ the expected value of the log-likelihood function of θ given θ^t (model parameters computed during the previous iteration). In the M-step, the algorithm finds the estimate of θ that maximizes the expected log-likelihood from the E-step. For a S-component GMM the the expected value of the log-likelihood function is

$$Q(\theta|\theta^t) = E_{Z|X,\theta^t}[\log L(\theta; X, Z)] = \sum_{i=1}^{N} E_{Z_i|X_i,\theta^t}[\log L(\theta; X_i, Z_i)] = \tag{4.2}$$

$$\sum_{i=1}^{N}\sum_{s=1}^{S} P(z_{is}=1|X_i,\theta^t)[\log\pi_s - \frac{1}{2}\log|\Theta_s| - \frac{1}{2}(X_i-\mu_s)^T\Theta_s^{-1}(X_i-\mu_s)].$$

For bi-clustering purposes, we define the parametrization of Θ_s as in eq. (3.1).

Initializing latent variables. We start by splitting X into S clusters via KMeans algorithm [20]. This provides an initial binary assignment for the latent variable z_{is}.

M-step. The following θ updates are derived from eq. (4.2):

$$\pi_s = \frac{1}{N}\sum_{i=1}^{N} z_{is}, \quad \mu_s = \frac{1}{N}\sum_{i=1}^{N} z_{is}X_i \tag{4.3}$$

$$\Sigma_s = \frac{1}{N}\sum_{i=1}^{N} z_{is}(X_i-\mu_s)(X_i-\mu_s)^T \tag{4.4}$$

Note that Σ_s in eq. (4.4) is arbitrary. To convert it to block-diagonal Θ_s, we input the MLE of Σ_s to one of the three proposed column clustering estimators (e.g. greedy, hierarchical, numerical). Then we compute the estimate of the covariance matrix with K blocks Θ_s as:

$$\Theta_s = \sum_{k=1}^{K} D_{ks}\Sigma_s D_{ks}. \tag{4.5}$$

In the case of the common covariance matrix, we compute $\Sigma = \sum_s \pi_s\Sigma_s$ and repeat the steps outlined above.

E-step. We recompute the values of latent variable z_{is} as below:

$$z_{is} = P(z_{is}=1|X_i,\theta^t) = \frac{\pi_s f(X_i|\mu_s,\Theta_s)}{\sum_h^S \pi_h f(X_i|\mu_h,\Theta_h)}. \tag{4.6}$$

When the algorithm converges, we obtain row clustering by taking argmax over the mixture components of the latent variable Z and column clusters by looking at the matrices D_k.

As was established empirically, practical implementation of the Gaussian Mixture model algorithm requires the handling of multiple sources of numerical instabilities. Firstly, it is not guaranteed that the covariance matrices Σ_s estimated during the maximization step will have a full rank. Therefore Σ_s^{-1} might not exist. We introduce a diagonal regularization matrix $\Sigma_{reg} = 10^{-6}\times I$ to overcome this issue. We substitute $\Sigma_s = \Sigma_s + \Sigma_{reg}$.

Secondly, the exponentiation in the computation of $f(X_i|\mu_s,\Sigma_s)$ in eq. (4.6) sometimes results in infinite values. To fix this, we compute $G = \max_s \log f(X_i|\mu_s,\Sigma_s)$ and rewrite:

$$z_{is} = \frac{e^{\log\pi_s + \log f(X_i|\mu_s,\Theta_s) - G}}{\sum_s e^{\log\pi_s + \log f(X_i|\mu_s,\Theta_s) - G}}$$

Chapter 5

Simulations & Experiments

In this chapter, we aim to conduct experiments with both simulated and real-world data to establish the following:

- Out of the three proposed approaches for estimating the block-diagonal matrix, which one is the most accurate and efficient? Section 5.1 compares the execution time and accuracy of the three approaches while varying the number of observations in the dataset. We find that the hieracherical approach is the the most accurate and efficient one.

- How does hierarchical algorithm compares to other known covariance estimators? In the section 5.2 we experiment with recovering two different structures of the covariance matrix with our vs state-of-the-art approaches, including Ledoit-Wolf [33], Graphical Lasso [23], factor-analyzer UCUU [49] and MLE. We find that methods that artificially enforce block-diagonal covariance structure are significantly more accurate and the proposed method is indeed more robust than UCUU [49] for matrices with negative covariances.

- How accurate and efficient is the best-proposed approach in the presence of varying block-diagonal covariance structure? What are the limitations? Section 5.3 explores the accuracy of the hierarchical method under varying number of features and blocks in the covariance matrix. The approach exhibits $> 90\%$ accuracy for most of the settings. We discovered that the usage of the hierarchical algorithm is limited when estimating high number of small blocks.

- How accurate and efficient is the proposed approach in the clustering scenario? In the section 5.4 we compare the performance of our algorithm to the performance of two clustering (KMeans [20], GMM [45]) and two bi-clustering (Spectral Co-clustering [16], factor-analyzer UCUU [49]) approaches. For every tested dataset, we find that our algorithm is comparable in accuracy and sometimes outperforms others.

5.1 Accuracy and Efficiency Comparison of the Proposed Approaches

In this section, we investigate the accuracy and efficiency of the three proposed approaches to the problem of recovering column clusters D_k. The experiment setup is similar to other proposed in the literature [49]. For every experiment, the input data $X \in R^{N\times 8}$ is generated from one component Gaussian Mixture Model with $\mu = \begin{bmatrix}0 & 1 & 2 & 3 & 4 & 5 & 6 & 7\end{bmatrix}$. Fig. 2.1 displays two options for the covariance matrix that are considered in the experiments: (1) with non-negative entries only; (2) allowing negative entries. The choice of covariance matrices is motivated by restrictions for off-diagonal elements to be strictly non-negative utilized commonly in the literature [53, 49, 36]. In this work, we aim to recover a covariance matrix of the general structure, including the possibility of negative off-diagonal elements. Note that both matrices at Fig. 2.1 have a natural block-diagonal structure, which provides us with ground truth for column cluster indicator matrices D_k. For example, for both Σ_1 and Σ_2 we have $D_1 = \text{diag}\begin{pmatrix}1 & 1 & 1 & 0 & 0 & 0 & 0 & 0\end{pmatrix}$.

Fig. 5.1 demonstates the result of our experiments with data generated from $\mathcal{N}(\mu, \Sigma_1)$. We vary the number of data points to understand the impact on both clustering accuracy and execution time of the proposed methods. For every size of dataset X, we run the experiment 200 times. The accuracy is calculated as the total number of times when an algorithm returned correct column clusters D_1, D_2, D_3 divided by the total number of attempts. Fig. 5.1 (*right*) demonstrates the average execution time in milliseconds over 200 runs together with it's standard deviation.

From Fig. 5.1, it is clear that the hierarchical estimator of column clusters D_k is the most accurate and efficient. It gets to 100% starting from 50 data points in the training set and demonstrates solid accuracy $> 90\%$ even with 20 examples. The numerical estimator benefits from the increasing dataset size. However, the time it needs to converge grows as well. In this experiment, the greedy estimator proved to be the least efficient and the least accurate.

Fig. 5.2 demonstrates the results of the analogous experiment with X generated from

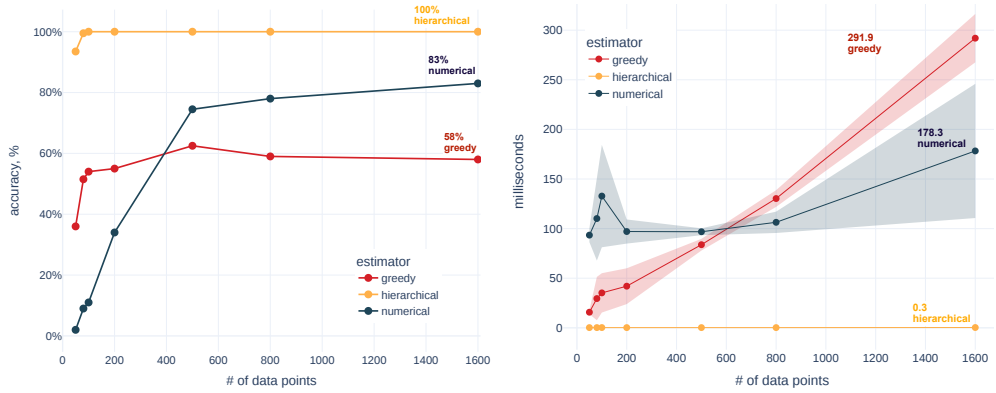

Figure 5.1: Comparison of accuracy & execution time over 200 runs for estimating D_k for block-diagonal 8×8 covariance matrix with 3 blocks, **positive entries only**. (*Left*) The accuracy of the proposed approaches is given a number of the data points in the generated dataset X. (*Right*) Execution time of the proposed approaches.

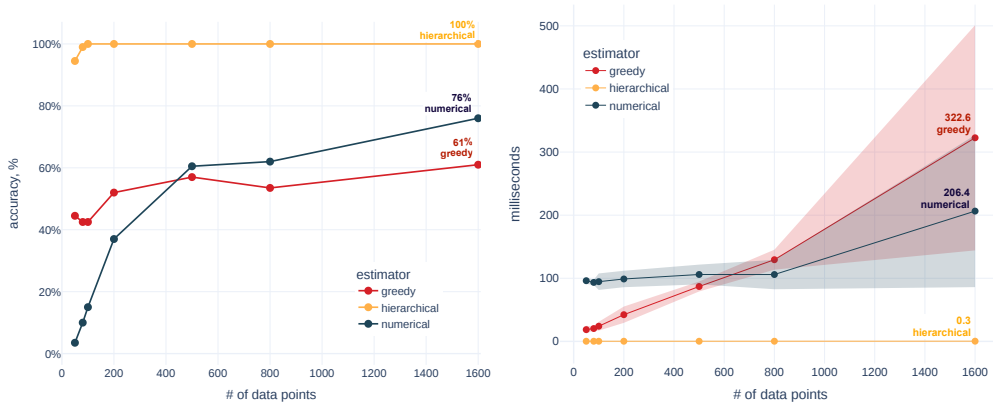

Figure 5.2: Comparison of accuracy & execution time for estimating D_k for block-diagonal 8×8 covariance matrix with 3 blocks, **negative entries allowed**. (*Left*) The accuracy of the proposed approaches given a number of the data points in the generated dataset X. (*Right*) Execution time of the proposed approaches.

$\mathcal{N}(\mu, \Sigma_2)$. Interestingly, the execution time of both numerical and greedy estimators increases. While the numerical estimator demonstrates degraded accuracy, the greedy one improves in the presence of negative covariances or a greater variety of values within a covariance block. At the same time, the hierarchical approach leads in terms of both efficiency and accuracy, and it can correctly cluster columns even for the matrix with negative

covariances. For our next experiment, we proceed with the hierarchical approach.

5.2 Recovering Block-Diagonal Covariance Matrix & Comparison with the Literature

In the previous experiments, we focused on the question of how accurate the detection of blocks in the block-diagonal covariance matrix Σ is. In this experiment, we aim to estimate the final block-diagonal covariance matrix Σ with the hierarchical method and compare it with other methods commonly utilized in the literature[49, 33, 22].

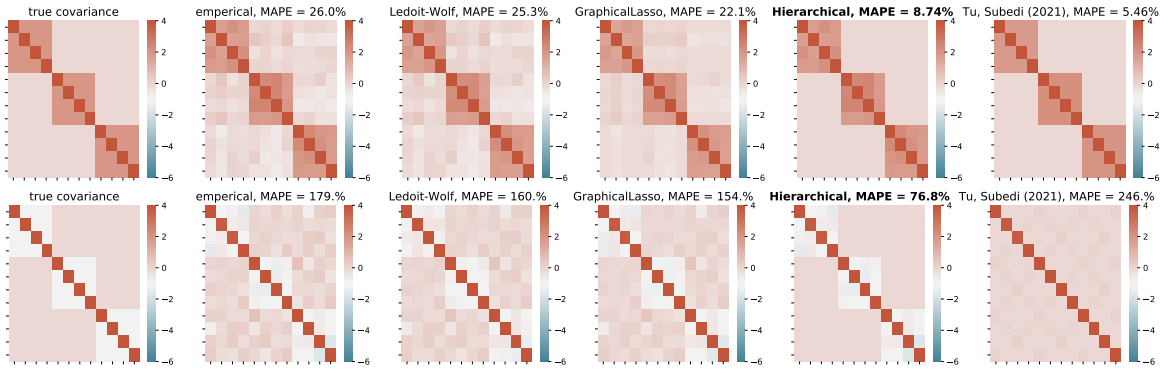

Figure 5.3: Recovering block-diagonal covariance matrix with only positive (*top*) / negative (*bottom*) off-diagonal entries with five methods including MLE (empirical), Ledoit-Wolf, Graphical Lasso, Hierarchical (proposed) & UCUU method from Tu, Subebi 2021 [49]

Fig. 5.3 (first column) shows two block-diagonal covariance matrices used to generate datasets $X \in R^{300\times 12}$ with 300 observations of 12 features. We also used $\mu = \begin{pmatrix}1 & 2 & \dots & 12\end{pmatrix}$ in both cases. The figure demonstrates estimates of Σ produced by five approaches and corresponding MAPE values (Mean Absolute Percentage Error). For every estimate $\hat{\Sigma}$ we compute MAPE % measure as following:

$$100 \times \frac{\sum_{i,j} |\sigma_{ij} - \hat{\sigma}_{ij}|}{\sum_{i,j} \sigma_{ij}}. \tag{5.1}$$

Our main conclusions from this experiment include: (i) artificially enforcing block-diagonal covariance structure is justified for the problems when this covariance structure is expected as such (hierarchical, UCUU) are at least two times more accurate (in terms of MAPE)

than others; (ii) the proposed hierarchical method is robust in the presence negative off-diagonal covariances, while UCUU [49] is not designed for such case.

In this example, MAPE is only helpful for relative algorithm accuracy comparison but not for absolute estimation quality assessment. It allows the comparisons of the form "method X is 2 times more accurate than Y". However, we do not deduce that all methods are highly inaccurate for the covariance matrix with negative off-diagonal entries because the denominator of equation 5.1 is small. Therefore even small errors will result in large percentages.

5.3 Impact of the Covariance Block Structure on the Performance of Hierarchical Estimator

The previous sections concluded that the hierarchical estimator of column clusters D_k is the most accurate and efficient among the three proposed ones. It quickly reached close to 100% clustering accuracy, even with small sample sizes (see Fig. 5.1).

This experiment aims to investigate the limitations of the hierarchical estimator by applying it to the problems of growing complexity and measuring the resulting accuracy of column clustering. In a nutshell, the complexity of the estimation of D_k from a data sample X generated by $\mathcal{N}(\mu, \Sigma)$ (where Σ is block-diagonal) is influenced by: (i) the size of X or the number of rows in the input dataset; (ii) the dimensionality of μ, Σ or the number of columns in the dataset (m); (iii) number of block in Σ or the actual number of column clusters (K).

To set up the experiment, we fix (i) - the number of rows in the dataset to $N = 50$, and we will vary $(ii), (iii)$ to understand its impact on the estimator's accuracy. This choice is motivated by common real-life scenarios (e.g. genomics, natural language processing etc.) when the number of observations is often limited compared to the number of potential features.

We test the accuracy of the hierarchical estimator given three options for the number of columns in the input dataset. For simplicity of comprehension, they will be referenced as low, medium and high and summarized in table 5.1. We also define three levels (low-medium-high) for the number of blocks in the covariance matrix Σ. The number of column clusters at every level is proportional to the level of dataset dimensionality. For example, the number of blocks in the low columns-low blocks is $3 \times 2^0 = 3$, while the number of blocks in medium columns-low blocks is $3 \times 2^1 = 6$. The calculations are summarized in the table 5.1.

level	low= 0	medium= 1	high= 2
m	24	96	384
K	$3 \times 2^{\text{column level}}$	$4 \times 2^{\text{column level}}$	$8 \times 2^{\text{column level}}$
N	50	50	50

Table 5.1: Summary of different combinations of the number of columns and the number of true column clusters in Σ. The number of column clusters (blocks) is proportional to the level of data dimensionality (e.g. low number of blocks is 3 for the data with 24 columns and 12 for the high-dimensional data with 384 columns)

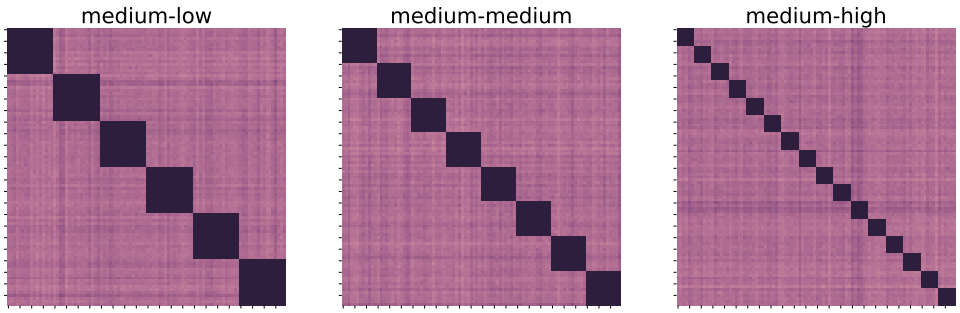

Figure 5.4: Examples of covariance matrices Σ_i used to generate data for the simulation study. In this example, the number of columns is fixed to medium=96 (table 5.1), and the number of blocks varies from 6 (low) to 16 (high). Note the added noise.

For every level of columns and number of blocks, we run the experiment 500 times. Every run i, we sample μ_i randomly from the multivariate continuous uniform distribution on the half-open interval $[0, 1)$. We generate every block B_{ij} of the block-diagonal Σ_i independently as $B_{ij} = A_{ij}^T A_{ij}$. When Σ_i is constructed, we add random noise $0.5 E_i^T E_i$ to bring this simulation closer to real-world scenarios of imperfect covariance matrix estimators. Entries of A_{ij} and E_i are independently generated from the continuous uniform distributions $U(1, 2)$ and $U(0, 1)$ correspondingly.

Fig. 5.5 demonstrated the accuracy of the hierarchical estimator for different problem complexities in two scenarios: known vs an unknown actual number of column clusters (blocks of the covariance matrix). In the second case, we pick the number of clusters that produces the maximum silhouette score with the search range $n \pm 2\sqrt{m}$, where n is a true

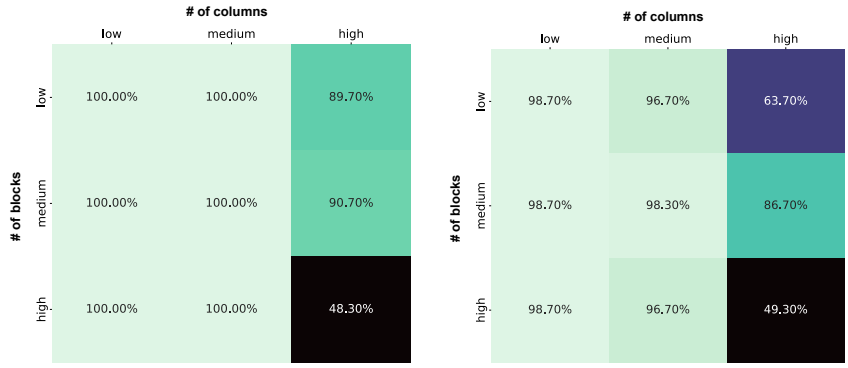

Figure 5.5: Accuracy of the hierarchical estimator of column clusters D_k for different problem complexities defined by the dimensionality of the dataset(low-high) and the true number of blocks in the covariance matrix Σ (number of column clusters). Tested for the scenario of known (*left*) and unknown (*right*) n clusters

number of clusters, m is a dimensionality of the data.

Fig. 5.5 proves that the proposed estimator is accurate for any covariance structure of the low- and medium-dimensional data and mostly even high-dimensional one (although the number of rows in the dataset is fixed to be 50). At the same time, the utility of the estimator is limited for very heterogeneous high-dimensional data (high-high scenario). We also see that accuracies for the scenarios of known vs unknown number of blocks are very similar, which signifies the usefulness of the estimator even in the case when the number of column clusters is unknown.

5.4 Clustering Performance

In this experiment, we test the row clustering performance of the proposed bi-clustering approach - the Gaussian Mixture Model with a block-diagonal covariance matrix estimated leveraging the hierarchical method. We compare the results of the proposed approach with two other bi-clustering methods [49, 16] and KMeans [20] and vanilla Gaussian Mixture Model [45]. We use KMeans, GMM and Spectral Co-Clustering implementations with their default parameters from the sklearn library [43]. Additionally, we implemented the factor-analyzer UCUU model [49] in Python.

This section reports on two sets of experiments. Firstly, we test the algorithms on well-known low-dimensional datasets such as Wine [13], Olive [21] and Ecoli [41]. Secondly,

we compare the algorithms' performance on the high-dimensional genomics classification problems, including Alon [2] and Golub [26]. We scale the input data before clustering to the mean 0 and variance equal to 1. We compare the algorithm's using the adjusted rand index (ARI) [54], accuracy and computational time.

The results of the first set of experiments are summarized in the table 5.2 below. The proposed algorithm compares with the known methods in terms of ARI and accuracy and even outperforms others on the Ecoli dataset, but it is almost 10× slower in terms of execution time than most of the listed approaches. However, the most similar algorithm to our proposed method is the UCUU model which runs 400× slower than KMeans, GMM and Spectral Co-Clustering.

We conclude that the proposed approach has similar accuracy compared to known row clustering algorithms while having an advantage in terms of lower execution time compared to the bi-clustering UCUU [49] model, which is very similar in utility.

Datasets	Wine			Olive			Ecoli		
	ARI	%, acc	time (sec)	ARI	%, acc	time (sec)	ARI	%, acc	time (sec)
KMeans	0.897	96.6%	0.015	0.448	76.5%	0.047	0.509	65.2%	0.036
GMM	0.831	92.7%	**0.009**	**0.601**	78.1%	**0.019**	0.646	74.4%	**0.015**
UCUU [49]	**0.948**	**98.3**%	4.320	0.517	79.1%	19.62	–	–	–
Proposed	0.945	98.3%	0.120	0.574	**80.4**%	0.169	**0.656**	**76.2**%	0.138
Spectral Co-Clustering	0.738	90.9%	0.020	0.237	57.3%	0.042	0.394	56.4%	0.048

Table 5.2: Average adjusted rand index (ARI), accuracy and execution time over 10 fits of different algorithms on given datasets.

In the second experiment, we compare the performance of the same pool of algorithms on a more complex task of high-dimensional data clustering.

The preprocessed version of Alon [2, 38] data consists of 42 tumorous and 22 normal observations of 461 genes. Unfortunately, we were unable to run the UCUU method [49] on the full set of features because of (*i*) increasing computational time; (*ii*) author's note that this algorithm is not designed for the scenario when the number of features is significantly higher than the number of observations. Therefore, we created Alon_{100} by subsampling 100 features with ANOVA F-test [19].

The first and third columns in the table 5.3 summarize the algorithm's performance on the restricted and full datasets, respectively. In the first case, the results of the proposed method, UCUU and KMeans, are identical. This can be explained by the fact that only

Datasets	Alon_{100}		Golub_{100}		Alon-full		Golub-full	
	ARI	%, acc	ARI	%, acc	ARI	%, acc	ARI	%, acc
KMeans	**0.592**	**88.7**%	0.889	97.2%	−0.003	55.1%	0.162	70.8%
GMM	0.553	87.4%	0.889	97.2%	**0.042**	**58.3**%	0.161	70.8%
UCUU [49]	**0.592**	**88.7**%	0.889	97.2%	–	–	–	–
Proposed	**0.592**	**88.7**%	0.889	97.2%	0.001	55.4%	**0.212**	**73.6**%
Spectral Co-Clustering	0.450	83.8%	0.837	95.8%	−0.006	54.8%	0.185	72.2%

Table 5.3: Adjusted rand index (ARI) and accuracy of different algorithms on given high-dimensional datasets and their preprocessed versions with 100 features selected with ANOVA F-test[19].

100 best features were selected, not leaving much noise in the data. Unfortunately, the performance of all methods degraded significantly the full Alon dataset, with the GMM method achieving maximum accuracy of 58.3%.

Golub [26] comprises 2030 gene expressions of 72 patients, 47 of them are diagnosed with acute lymphoblastic leukemia and 25 with acute myeloid leukemia. As previously, we construct Golub_{100} with a restricted number of features selected with ANOVA F-test [19]. The results of this experiment are similar to those above: four of the five methods demonstrate the identical performance of the subsampled dataset, and the performance of all the methods degrades on the full one. Golub data has $5\times$ more columns than Alon, and we now see that both bi-clustering algorithms (proposed and Spectral Co-Clustering) perform better than row clustering ones. This can be attributed to the bi-clustering ability to homogenize high-dimensional and heterogeneous data.

Chapter 6

Application to Topic Modelling

In this chapter, we apply the proposed bi-clustering algorithm to the topic modelling problem and discuss the results. We aim to demonstrate that the proposed algorithm is a powerful tool for summarizing customer reviews data.

We test our algorithm on two real-world datasets of customer reviews. The first is Trip Advisor hotel[1] data that consists of 20491 unique free-text hotel reviews accompanied by a numerical rating (from 1 to 5). The second one includes reviews and ratings of three Disneyland[2] branches. We use only a subset of the data that reviews Disneyland Paris (13630 free-text responses). The language of reviews in both datasets is English.

For better explainability of the resulted topics and column groupings, we decided to apply the bi-clustering algorithm to the document-terms matrix instead of document-embeddings [27] data type. Testing the proposed algorithm with semantic text embeddings is left for future work.

The entries of the document-terms matrix are term frequency-inverse document frequency scores (TF-IDF) [1]. Let us define $\text{tf}(t, d)$ as a relative frequency of the term t in some document d. Then, the inverse document frequency of the term t given a collection D of N documents is computed as $\text{idf(t, D)} = \log \frac{N}{|d \in D : t \in d|}$.

To simplify, the term frequency measures how frequent a term is for a given document. The inverse document frequency measures how rare this term is in the overall collection of documents. We define a TF-IDF score of a term t int he document d as $\text{tf-idf}(t, d) = \text{tf}(t, d)\text{idf(t, D)}$. The score is maximized when a specific term is frequent in a

[1] https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews
[2] https://www.kaggle.com/datasets/arushchillar/disneyland-reviews

given document but uncommon for other documents. The TF-IDF score of the stop words is close to zero, as, by definition, stop words have a high probability of being present in every document in the collection.

Large collections of documents may include over a thousand unique words. Therefore, before fitting the bi-clustering model, we select only 1000 features with the highest average TF scores. Additionally, we utilize `SelectKBest` from the sklearn [43] Python library to select 400 terms that are the most predictive of the review rating.

We used an elbow method and a silhouette score metric to select the optimal number of row clusters for every dataset. The proposed algorithm also inputs the number of column groupings. We discovered that small values k for the number of groups result in most columns being grouped in one cluster, with other clusters consisting of $2-3$ columns. This is expected with hierarchical clustering. Therefore, we suggest selecting $k > 15$ for the purposes of topic modelling. Additionally, we utilized a family of Gaussian models with a shared covariance matrix. This simplifies the interpretation of feature groupings.

Fig. 6.1 and Fig. 6.2 demonstrate the column and row groupings for the Trip Advisor and Disneyland Paris data. The heatmap entries are computed as the average TF-IDF score of terms inside the column grouping A row cluster B divided by average TF-IDF scores of A among all row clusters and multiplied by 100%. This can be interpreted as how much "more important" a column cluster A is for a row cluster B than average.

From Fig. 6.1, we infer that lower hotel ratings may be attributed to the cigarette smell, rude managers, dirty conditions, problems with the check desk and unexpected credit card charges. The highest ratings are correlated with access to public transport or the hotel's location within walking distance of the city's landmarks, friendly hotel staff and beautiful decorations of the premises. Fig. 6.2 demonstrates that both feature clusters of food (ice cream and chips-burgers-fries) are highly associated with the lowest ratings of Disneyland Paris. It may signal that the food quality is lower than expected. At the same time, people are unhappy with closed attractions, rude staff, pushing and perhaps the fact that most people in Paris speak exclusively French.

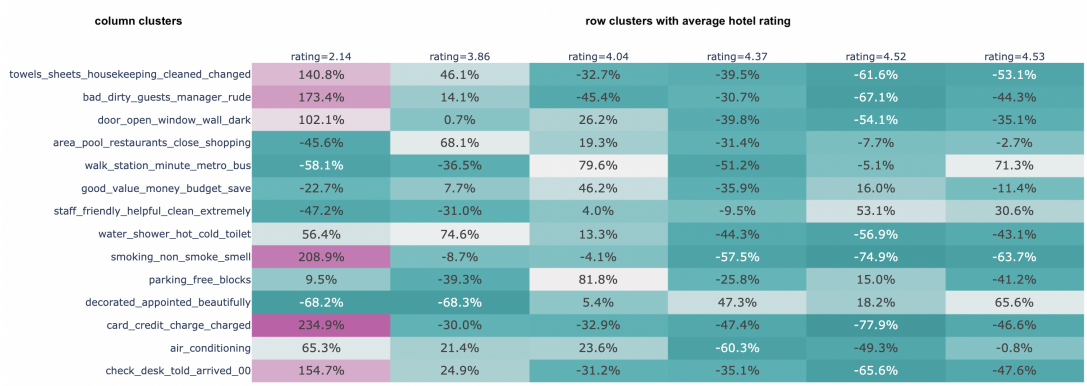

Figure 6.1: Resulting column and row clusters from the Trip Advisor hotel reviews. Entries of the heatmap represent how much "more important" a given column cluster is for a given row cluster than average.

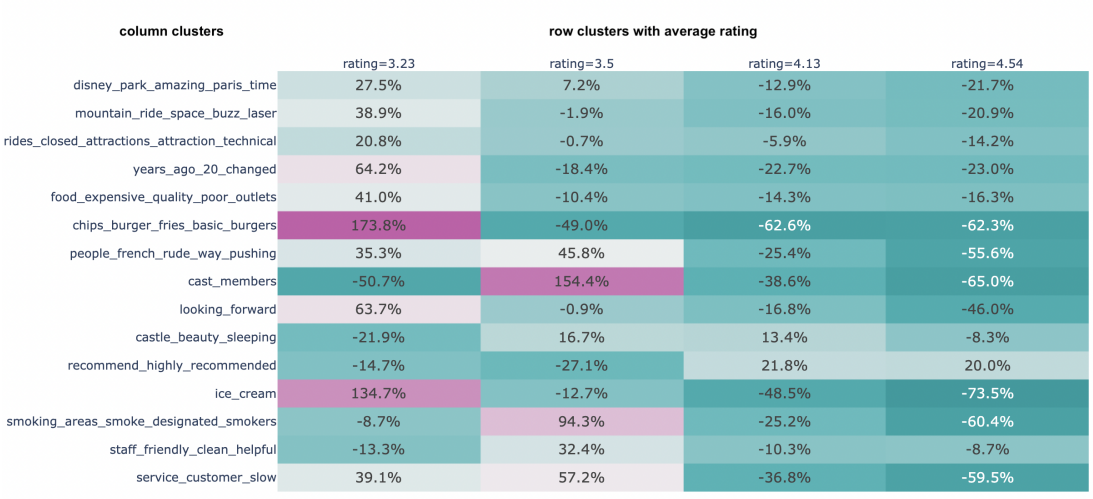

Figure 6.2: Resulting column and row clusters from the Disneyland Paris reviews. Entries of the heatmap represent how much "more important" a given column cluster is for a given row cluster than average.

Chapter 7

Conclusions

In this work, we have proposed a bi-clustering algorithm based on a finite mixture of multivariate Gaussian models. We demonstrated (table 5.2) that our approach has a comparable clustering performance to the state-of-the-art algorithms in the field. We provided an example of applying the proposed bi-clustering to real-world topic modelling problems.

The following is left for future work:

- Proposed bi-clustering algorithm's topic modelling performance with the semantic embedding data. We assume it may result in higher-quality row clusters.
- Topic modelling based on clustering of separate subsets of the data features identified by the output of bi-clustering. We think it will be equivalent to pulling distinct homogeneous views from the multi-view data.
- Propose an algorithm for selecting the optimal number of column groupings such that the resulting grouping sizes are better balanced.

References

[1] Akiko Aizawa. "An information-theoretic perspective of tf–idf measures". In: *Information Processing & Management* 39.1 (2003), pp. 45–65.

[2] Uri Alon et al. "Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJBroad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc Natl Acad Sci USA 96: 6745-6750". In: *Proceedings of the National Academy of Sciences of the United States of America* 96 (July 1999), pp. 6745–50. DOI: 10.1073/pnas.96.12.6745.

[3] Maria-Florina Balcan, Yingyu Liang, and Pramod Gupta. "Robust hierarchical clustering". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3831–3871.

[4] Doug Beeferman and Adam Berger. "Agglomerative clustering of a search engine query log". In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000, pp. 407–416.

[5] Amir Ben-Dor et al. "Discovering local structure in gene expression data: the order-preserving submatrix problem". In: *Proceedings of the sixth annual international conference on Computational biology*. 2002, pp. 49–57.

[6] Sven Bergmann, Jan Ihmels, and Naama Barkai. "Iterative signature algorithm for the analysis of large-scale gene expression data". In: *Physical review E* 67.3 (2003), p. 031902.

[7] Pavel Berkhin. "A survey of clustering data mining techniques". In: *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[8] Parmeet Bhatia, Serge Iovleff, and Gerard Govaert. "blockcluster: An R Package for Model Based Co-Clustering". In: *Journal of Statistical Software* 76 (Dec. 2014). DOI: 10.18637/jss.v076.i09.

[9] David Blei, Andrew Ng, and Michael Jordan. "Latent Dirichlet Allocation". In: vol. 3. Jan. 2001, pp. 601–608.

[10] Baptiste Broto et al. *Block-diagonal covariance estimation and application to the Shapley effects in sensitivity analysis*. 2019. DOI: 10.48550/ARXIV.1907.12780. URL: https://arxiv.org/abs/1907.12780.

[11] Stanislav Busygin, Oleg Prokopyev, and Panos M Pardalos. "Biclustering in data mining". In: *Computers & Operations Research* 35.9 (2008), pp. 2964–2987.

[12] Yizong Cheng and George M Church. "Biclustering of expression data." In: *Ismb*. Vol. 8. 2000. 2000, pp. 93–103.

[13] Paulo Cortez et al. "Modeling wine preferences by data mining from physicochemical properties". In: *Decision support systems* 47.4 (2009), pp. 547–553.

[14] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[15] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

[16] Inderjit S. Dhillon. "Co-Clustering Documents and Words Using Bipartite Spectral Graph Partitioning". In: KDD '01. San Francisco, California: Association for Computing Machinery, 2001, pp. 269–274. ISBN: 158113391X. DOI: 10.1145/502512.502550. URL: https://doi.org/10.1145/502512.502550.

[17] Edwin Diday and JC Simon. "Clustering analysis". In: *Digital pattern recognition*. Springer, 1976, pp. 47–94.

[18] E Erica, BD Handari, and GF Hertono. "Agglomerative clustering and genetic algorithm in portfolio optimization". In: *AIP Conference Proceedings*. Vol. 2023. 1. AIP Publishing LLC. 2018, p. 020217.

[19] Betty J. Feir-Walsh and Larry E. Toothaker. "An Empirical Comparison of the Anova F-Test, Normal Scores Test and Kruskal-Wallis Test Under Violation of Assumptions". In: *Educational and Psychological Measurement* 34.4 (1974), pp. 789–799. DOI: 10.1177/001316447403400406. eprint: https://doi.org/10.1177/001316447403400406. URL: https://doi.org/10.1177/001316447403400406.

[20] Edward W Forgy. "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". In: *biometrics* 21 (1965), pp. 768–769.

[21] Massimo Forina and E Tiscornia. "Pattern-recognition methods in the prediction of Italian olive oil origin by their fatty-acid content". In: *Annali di Chimica* 72.3-4 (1982), pp. 143–155.

[22] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Sparse inverse covariance estimation with the graphical lasso". In: *Biostatistics* 9.3 (Dec. 2007), pp. 432–441. ISSN: 1465-4644. DOI: 10.1093/biostatistics/kxm045. eprint: https://academic.oup.com/biostatistics/article-pdf/9/3/432/17742149/kxm045.pdf. URL: https://doi.org/10.1093/biostatistics/kxm045.

[23] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Sparse inverse covariance estimation with the graphical lasso". In: *Biostatistics* 9.3 (2008), pp. 432–441.

[24] M. P. B. Gallaugher, C. Biernacki, and P. D. McNicholas. *Parameter-wise co-clustering for high-dimensional data*. 2018. DOI: 10.48550/ARXIV.1808.08366. URL: https://arxiv.org/abs/1808.08366.

[25] Maruf Gogebakan and Hamza Erol. "A new semi-supervised classification method based on mixture model clustering for classification of multispectral data". In: *Journal of the Indian Society of Remote Sensing* 46.8 (2018), pp. 1323–1331.

[26] T. R. Golub et al. "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring". In: *Science* 286.5439 (1999), pp. 531–537. DOI: 10.1126/science.286.5439.531. eprint: https://www.science.org/doi/pdf/10.1126/science.286.5439.531. URL: https://www.science.org/doi/abs/10.1126/science.286.5439.531.

[27] Maarten Grootendorst. *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*. 2022. DOI: 10.48550/ARXIV.2203.05794. URL: https://arxiv.org/abs/2203.05794.

[28] Jiajun Gu and Jun S Liu. "Bayesian biclustering of gene expression data". In: *BMC genomics* 9.1 (2008), pp. 1–10.

[29] John A Hartigan. "Direct clustering of a data matrix". In: *Journal of the american statistical association* 67.337 (1972), pp. 123–129.

[30] Raymond Hemmecke et al. "Nonlinear Integer Programming". In: *50 Years of Integer Programming 1958-2008*. Springer Berlin Heidelberg, Nov. 2009, pp. 561–618. DOI: 10.1007/978-3-540-68279-0_15.

[31] Kamran Khan et al. "DBSCAN: Past, present and future". In: *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*. IEEE. 2014, pp. 232–238.

[32] Yuval Kluger et al. "Spectral Biclustering of Microarray Data: Coclustering Genes and Conditions". In: *Genome research* 13 (May 2003), pp. 703–16. DOI: 10.1101/gr.648603.

[33] Olivier Ledoit and Michael Wolf. "A well-conditioned estimator for large-dimensional covariance matrices". In: *Journal of Multivariate Analysis* 88.2 (2004), pp. 365–411. ISSN: 0047-259X. DOI: https://doi.org/10.1016/S0047-259X(03)00096-4. URL: https://www.sciencedirect.com/science/article/pii/S0047259X03000964.

[34] Sara C Madeira and Arlindo L Oliveira. "Biclustering algorithms for biological data analysis: a survey". In: *IEEE/ACM transactions on computational biology and bioinformatics* 1.1 (2004), pp. 24–45.

[35] Sara C Madeira et al. "Identification of regulatory modules in time series gene expression data using a linear time biclustering algorithm". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 7.1 (2008), pp. 153–165.

[36] Francesca Martella, Marco Alfò, and Maurizio Vichi. "Biclustering of Gene Expression Data by an Extension of Mixtures of Factor Analyzers". In: *The International Journal of Biostatistics* 4 (2008).

[37] Leland McInnes, John Healy, and Steve Astels. "hdbscan: Hierarchical density based clustering." In: *J. Open Source Softw.* 2.11 (2017), p. 205.

[38] Paul D. McNicholas and Thomas Brendan Murphy. "Parsimonious Gaussian mixture models". In: *Statistics and Computing* 18 (2008), pp. 285–296.

[39] Glenn W Milligan and Martha C Cooper. "Methodology review: Clustering methods". In: *Applied psychological measurement* 11.4 (1987), pp. 329–354.

[40] Fionn Murtagh and Pierre Legendre. "Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion?" In: *Journal of classification* 31.3 (2014), pp. 274–295.

[41] Kenta Nakai and Minoru Kanehisa. "UCI machine learning repository". In: *URL: https://archive. ics. uci. edu/ml/datasets/ecoli* (1991).

[42] Victor A Padilha and Ricardo JGB Campello. "A systematic comparative evaluation of biclustering techniques". In: *BMC bioinformatics* 18.1 (2017), pp. 1–25.

[43] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[44] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. DOI: 10.48550/ARXIV.1908.10084. URL: https://arxiv.org/abs/1908.10084.

[45] Douglas A Reynolds. "Gaussian mixture models." In: *Encyclopedia of biometrics* 741.659-663 (2009).

[46] Daniel Rugeles et al. "Biclustering: An application of dual topic models". In: *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM. 2017, pp. 453–461.

[47] Carson Sievert and Kenneth Shirley. "LDAvis: A method for visualizing and interpreting topics". In: *Proceedings of the workshop on interactive language learning, visualization, and interfaces*. 2014, pp. 63–70.

[48] Robert R. Sokal and Charles Duncan Michener. "A statistical method for evaluating systematic relationships". In: *University of Kansas science bulletin* 38 (1958), pp. 1409–1438.

[49] Wangshu Tu and Sanjeena Subedi. "A family of mixture models for biclustering". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 15 (2022), pp. 206–224.

[50] Ike Vayansky and Sathish AP Kumar. "A review of topic modeling methods". In: *Information Systems* 94 (2020), p. 101582.

[51] Vijaya, Shweta Sharma, and Neha Batra. "Comparative Study of Single Linkage, Complete Linkage, and Ward Method of Agglomerative Clustering". In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. 2019, pp. 568–573. DOI: 10.1109/COMITCon.2019.8862232.

[52] Hanna M Wallach. "Topic modeling: beyond bag-of-words". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 977–984.

[53] Monica Wong, David Mutch, and Paul McNicholas. "Two-way learning with one-way supervision for gene expression data". In: *BMC Bioinformatics* 18 (Mar. 2017). DOI: 10.1186/s12859-017-1564-5.

[54] Shaohong Zhang, Hau-San Wong, and Ying Shen. "Generalized adjusted rand indices for cluster ensembles". In: *Pattern Recognition* 45.6 (2012), pp. 2214–2226.

[55] Yi Zhang et al. "Gaussian Mixture Model Clustering with Incomplete Data". In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 17 (Mar. 2021), pp. 1–14. DOI: 10.1145/3408318.

Appendix A

Hierarchical Approach Variations

This chapter compares two formulations of the hierarchical estimator suggested in the section 3.4, based on the features correlation matrix $\hat{R}^\star$ eq. (3.14): (i) $D = 1 - \hat{R}^\star$ as distance matrix to the clustering algorithm; (ii) cluster $\hat{R}^\star$ directly, assuming rows of $\hat{R}^\star$ to be the representation vectors of the dataset features.

We assume that two algorithms will have similar performance on the simple illustrative examples of covariance matrices from section 5.1 and section 5.2. Therefore, we will compare the performance of two methods in the setting of the experiments described in section 5.3, on the clustering and topic modelling task.

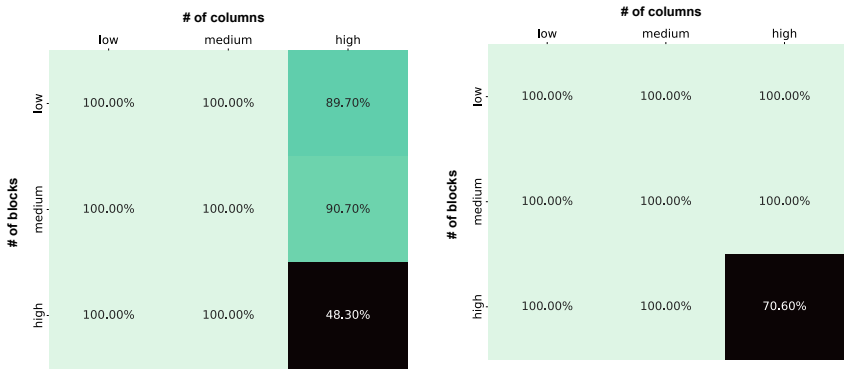

Figure A.1: Accuracy of the (i) (*right*) and (ii) (*left*) estimators of column clusters D_k for different problem complexities defined by the dimensionality of the dataset(low-high) and number of blocks in the covariance matrix Σ. Tested for the scenario of known K.

Fig. A.1 demonstrates that (i) performs significantly better on the simulations - in the

presence of the true block-diagonal covariance matrix that used to generate the simulation data. However, this is not often the case for real-world datasets. From table A.1 we conclude that (i) and (ii) have very comparable accuracy, ARI and execution time on clustering task with multiple real-world datasets. Fig. A.2 summarizes topics detected in TripAdvisor hotel reviews with (i) method, which seem less coherent than those generated with (ii) on Fig. 6.1. We conclude that (i) is more suitable for the tasks when the true covariance matrix is known to be block-diagonal, while (ii) is more suitable for the topic modelling applications because it is more context aware (two features are clustered together not only by correlations with each other, but takes into consideration their correlations with other features).

Datasets	Wine			Olive			Ecoli		
	ARI	%, acc	time (sec)	ARI	%, acc	time (sec)	ARI	%, acc	time (sec)
Proposed (i)	0.911	97.1%	0.182	**0.608**	**81.9**%	0.226	0.653	76.1%	0.170
Proposed (ii)	**0.945**	**98.3**%	**0.120**	0.574	80.4%	**0.169**	**0.656**	**76.2**%	**0.138**

Table A.1: Average adjusted rand index (ARI), accuracy and execution time over 10 fits of two algorithms on given datasets.

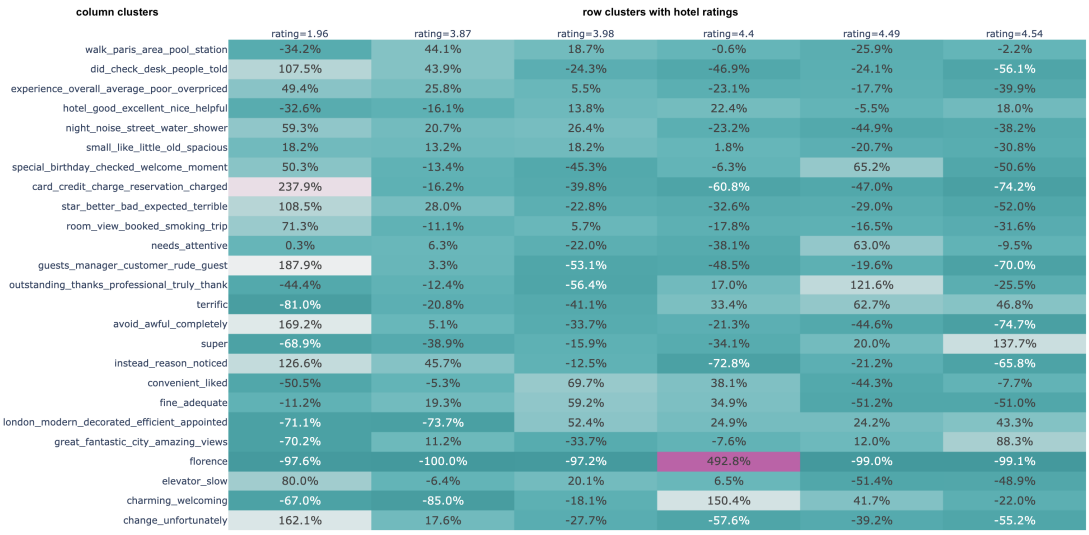

Figure A.2: Resulting column and row clusters from the Trip Advisor hotel reviews obtained with (i) of the hierarchical approach. Entries of the heatmap represent how much "more important" a given column cluster is for a given row cluster than average.