École polytechnique fédérale de Lausanne

Semester project spring 2023

Master in Mathematics

Multilayer network clustering

Author:
Wael Gafaiti

Supervisor:
Youngseok Song
Sofia Olhede

Contents

1 Introduction

Clustering has been the subject of scientific research since the 1940s. In fact, such analyses are useful in a lot of applied fields, as biology (see, e.g., the excellent article ofAltman and Krzywinski [2017]), psychology, etc. These efforts result in the discovery of the K-means algorithm in the 1950s. It is a famous algorithm which retrieves clusters of a data set given a specific distance and the number of clusters.

The most common data structure for this algorithm are data set in $\mathbb{R}^p$ with p the number of features. However, it happens sometimes that the structure is not common at all, as networks just to name one. In this work, we focus on clustering over a network. Instead of observing a data set over $\mathbb{R}^p$, we observe interactions in multiple contexts, which correspond to layers of a network, between individuals. Community clustering on such networks arise in many areas of research and application at the end of the 90s. For example, domains, such as brain connectivity study, cybersecurity or evaluation and prediction of social relationships within large group of individuals study, apply community clustering to carry out their analyses. Even today, new methods frameworks and algorithms arise in the scientific literature over this subject, which means that this field is still in expansion.

Unfortunately, it is impossible to get a look at all the literature at once (see, e.g, the excellent book of Bianconi [2018]). That is why we will only focus on two frameworks, *Multi-layer Stochastic Block Model (MSBM)* and *Mixture Multi-layer Stochastic Block Model (MMSBM)*, and three approaches to lead community clustering. These approaches were respectively developed by Lei et al. [2019], Jing et al. [2020] and Fan et al. [2021]. We recall that the goal is, given an observable multi-layer network, we would like to classify the individuals in communities. The MMSBM framework goes even further since we would like also to retrieve the membership community for each layer of the network, and, within a such community of layers, derive the community membership for the individuals.

Such analyses leads to two main interrogations: how to represent a network? And, how to define a community? To represent a network, we will use the graph representation, i.e. each individuals of the network is represented by a node, and every interaction between a pair of individuals is represented by an undirected edge and all the edges have similar weights. However, a graph is still too much abstract to drive algorithms on it. That is why we use its corresponding adjacency matrix, i.e. a matrix $A \in \{0,1\}^{n\times n}$ where n is the number of individuals. Of course, we will see how to represent adjacency matrices in multi-layer network context. However, the main data structure of a network is a set of adjacency matrices.

To answer to the second interrogation, we should think on what defines a community in the common sense. Let's think about the social case to help us to think about it. People in a community may probably have more interactions between them than with a person exterior to this community. In fact, belonging to a community could fix the probability of having an interaction with a person of one another (or even the same) community. That is exactly the assumptions of the frameworks we focus on this report.

Under this probabilistic aspect, the three approaches will try to approximate the best as they can the true community memberships of individuals and layers according to the framework using an estimator of the expectation. Except for the first one, all the approaches also use some specific linear algebra results, such as the *Singular Value Decomposition (SVD)* and projection on low-ranks matrices with respect to the *Frobenius* norm. Note that, now, we would like to extend these results on tensors. To talk about the algorithms on themselves, in term of implementation, they are not so far of the K-means algorithm. In the sense, that we also proceed iteratively up to certain stopping criterion, which could be the distance of two vectors from the last two iterations or on a threshold of the number of iterations.

Hence, in this work, we describe first in details the different frameworks. Then, we present and explain the different approaches by adding some theoretical justifications on why they converge to wanted memberships vectors. After that, we spend some time on comparing them under different contexts and determine which one performs the best under each framework. And, naturally, the goal is to apply them on a real context. However, we need to know at least the number of communities to perform these algorithms. Hence, we will discuss of these choices of parameters to perform the approaches on a real data set.

2 Notations

We specify here notations that will be used throughout this report. We denote by $[n]$ the set $\{1, ..., n\}, \forall n \in \mathbb{N}$. For any matrix $M \in \mathbb{R}^{n_1 \times n_2}$, we denote the entry in the i-th row and j-th column by $M[i, j]$. The i-th row of M is denoted by $M[i, :]$ and, similarly, the j-th column of M is denoted by $M[:, j]$.

For any tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we use the similar notations as matrices, i.e. for $i \in [n_1]$, $A[i, :, :]$ represents the i-th layer of A.

We denote a network by its adjacency tensor $A \in \{0, 1\}^{L \times n \times n}$, where L corresponds to the number of layers and n corresponds to the number of individuals. Then, for any $l \in [L]$, A_l corresponds to the l-th layer of A, i.e. $A_l = A[l, :, :]$.

In the MMSBM framework, we will introduce the notion of layer classes. We denote by M the number of such classes. We use $S_m \subset [L]$ to denote a layer class $\forall m \in [M]$.

Consider again a multi-layer network $A \in \{0, 1\}^{L \times n \times n}$ with $M \geq 1$. We use k_m to denote the number of communities in layer class m. We also use $G_{m,i} \subset [n]$ to denote the $i - th$ community in the $m - th$ layer class and denote by $K_{m,i}$ its respective size.

We also introduce two ways to describe the community membership in layer class $m \in [M]$. The first way is to use a vector $g_m \in [k_m]^n$, where $g_m[i]$ is the label of the corresponding community of $i \in [n]$. The other way is to use a matrix $Z \in \{0, 1\}^{n \times k_m}$ such that $Z[i, j]$ is equal to 1 if j corresponds to the label of the community of i, otherwise, it is equal to 0.

We give here operators on tensors and matrices that we will use in the different approaches (see the article of Kolda and Bader [2009] to have more details on those operators). Consider a matrix $X \in \mathbb{R}^{n_1 \times n_2}$. We denote by $\prod_0(X)$ the projection operator $X(X^T X)^{-\frac{1}{2}}$. Consider a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and a matrix $X \in \mathbb{R}^{m \times n_1}$. We denote by $A \times_1 X \in \mathbb{R}^{m \times n_2 \times n_3}$ their mode-1 product such that

$$[A \times_1 X](j, i_2, i_3) = \sum_{i_1=1}^{n_1} A[i_1, i_2, i_3] X[j, i_1].$$

Finally, consider again $A \in \mathbb{R}^{n \times n_2 \times n_3}$ and $B \in \mathbb{R}^{m \times n_2 \times n_3}$. We denote by $A \times_{2,3} B \in \mathbb{R}^{n \times m}$, their $(2, 3)$-mode product such that

$$[A \times_{2,3} B](i_1, i_2) = \sum_{j_2=1}^{n_2} \sum_{j_3=1}^{n_3} A[i_1, j_2, j_3] B[i_2, j_2, j_3].$$

3 Model of Multi-layer Network

3.1 What is a network?

First of all, let's explain intuitively what a multi-layer network is. To do so, we consider the most simple case: the single layer network. It consists of a network with one layer of n individuals $\{i_1, i_2, ..., i_n\}$ that we will replace by $[n]$ for simplicity. It displays pairwise interactions among the n individuals. One way to represent those interactions is to use a binary system, i.e. if s and t have an interaction, then the pairs (s, t) and (t, s) are equal to one, otherwise, we set them to zero. Naturally, we can represent all these interactions under one mathematical object: a matrix $A \in \{0, 1\}^{n \times n}$, where $A[s, t] = (s, t)$. Since the interactions are symmetric, A is symmetric. We set here the diagonal entries of A (i.e. $A[i, i], \forall i \in [n]$) to be equal to 0 since we do not consider interaction of an individual with himself in this framework.

Now that we know how to handle a single layer network, we can jump in the case of the multi-layer case. Basically, instead of having one adjacency matrix A, one has $L \geq 2$ adjacency matrices A_l, $l \in [L]$. The i-th layer corresponds to the adjacency matrix A_i. To represent all of theses matrices at once, we use a tensor form. A tensor A is a multi-dimensional array such that $A \in \{0, 1\}^{n_1 \times n_2 \times ... \times n_d}$. In our context, a third order tensor, i.e. $d = 3$, will be enough to represent a multi-layer network. In fact, a observed multi-layer network will be represented by $A \in \mathbb{R}^{L \times n \times n}$ where n corresponds to the number of individuals. So, $A[i, j, k] = A_i[j, k] \in \{0, 1\}$, $\forall i \in [L], j, k \in [n]$. The adjacency matrix (or tensor) is a kind of measurement of the interactions between each individuals. We remember that our goal is to retrieve the different communities in a multi-layer network. So, we have to introduce a new notion of measurement for the communities in a network.

Before knowing which individual belongs to which community, we should spend some time on the characterization of a community. By the literal translation of the latter, a community could be characterized by a measurement of interactions among its members. In fact, members of a community should have more interactions between them than if they were not members of the same community. Under a probabilistic view, this measurement is given by the following statement:

$$\mathbb{P}((s,t)=1) \geq \mathbb{P}((s,u)=1), \tag{1}$$

where s and t belong the same community G and u does not necessarily belongs to it. In this report, we will consider frameworks which fix the probability of having an interaction given the respective communities of two individuals. This leads to the introduction of the *Stochastic Block Matrix* (*SBM*) $B \in [0,1]^{k\times k}$. In fact, the entries of B correspond to the probability of having an interaction between individuals given their community.

$$\mathbb{P}((s,t)=1) = B[i,j],$$

where $s \in G_i, t \in G_j$. The frameworks also fix the probabilities among two different communities by principle of cross community density.

Example 3.1. *Let's get a look at a single layer network with three communities and* 9 *individuals. So,* $k=3$ *and* $n=9$*.*

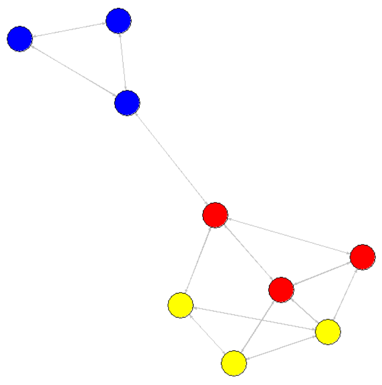


Figure 1: Such graphs are methods of visualisation of networks. In this case, each dot is colored with respect to its community membership $G_1 = \{1,2,3\}, G_2 = \{4,5,6\}$ and $G_3 = \{7,8,9\}$.

Its relative stochastic block matrix is given by:

$$B = \begin{pmatrix} 0.8 & 0.3 & 0.2 \\ 0.3 & 0.7 & 0.3 \\ 0.2 & 0.3 & 0.8. \end{pmatrix}$$

And, its adjacency matrix is:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0. \end{pmatrix}$$

We can then generalize the previous definitions for a multi-layer network. However, it is not as simple as that. There are many ways to do such generalization over several layers. As previously stated, we focus on two of them in this report.

3.2 Multi-layer Stochastic Block Model

The *Multi-layer Stochastic Block Model* (MSBM) framework is a multi-layer network model where each layer i follows a stochastic block matrix B_i and the community membership is constant across the layers. i.e. for all individuals $i \in [n]$, i belongs to the same community across the L layers. However, the edge density can vary across the layers since $\mathbb{P}(A_l[s,t] = 1) = B_[i,j]$, where $s \in G_i$ and $t \in G_j$. We get a better understanding of this model through the next example.

Example 3.2. *Consider a network of* $n = 10$ *individuals,* $k = 3$ *communities and* $L = 3$ *layers. We also give the respective stochastic block matrices:*

$$B_1 = \begin{pmatrix} 0.8 & 0.3 & 0.2 \\ 0.3 & 0.7 & 0.3 \\ 0.2 & 0.3 & 0.8 \end{pmatrix}, B_2 = \begin{pmatrix} 0.9 & 0.3 & 0.2 \\ 0.3 & 0.6 & 0.2 \\ 0.2 & 0.2 & 0.75 \end{pmatrix}.$$

From this information, we can generate a network, which we will visualize as an adjacency tensor $A \in \{0,1\}^{L\times n\times n}$. *We give here as an example the* 2^{nd} *layer under its matrix form.*

$$A_2 = A[2,:,:] = \begin{pmatrix} 0&1&1&1&1&0&0&0&0 \\ 1&0&1&1&0&1&0&0&0 \\ 1&1&0&0&0&1&1&0&0 \\ 1&1&0&0&1&1&1&1&0 \\ 1&0&0&1&0&0&1&0&1 \\ 0&1&1&1&0&0&0&1&1 \\ 0&0&1&1&1&0&0&1&0 \\ 0&0&0&1&0&1&1&0&0 \\ 0&0&0&0&1&1&0&0&0 \end{pmatrix},$$

where $G_1 = \{1,2,3\}, G_2 = \{4,5,6\}, G_3 = \{7,8,9\}$. *This layer follows then the stochastic block matrix* B_2.

3.3 Mixture of Multi-layer Stochastic Block Model

The *Mixture of Multi-layer Stochastic Block Model* (MMSBM) model is a mixture of the previous one. It is also a multi-layer network of $L \geq 2$ layers but now, the community membership can vary across the layers. This leads to a new definition: the *layer class*.
The layer class is a partition $\bigcup_{i\in[M]} S_i = [L]$ such that among layers in a class S_i is constant. So, $M \leq L$. And the layers of the same class S_m are generated by the same stochastic block matrix $B_{m,i} \in [0,1]^{K_{m,i}\times K_{m,i}}, m \in [M]$.

Remark 3.1. *The MSBM is a kind of a subclass of MMSBM by restraining* $M = 1$. *It is not exactly a subclass since the stochastic block matrices should not vary as well across the layers in a MMSBM.*

4 Presentation of the approaches

Here are the three methods to detect communities after observing adjacency tensor (or matrix) $Y \in \{0,1\}^{L\times n\times n}$. So, we want retrieve the community membership for each individual in the network in each layer class and layer class membership for each layer. We have to pay attention to the fact that no information is available other than the adjacency tensor, the number of class and the number of communities in each class.

4.1 Least square approach

In a MSBM case, one could think to an intuitive way to lead this detection by applying an average of the adjacency matrices. By doing this, we lose some information since the stochastic block matrices are different from each other. The following example illustrates this issue.

Example 4.1. *Let's consider these three stochastic block matrices:* $B_1, B_2, B_2 \in [0,1]^{3\times 3}$.

$$B_1 = \begin{pmatrix} 0.6 & 0.4 & 0.4 \\ 0.4 & 0.2 & 0.2 \\ 0.4 & 0.2 & 0.2 \end{pmatrix}, B_2 = \begin{pmatrix} 0.2 & 0.4 & 0.2 \\ 0.4 & 0.6 & 0.4 \\ 0.2 & 0.4 & 0.2 \end{pmatrix}, B_3 = \begin{pmatrix} 0.2 & 0.2 & 0.4 \\ 0.2 & 0.2 & 0.4 \\ 0.4 & 0.4 & 0.6 \end{pmatrix}.$$

We observe that the average $(B_1 + B_2 + B_3)/3 = \frac{1}{3} I_3$. So, all the entries of this average would be equal which theoretically implies that the detection is impossible, since the probability of having an interaction between different communities is the same as having one within a community.
However, communities exist. So, we have to propose a more consistent approach.

The approach we propose here is called the *Least squares approach* (more details in Lei et al. [2019]). This amounts to derive the following minimizers:

$$(\hat{g}, \hat{B}) = \underset{g \in [k]^n, B \in [0,1]^{k \times k}}{\operatorname{argmin}} \sum_{i=1}^{L} \sum_{1 \leq j \neq l \leq n} (Y_{i,j,l} - B_{i,g_j,g_l})^2. \tag{2}$$

4.1.1 Algorithm

This leads to the following iterative algorithm, such as k-means one, after observing an adjacency tensor $A \in \{0,1\}^{L \times n \times n}$.

```
Algorithm 1: Least square approach
  Require: $A \in \{0,1\}^{L \times n \times n}, k \geq 2$
  Ensure: $\hat{g} \in [k]^n, \hat{B} \in [0,1]^{k \times k}$
    $\tilde{A} \leftarrow$ reshape of $A$ of size $(n \times (L * n))$
    $g_{old} \leftarrow kmeans(\tilde{A})$
    for $i \in [L], k_1, k_2 \in [k]$ do
        $B_{old}[i, k_1, k_2] \leftarrow \frac{\sum_{j \neq l} A[i,j,l] \mathbb{1}_{g_{old}[j]=k_1} \mathbb{1}_{g_{old}[l]=k_2}}{\sum_{j \neq l} \mathbb{1}_{g_{old}[j]=k_1} \mathbb{1}_{g_{old}[l]=k_2}}$
    end for
    repeat
        $g_{new}[j] \leftarrow \underset{k_1 \in [k]}{\operatorname{argmin}} \sum_{i=1}^{l} \sum_{j \neq l} \{Y[i,j,l] - B_{old}[i, k_1, g_{old}[l]]\}^2, \forall j \in [n]$
        $B_{old}[i, k_1, k_2] \leftarrow \frac{\sum_{j \neq l} Y[i,j,l] \mathbb{1}_{g_{old}[j]=k_1} \mathbb{1}_{g_{old}[l]=k_2}}{\sum_{j \neq l} \mathbb{1}_{g_{old}[j]=k_1} \mathbb{1}_{g_{old}[l]=k_2}}, \forall i \in [L], k_1, k_2 \in [k]$
    until $f(g_{old}) \leq f(g_{new})$
```

4.1.2 Theoretical explanation

We give here some ideas which help us to have a better understanding of why this algorithm converges. The goal is to find the minimizers of problem (2). If the membership vector $g \in [k]^n$ is fixed, then the minimization problem becomes simply a least square problem by using the fact that for the true stochastic block matrices $B_i^{true} \in [0,1]^{k \times k}$:

$$B_i^{true}[h_j, h_k] = \mathbb{P}(A[i, k_1, k_2] = 1), \forall i \in [L], k_1, k_2 \in [k].$$

So, $\tilde{B}$ become an average of the corresponding entries in A given by the (fixed) membership h contrary to the average in 4.1.

If we consider now that the stochastic block matrix B is fixed, we want to find the correspondent $g \in [k]^n$ such that the total variance over the network is minimized (as in K-means algorithm for example). However, the algorithm proposes to proceed in a iterative way and directly choose g such as it minimizes the sum of squares.

4.2 TWIST

The TWIST approach is applied on MMSBM, i.e. we consider now a different model with aspect of class of layer. However, it can also be applied on MSBM framework too. The TWIST method is based on tensor decomposition which reveals the structure of the network. There is a particularity with this approach, as it does not only reveal the local membership, i.e. the membership for each class of layer, but it derives also directly the global membership, which is define in the following way. We say that two individuals $i, j \in [n]$ belong to the same global community if and only they belong to the same local community in each layer class $m \in [M]$.

Example 4.2. *Let's suppose that we have a network with $L = 3, M = 2$ (The number of individuals is not relevant here.) and each class of layer have three communities, i.e. $k_1 = k_2 = 3$. From these, we get the following illustrations:*

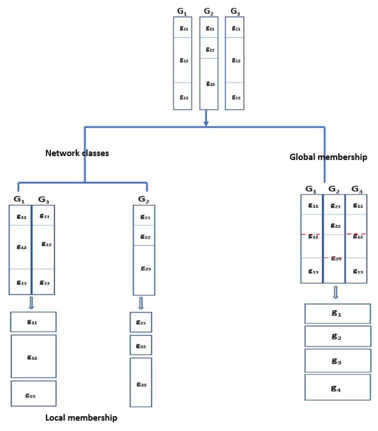


Figure 2: This figure shows the relation between the global and local communities.

4.2.1 Algorithm

This leads then to the method after observing an adjacency tensor $A \in \{0,1\}^{L \times n \times n}$.

- We approximate $\bar{U}$ by $\hat{U}$ and $\bar{W}$ by $\hat{W}$ of 9 by applying algorithm 2.
- Then, we apply K-means algorithm, with l_2 norm on the rows of $\hat{U}$ to derive the global community membership $\mathbb{V}$.
- The next step is to identify the layer class memberships by applying K-means algorithm to the rows of $\hat{W}$.
- Finally, we retrieve the local community memberships $g_{nodes} = (g_{nodes,1}, .., g_{nodes,m})$ where $g_{nodes,i} \in [k_i]^n$ for each $i \in [M]$ by K-means to the sub-tensor $A(\hat{l}_l = j, :, :)$ for each $j \in [M]$

Algorithm 2: Regularized power iterations for sparse tensor decomposition

```
Require: $A \in \{0,1\}^{n \times n \times L}$, warm initialization $\hat{U}^{(0)}, \hat{W}^{(0)}$, rank $(r, r, M)$, maximum iterations parameter
    $iter_{max}$, regularization parameters $\delta_1, \delta_2 \geq 0$ and tolerance parameter $\epsilon_{tol}$.
Ensure: $\hat{U} \in \mathbb{R}^{n \times r}$ and $\hat{W} \in \mathbb{R}^{L \times M}$
    $iter \leftarrow 0$
    while $iter < iter_{max}$ and $||W^i - W^{i+1}||_F \geq \epsilon_{tol}$ do
        Regularization: $\tilde{U}^{(iter)} \leftarrow \mathcal{P}_{\delta_1}(\hat{U}^{(iter)})$ and $\tilde{W}^{(iter)} \leftarrow \mathcal{P}_{\delta_2}(\hat{W}^{(iter)})$ by 3
        $iter \leftarrow iter + 1$
        Set $\hat{U}^{(iter)}$ to be the top $r$ left singular vectors of $\mathcal{M}_1(A \times_2 \tilde{U}^{(iter-1)T} \times_3 \tilde{W}^{(iter-1)}T)$.
        Set $\hat{W}^{(iter)}$ to be the top $m$ left singular vectors of $\mathcal{M}_3(A \times_1 \tilde{U}^{(iter-1)T} \times_2 \tilde{U}^{(iter-1)}T)$.
    end while
    $\hat{U} \leftarrow \hat{U}^{(iter)}$
    $\hat{W} \leftarrow \hat{W}^{(iter)}$
```

In this algorithm, we used the following regularization function.

$$\mathcal{P}_\delta(U) = SVD_r(U_\star), \tag{3}$$

where $U_\star(i,:) := U(i,:) \cdot \dfrac{min(\delta, ||U(i,:)||)}{||U(i,:)||}, i \in [n]$. Additionally, we use $\mathcal{M}_1(\cdot)$ and $\mathcal{M}_\ni(\cdot)$ notations as in article Kolda and Bader [2009]. We mentioned also a warm initialisation, i.e. a convenient first guess. To do so, we follow this algorithm,

Algorithm 3: Warm Initialization

```
**Require:** $A \in \{0,1\}^{L\times n\times n}$, $rank = (r, r, M)$
**Ensure:** $U^{(0)} \in \mathbb{R}^{n\times r}$ and $W^{(0)} \in \mathbb{R}^{L\times M}$
    $U^{(0)} \leftarrow$ top-r left singular vectors of $\sum_{i=1}^{L} A[i,:,:]$
    $\delta_1 \leftarrow max_j \|e_j^T U^{(0)}\|$
    $\tilde{U}^{(0)} \leftarrow \mathcal{P}_{\delta_1}(U^{(0)})$
    $W^{(0)} \leftarrow \mathcal{M}_3(A \times_1 \tilde{U}^{(0)T} \times_2 \tilde{U}^{(0)})$
```

4.2.2 Theoretical explanations

The method is based on the following ideas. For each adjacency matrix $A_l \in \{0,1\}^{n\times n}$ and $i_1, i_2 \in [n]$, $A_l[i_1, i_2] \sim_{iid} Bern(Z_l(i_1,:)B_{ml}Z_l(i_2,:)^T)$, where $Z_l \in \{0,1\}^{n\times k_l}$ is the community membership matrix for the l-th layer as described in section 2. This implies that:

$$A_l \sim_{iid} Bern(Z_l B_l Z_l^T). \tag{4}$$

Hence, it turns out that:

$$\mathbb{E}(A_l|l) = Z_l B_l Z_l^T. \tag{5}$$

It leads to the following lemma.

Lemma 4.1. *For a adjacency tensor A (from a MMSBM framework), we get the following decomposition:*

$$\mathbb{E}(\mathbf{A}|\mathbb{L}) = \mathbf{B} \times_1 \bar{Z} \times_2 \bar{Z} \times_3 W, \tag{6}$$

where:

- $\mathbb{L} = [L]$,
- $\bar{Z} = (Z_1, ..., Z_L) \in \{0,1\}^{n\times \mathring{K}}$ *corresponds to the global membership matrix, and each* Z_j *is the local membership matrix where* $\mathring{K} = \sum_{i=1}^{M} k_i$.
- *and,* $B \in \mathbb{R}^{\mathring{K}\times\mathring{K}\times L}$ *is a* 3*-way probability tensor such that:*

$$B(:,:,j) = diag(0_{k_1}, ..., 0_{k_{j-1}}, B_j, 0_{k_{j+1}}, ..., 0_{k_m}), 1 \le j \le M, \tag{7}$$

where 0_k *is zero k by k matrix.*

We suppose that $\bar{Z}$ has rank $r = (r, r, M)$. From this result, we can go one step further by considering the singular value decomposition over a tensor of $\bar{Z}$, i.e.:

$$\bar{Z} = \bar{U}\bar{D}\bar{R}^T, \tag{8}$$

where $\bar{U} \in \mathbb{R}^{n\times r}$,$\bar{R} \in \mathbb{R}^{\mathring{K}\times n}$ have orthonormal columns, and $\bar{D} \in \mathbb{R}^{r\times r}$ is a strictly positive definite diagonal matrix. This leads to the next result.

Lemma 4.2. *For* $i_1, i_2 \in [n]$ *such that they do not belong to the same global community, we have:*

$$\|\bar{U}(i_1,:) - \bar{U}(i_2,:)\|_{l_2} \ge d_1^{-1},$$

where $d_1 \ge 0$ *is the first diagonal value of* $\bar{D}$.

Hence, this ensures that we can retrieve the global communities if $\bar{D}$ is strictly positive definite. We should also remark that we can apply *Tucker decomposition* to equality (6), which gives us:

$$\mathbb{E}(\mathbf{A}|\mathbb{L}) = \bar{\mathbf{C}} \times_1 \bar{U} \times_2 \bar{U} \times_3 \bar{W}. \tag{9}$$

This final decomposition of $\mathbb{E}(\mathbf{A}|\mathbb{L})$ is sufficient for us since it can be shown that the singular vectors in the first dimension of $\mathbb{E}(\mathbf{A}|\mathbb{L})$ contains the information related to the global community and singular vectors in the third dimension contains the information related to the latent network label, which can be used to derive the local communities afterward.

4.3 ALMA

This is the last method presented which is applied under MMSBM framework. Unfortunately, we can not apply on MSBM framework. It is also based on a tensor decomposition of $A \in \{0,1\}^{L \times n \times n}$. Contrary to TWIST, ALMA derives first the local communities. Then, we can compute the global manually if wanted.

4.3.1 Algorithm

By equation (11), ALMA method proposes to derive the minimzers:

$$\hat{Q}, \hat{W} = \underset{Q,W}{\operatorname{argmin}} ||A - Q \times_1 W^T||_F, \tag{10}$$

where Q, W lay on the following manifold $\{Q \in \mathbb{R}^{M \times n \times n}, W \in \mathbb{R}^{L \times M} \ | \ W^T W = I_M, rank(Q[m,:,:]) \leq k_m \ \ \forall m \in [M]\}$.
To solve problem (10), we proceed similarly as in TWIST after observing a tensor $A \in \{0,1\}^{L \times n \times n}$.

- We approximate Q by $\hat{Q}$ and W by $\hat{W}$ of (10) by applying algorithm 4.
- We apply K-means algorithm on the rows $\hat{W}$ to retrieve the layer class memberships.
- Finally, we apply M times K-means algorithm on the rows of matrices Q_m, $\forall m \in [M]$ to retrieve the community memberships in each layer class.

Algorithm 4: Alternating Minimization Algorithm

```
Require: $A \in \{0,1\}^{L \times n \times n}$, number of classes $M$, number of communities in each classes $\{k_m\}_{m=1}^M$, a Warm
  initialization matrix $W^{(0)} \in \mathbb{R}^{L \times M}$ such that $(W^{(1)})^T W^{(1)} = \mathbf{I}$. A tolerance parameter $\epsilon_{tol} > 0$.
Ensure: A clustering matrix $\hat{W} \in \mathbb{R}^{L \times M}$ such that $\hat{W}^T \hat{W} = \mathbf{I}$ and a tensor $\hat{Q} \in \mathbb{R}^{M \times n \times n}$.
  $iter \leftarrow 0$
  while $iter < iter_{max}$ and $||W^{(iter)} - W^{(iter+1)}||_F \geq \epsilon_{tol}$ do
    $Q^{(iter+1)} \leftarrow \prod_{\bar{K}}(A \times_1 W^{(iter)})$.
    $W^{(iter+1)} = \prod_0(A \times_{2,3} Q^{(iter+1)})$
    $iter \leftarrow iter + 1$
  end while
  $\hat{W} \leftarrow W^{(iter-1)}$.
  $\hat{Q} \leftarrow Q^{(iter-1)}$.
```

In algorithm 4, we mentioned a warm initialization algorithm for $W^{(0)}$.

Algorithm 5: Warm Initialization for ALMA approach

```
Require: $A \in \{0,1\}^{L \times n \times n}$, number of classes $M$ and number of communities in each class $\bar{K}$.
Ensure: $W^{(1)}$
  $W^{(0)} \leftarrow$ top $M$ left singular vectors (SVD) of $\mathcal{M}_3(A)$.
  $Z \leftarrow$ membership matrix of K-means algorithm applied on the rows of $W^{(0)}$.
  $W^{(1)} \leftarrow \prod_0(Z)$.
```

4.3.2 Theoretical explanation

We consider again the observed adjacency tensor $A \in \{0,1\}^{L \times n \times n}$ and the true stochastic block matrices $\{B_i\}_{i=1}^M$ and recall that M is equal to the number of class of layers. For all $m \in [M], B_m \in [0,1]^{k_m \times k_m}$. From that, we can define $\tilde{Q}_{i\star} \in [0.1]^{n \times n}$ such that:

$$\tilde{Q}_{l\star}[s,t] = B_m[i,j], \forall l \in S_m,$$

where $s \in G_{m,i}$ and $t \in G_{m,j}$. Then, we have to consider the following set of matrices $\mathcal{F}_{L,M}$,

$$\mathcal{F}_{L,M} = \{Z \in \{0,1\}^{l \times m} | \ Z\mathbf{1} = \mathbf{1}, Z^T\mathbf{1} \neq \mathbf{0}\}.$$

We can interpret this $\mathcal{F}_{L,M}$ as the set of clustering matrices, i.e. $Z \in \mathcal{F}_{L,M}$ such that $Z[l,m] = 1$ if and only if $l \in S_m$, otherwise $Z[l,m] = 0$. Now, for each $Z \in \mathcal{F}_{L,M}$, we transform it into $W_\star = W_\star(Z) = Z(Z^T Z)^{-\frac{1}{2}} \in \mathbb{R}^{L \times M}$.

Hence, $W_\star$ satisfies $W_\star^T W_\star = I_M$ and $W_\star[l,m] = L_m^{-\frac{1}{2}}$ if $l \in S_m$, where L_m is the size of the m-th class of layers, otherwise, $W_\star[l,m] = 0$. Similarly, we define $\mathcal{F}_{n,K_m}$ for the community clustering (within a class of layer m). So, we can interpret $\Theta_m \in \mathcal{F}_{n,K_m}$ as a membership matrix such that $\Theta_m[i,k] = 1$ if and only if the individuals i belongs to community k in layer class m. One last step, before the main result, is to define $Q_{\star,m} = \sqrt{|S_m|}\tilde{Q}_{\star,m}$, $\forall m \in [M]$. This leads to the tensor $Q_\star \in \mathbb{R}^{m\times n\times n}$ where $Q_\star[m,:,:] = Q_{\star m}$. Hence, we can conclude with the following lemma.

Lemma 4.3. *For an adjacency tensor $A \in \{0,1\}^{L\times n\times n}$ under a MMSBM framework, we have:*

$$\mathbb{E}(A) = P_\star = Q_\star^{true} \times_1 W_\star^{trueT}, \tag{11}$$

where $Q_\star^{true}$ and $W_\star^{true}$ are respectively the true memberships.

Hence, from (11), we understand why the algorithm can act iteratively.

4.4 Discussion of the approaches

One general observation is the fact that all of these approaches are iterative in the sense they converge. Please note that we do not say that it necessarily converges to the true values. However, in practice the number of iterations is finite. It explains why we have a stopping criterion over the norm of the difference of the last two step, since if the norm is small, we could consider that the algorithm is close enough of the true value. Moreover, we have theoretical guarantees over TWIST and ALMA that they converge to the true memberships as n tends to ∞. Hopefully, this convergence is observable numerically in section 5.

Secondly, we remark that we can not apply ALMA to a MSBM, i.e. when $M = 1$. Unfortunately, it is not possible to apply eigenvalue decomposition or singular value decomposition on a vector. However, we have to apply algorithm 4 as the first step of the approach. Hence, we can not use this approach. In fact, we can not apply algorithm 2 either. But, to retrieve communities via TWIST, we do not need to apply algorithm 2. That is why we can apply TWIST on MSBM framework. Hence, we will compare ALMA and TWIST on MMSBM framework and compare the Least square approach and TWIST on MSBM framework.

5 Numerical comparisons

We show here the performances of our approaches under the two different frameworks over simulated networks. Beyond the choice of the model, the structure of the generated networks is particular. So, the conclusions we will draw are only applicable in this context. We could think to other context as discussed in section 7. You can find the source code at the following link: `https://github.com/Waelgaf/Semesterproject`.

5.1 Performance measurements

One question could arise, how to measure the accuracy of our algorithms if we know the structure? There are multiple ways to perform such measurements. We only pick one here and discuss of the other possibilities in section 7. In fact, we choose the more intuitive approach to measure these performances: the missclassification rate. Remember that under a MMSBM framework, we classify two objects, the layers and the individuals. So, the performance can be measured using two error measurements simultaneously given by:

$$R_{layers} = L^{-1} \min_{\sigma\in\mathcal{P}(M)} \sum_{m=1}^{M} |S_m \setminus \hat{S}_{\sigma(m)}| \tag{12}$$

and,

$$R_{ind} = M^{-1} \sum_{i=1}^{M} n^{-1} \min_{\sigma\in\mathcal{P}(K_m)} \sum_{k=1}^{K_m} |G_{m,k} \setminus \hat{G}_{m,\sigma(k)}|, \tag{13}$$

where S_m and $G_{m,k}$ correspond to the true community sets and $\hat{S}_m$ and $\hat{G}_{m,k}$ are the estimated one.

5.2 Generation of network

To proceed to these comparisons, we have to generate a network, i.e. an adjacency tensor $A \in \{0,1\}^{L\times n\times n}$, and its respective true memberships (layers and node) to be able to compute the relative errors. Here are the main steps given the number of individuals n, the number of layers L, the number of class of layers M and the number of communities k, which, for simplicity, remains constant through the layer classes. First, we have to create only one SBM $B \in [0,1]^{k\times k}$ in function of $\alpha, \rho \in]0,1[$ such that:

$$B = \begin{pmatrix} \rho & \alpha\rho & \alpha\rho \\ \alpha\rho & \rho & \alpha\rho \\ \alpha\rho & \alpha\rho & \rho \end{pmatrix},$$

where $k = 3$. Then, we can easily control the sparsity of our networks.

After that, we have to generate M true community memberships of k communities. To do so, for each layer class $m \in [M]$ and for each individual $i \in [n]$, we consider that the probability of i belonging to any community of the layer class is uniform. Hence, the rows of the true membership $Z_m \in \{0,1\}^{n\times k}$ follow an uniform density over the set $\{(1,0,..,0),(0,1,...,0),...,(0,...,0,1)\}$.

Now that we have the true community memberships for each layer class, we want to create the true layer class membership for each layer $l \in [L]$. We can use the multinomial distribution to generate vectors $(m_1,...,m_L)$, which be the true layer class membership, such that $m_l \in [M], \forall l \in [L]$, with uniform probability $p_m = \frac{1}{M}, \forall m \in [M]$.

The last step is to generate the adjacency tensor $A \in \{0,1\}^{L\times n\times n}$. In fact, we will generate L adjacency matrices according to their layer class, that we will merge to get a tensor. To generate an adjacency matrix $A_l\{0,1\}^{n\times n}$ of layer class $m \in [M]$, we recall that $A_l \sim_{iid} Bern(Z_m^T B_m Z_m)$ and is symmetric. So, it suffices to generate $\frac{n(n-1)}{2}$ entries with respect to their correspondent parameters ρ or $\alpha\rho$.

5.3 Simulations environment

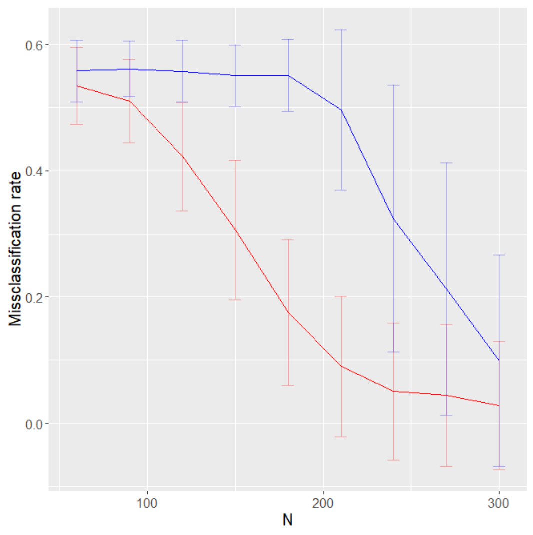
As the goal is to evaluate and compare the results of these approaches, we perform several simulations by varying one parameter at a time, which will be the number of individuals n, the number of communities in each layer class k and the sparsity of the network ρ. For each set up of parameters, we generate 100 networks and apply the two approaches whose performance is to be compared. Since we only vary one parameter at a time, we should fix default values.

Default parameters		
n (number of individuals)	L (number of layers)	k (number of communities)
100	40	3
ρ (sparsity)	M (number of layer class)	α (control of cross density)
0.6	3	0.5

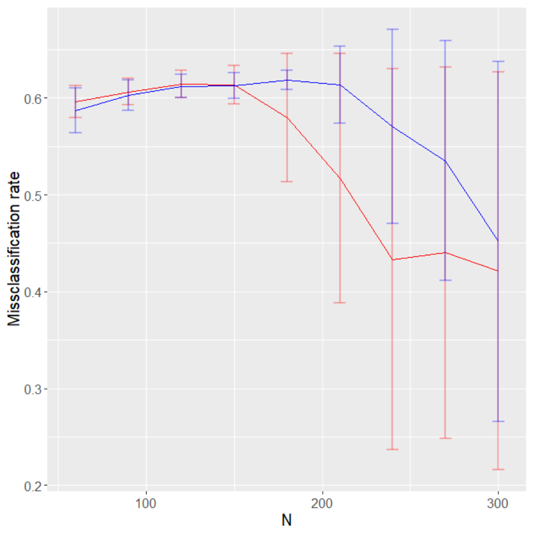
These parameters concern particularly the generation of networks. Recall that the approaches also have some parameters we can control as the tolerance factor ϵ_{tol} and the upper threshold for the number of iterations iter$_{max}$. For the most objective comparison possible, we set the parameter ϵ_{tol} to 10^{-3} and the number of iterations can go up to $iter_{max} = 500$ for all the approaches. We take a high threshold for the iterations in the hope that the algorithms will converge before reaching it.

5.4 ALMA / TWIST

Please note that you can retrieve the remaining figures in appendix C.



(a) Missclassification rate between layers



(b) Missclassification rate within layers

Figure 3: The blue curves correspond to ALMA and the red ones correspond to TWIST.

In 3, i.e. when n varies, we observe that the errors seem to converge to 0 as n gets larger. In fact, this is proved that the missclassification errors converge to 0 as n tends to ∞ for the both approaches. So, our results are consistent with the theory. However, with $\epsilon_{tol} = 10^{-4}$, we expected that ALMA to have the best performance as in the results of the article of Fan et al. [2021]. Hence, it seems that the tolerance parameter plays a key role since the conclusion over the results would be totally different with different tolerance.

In general TWIST performs better than ALMA. However, their performances have almost the same behavior, i.e. their curves seems similar in many points. It shows that these approaches are close in some sense. Indeed, tensor decomposition of both approaches are equal to $\mathbb{E}(A)$.

We may also observe that the two missclassification errors are strongly related. In one way, we can observe it. For example, in figure 3, the ALMA curve takes larger value for the missclassification rate between layers than TWIST and it coincides with the gap we find for the nodes part. In the other way, it makes sense theoretically. If we missclassify the layers, this leads to a missclassification for the nodes in layer classes since multiple layers will be combined even if they do not belong to the same layer class. And, we observe that missclassification rate standard deviation is the same concerning the layer. However, ALMA have the lowest standard deviation for the nodes community part. Note that it is increasing for the both approaches.

5.5 Least square approach / TWIST

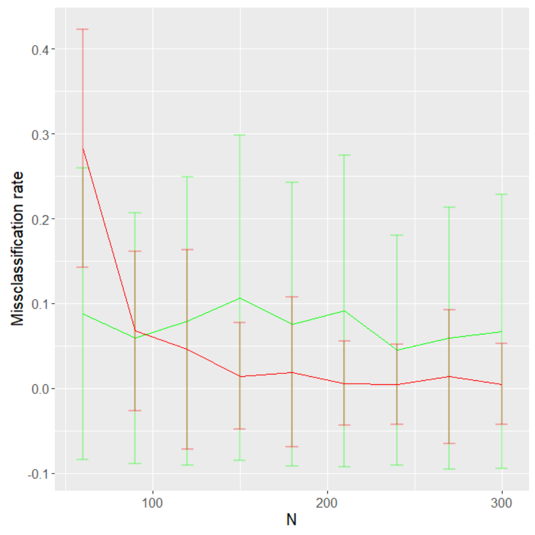


Figure 4: The green curve corresponds to the Least square approach and the red one correspond to the TWIST one.

In 4, i.e. when n varies, we remark that for n lower than 90, it is preferable to use the least square approach to retrieve the node communities. After this threshold, TWIST outperforms it. We should also pay attention to the fact that the missclassification error rate of Least square method does not necessarily converge to 0 as n grows to ∞, even if the envelop of its curve seems to decrease. We also remark that the order of the standard deviation stays constant in the Least square method while it decreases for the TWIST one.

6 Real Data Analysis

We apply now our methods to a real data set: airports and companies over Europe. The airports correspond to the nodes and the companies correspond to layers. Hence, we observe flights of many companies in a set of airports.

However, the adjacency tensor $A \in \{0, 1\}^{L\times n\times n}$ is the only available observation. Remember that all our methods require additional inputs such as the number of classes of layer (in MMSBM framework) M, the number of communities in each layer $(K_1, ..., K_M)$. Basically, before performing any method, we should clean our data set and choose M and $(k_1, ..., k_m)$.

6.1 Data set cleaning

What do we mean by cleaning the data set? In general, data cleaning corresponds to removing some incorrect data to perform efficient analysis. However, in our case, the data we remove are not incorrect, it corresponds to companies or airports which are unnecessary. They are unnecessary if they can not belong to a community or layer class. For example, we have some companies which only travel between two airports or airports that are only visited by very few companies. Hence, they can be removed since they do not bring a lot of information and will complicate the analysis. For us, the unnecessary layers, which correspond to companies, are the single-layer networks with an edge density lower an particular threshold U_l. And, the unnecessary nodes, which correspond to airports, are the one with a neighborhood size over all the cleaned layers lower than a particular threshold U_n.

6.2 How to choose the parameters?

We simplify the choice of parameters by assuming that each layer class have the same number of communities k. There are many methods to choose the best set of parameters. We present some of them below that we will combine to retrieve M and k. However, we should also use prior knowledge given by experts on the field.

6.2.1 Hierarchical clustering

The hierarchical clustering is the direct competitor of K-means algorithm. More details and explanations are given in the excellent book of James et al. [2021]. In fact, given a data set of size n and p features, it can separate into smaller groups following a tree-based representation of the observations. As there are also many approaches in the hierarchical clustering spectrum, we will focus here on the most common one, the *bottom-up* approach.

The algorithm is pretty simple. Given a distance, which allows us to measure the dissimilarity between two individuals, we consider first the n individuals as n clusters. The two clusters that are the most close are fused and form now a new cluster. What remains now is $n - 1$ clusters, and we proceed iteratively until it remains only one cluster. Then, we represent the results under a tree structure as in figure 5a.

However, there's one ongoing issue. Where should we cut the dendrogram to obtain a meaningful number of clusters. Several approaches to meet the challenge are discussed in the article of Rousseuw [1987]. Here, we will simply select the parameter which makes more sense visually.

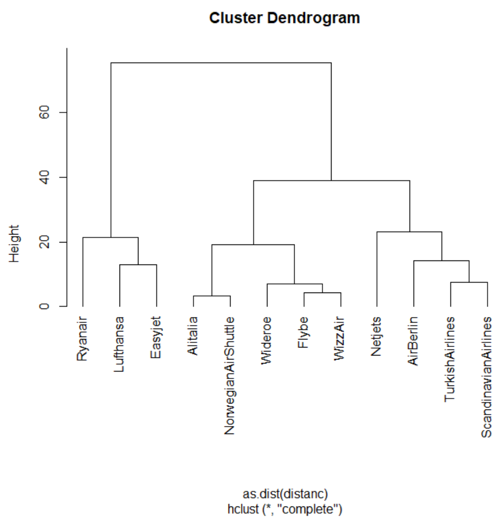
6.2.2 Elbow method

This method is based on the following observation. If we group our data (of size n) in k groups in a optimal way, such that the sum of within variance, that we will call $g(k)$ of each cluster is minimized, then $g(k)$ is a decreasing function (w.r.t. k). Hence, the optimal number of clusters $k^\star$ would be the one where we observe the biggest gap between its predecessor. So, in figure 5b, the optimal choice would be $k^\star = 4$. More details and explanation are given in the article of Tibshirani et al. [2001]. In fact, the main subject of this latter is the gap statistics method on which we discuss in section 7.

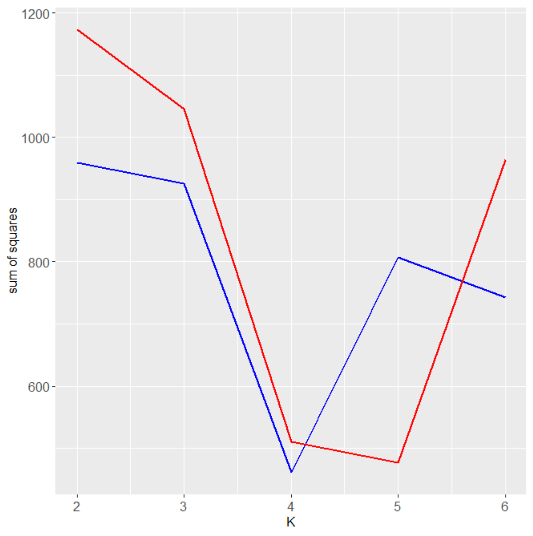
6.3 Results

Hence, we choose first the parameter M by applying hierarchical clustering to the layers. In fact, each layer was represented by its adjacency matrix $A_i = A[i, :, :] \in \{0, 1\}^{n \times n}$, where $A \in \{0, 1\}^{l \times n \times n}$ is the cleaned observed adjacency tensor. We need also to precise the distance to apply the hierarchical clustering. We use the normalized spectral distance since we are considering adjacency matrices proposed in the article of Donnat and Holmes [2018].

After choosing M, we choose k by applying the elbow method to sub tensors of A as in TWIST methodology. Then, we sum the within sum of squares in each layer class $m \in [M]$ to obtain this graph. The elbow is not exact here since the layer classes also depend on k. However, the both approaches seem to agree that the gap is maximized when $k = 4$.



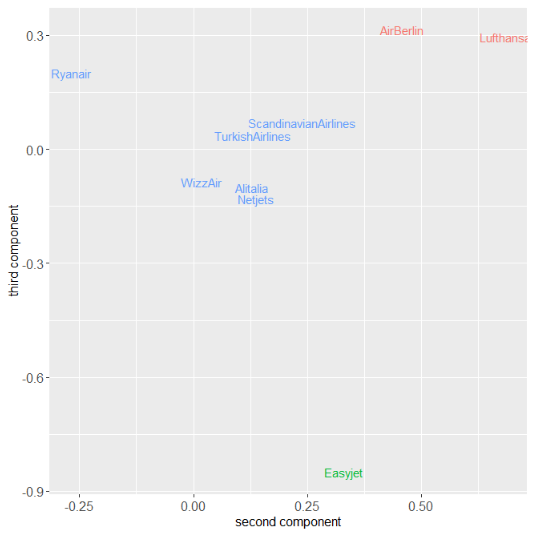
(a) Missclassification rate between layers



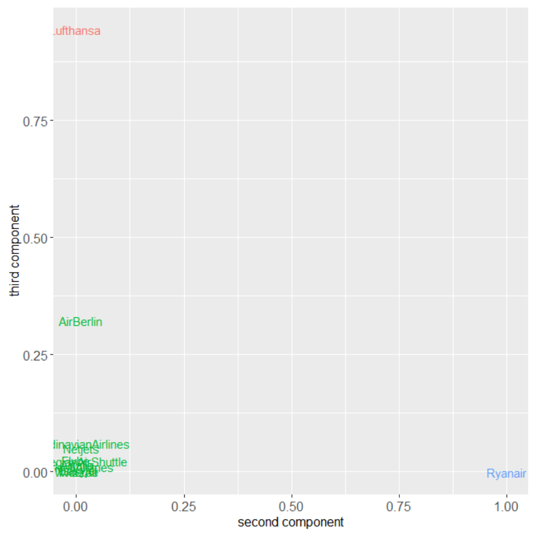
(b) Missclassification rate within layers

Figure 5: These graphs allow us to choose the parameters $M = 3$ by making a cut at height 30 and $k = 4$.

Hence, we can apply ALMA and TWIST to retrieve the cluster of the companies. In fact, we are not very interested to retrieve the communities of airports.



(a) TWIST

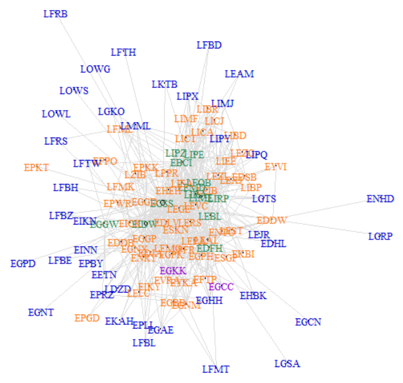


(b) ALMA

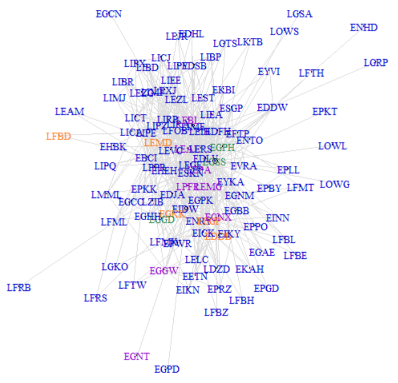
Figure 6: These figures are the results of TWIST and ALMA on which we apply the K-means algorithm to derive the layer class memberships.

Unfortunately, the layer class memberships are slightly different for the both approaches. They differ in three companies. They do not agree on *Ryanair* and *Easyjet* which should be exchanged. This finding is surprisig since we could think that both these companies should belong to the same layer class since they are the main low-cost companies. Also, TWIST consider that *AirBerlin* should be regroup with *Lufthansa*, which is not the case of ALMA which decide to join *AirBerlin* to the largest cluster. However, they both agree on the remaining companies. We remark that the clustering seems more obvious in ALMA than in TWIST. Please note that we choose to plot over the best coordinates of the matrices which allow us to visualize the best separation.

Let's then observe how the approaches classify in communities the network related to *Ryanair* since we can not look at them all. We choose *Ryanair* since it is a point of contention of the approaches. To get a clearer network, we remove the airports which are not visited by the company.



(a) TWIST



(b) ALMA

Figure 7: These are the networks with nodes colored with respect to their community membership.

As in the layer class case, we remark that the classification in communities are very different different from each other. Indeed, ALMA contains a large cluster while TWIST contains two large ones. We could also take a look at the whole network, without removing the non-visited airports, to make further analysis. However, it is hard to draw conclusion visually as you can see in appendix D.

All of these results lead to one main question: which approach should we believe? After cleaning the data set, we get $L = 12$ and $n = 242$. Hence, by numerical results of section 5, we should more trust TWIST approach. However, the set-up is not the same anymore. We can look at the *Ryanair*'s network heatmap in figure 8 and observe that its structure is different than generated stochastic block matrices. In fact, it suffices to look at the networks of 7, and remark that the edge density in each community is not the same. So, nothing assures us that we have to trust TWIST more than ALMA. We discuss about this issue in the next section.

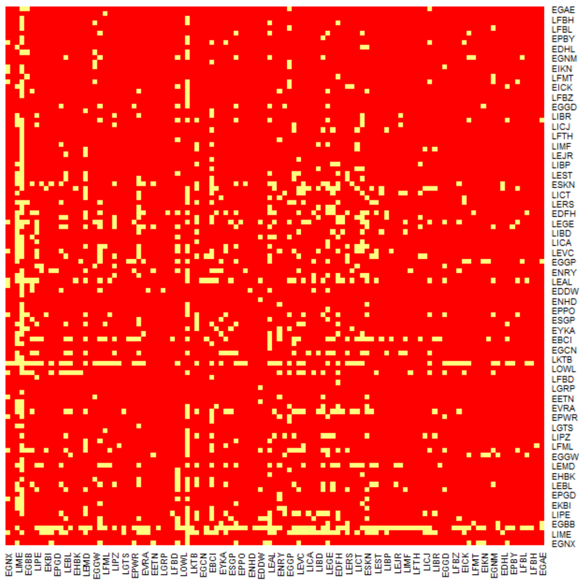


Figure 8: The adjacency matrix heatmap of *Ryanair* layer. If it is red, this means that there is no edge between these two airports of the corresponding row and column.

7 Discussion

A lot of choices have been made in this report such as the choice of the approaches we would focus on and/or the choice of frameworks. For example, we could consider, instead of MSBM, the *Multi-layer Weighted Stochastic Block Model* framework, where each layer is weighted. Such framework could be useful if we are considering the evolution of a network over the time for example. This framework is discussed in the excellent article of Ng and Murphy [2021].

Even on the performance measurements, we could think of different approach of measure such as a measure based on the stochastic block matrices. We discuss more about it in appendix A. We could think of a more complex numerical study by construction different type of network, e.g. letting each layer class having a different stochastic block matrices, having different number of communities. It leads to more robust conclusions that we could use in a real data analysis study.

Now, let's discuss on the real data set analysis. Our first step was to clean the data set. However, at the end of the analysis, we remark that the way we proceed to the cleaning can impact the results. That is the article subject of Li et al. [2021]. Then, we had to choose the number of clusters of layers and nodes. We could also mention the gap statistics method. We talk more about it in appendix B. Moreover, to run a complete study analysis, we should run numerical simulations to measure the performance of a such method. Finally, we discuss the main issue we meet at the end of the analysis. Which approach should we trust? The article of Demšar [2006] answer partially to this issue. However, we should also take into account prior knowledge over the given field on which we apply clustering.

References

Naomi Altman and Martin Krzywinski. Clustering. *Nature Methods*, 14:545–546, 2017. doi: https://doi.org/10.1038/nmeth.4299.

Ginestra Bianconi. *Multilayer Networks*. Oxford University Press, 2018.

Jing Lei, Kehui Chen, and Brian Lynch. Consistent community detection in multi-layer network data. *Biometrika*, 107(1):61–73, 2019. doi: https://doi.org/10.1093/biomet/asz068.

Bing-Yi Jing, Ting Li, Zhongyuan Lyu, and Dong Xia. Community detection on mixture multi-layer networks via regularized tensor decomposition. 2020. doi: https://doi.org/10.48550/arXiv.2002.04457.

Xing Fan, Marianna Pensky, Feng Yu, and Teng Zhang. Alma: Alternating minimization algorithm for clustering mixture multilayer network. 2021. doi: https://doi.org/10.48550/arXiv.2102.10226.

Tamara G. Kolda and Brett W. Bader. Tensor decomposition and applications. *Siam Review*, 51(3):455–500, 2009. doi: https://doi.org/10.1137/07070111X.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to Statistical Learning, second edition*. Springer Texts in Statistics, 2021.

Peter J. Rousseuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 63(2):411–423, 2001. doi: https://doi.org/10.1111/1467-9868.00293.

Claire Donnat and Susan Holmes. Tracking network dynamics: a survey using graph distances. *The Annals of Applied Statistics*, 12(2):971–1012, 2018. doi: https://doi.org/10.1214/18-AOAS1176.

Tin Lok James Ng and Thomas Brendan Murphy. Weighted stochastic block model. *Statistical Methods and Applications*, 30:1365–1398, 2021. doi: https://doi.org/10.1007/s10260-021-00590-6.

Peng Li, Xi Rao, Jennifer Blase, Yue Zhang, Xu Chu, and Ce Zhang. Cleanml: A study for evaluating the impact of data cleaning on ml classification tasks. *Georgia Institute of Technology*, 2021. doi: https://doi.org/10.48550/arXiv.1904.09483.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

A Performance measurement

One possible answer is the fact that one could compute SBM_{estim}, estimated SBM for each class and use the distance between SBM_{true} and SBM_{estim} as an error related to the accuracy. We have to pay attention the fact that we may retrieve the true memberships up to a permutation of the number of clusters. So, if we use what has been stated previously as an error, we would get:

$$error = \underset{\sigma \in \mathcal{P}(k)}{\text{argmin}} ||\text{SBM}_{true} - \text{SBM}^{\sigma}_{estim}||_{\mathcal{F}}$$

, where $\mathcal{P}(i)$ corresponds to the permutations of $[i]$, $\forall i \in \mathbb{N}$ and $\text{SBM}^{\sigma}_{estim}$ corresponds to the SBM's with respect to permutation indexes of the communities.
However, there's an issue with this method. In the case where the cross-community densities are high, i.e. close to inner-community densities, we may retrieve an estimated SBM close to the true one, even in the individuals are not well classified. Hence, this error does not measure the accuracy but can be useful as complementary error measurements.

B Gap statistic

This approach is closely related to the previous one. We suppose that we have n data points which lay in $\mathbb{R}^p$ and consider the following equality given k clusters $C_1, ..., C_k$:

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r, \tag{14}$$

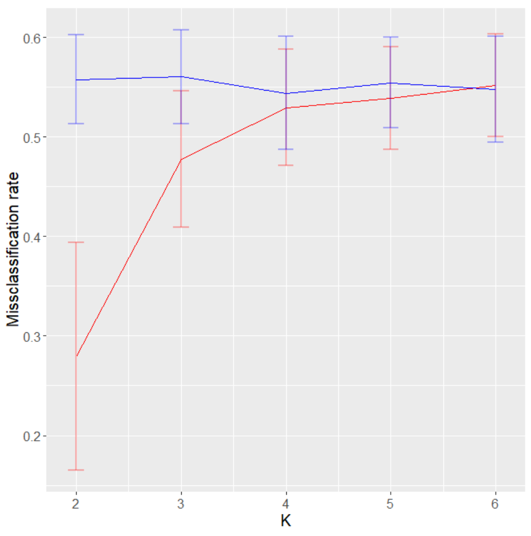
, where $D_r = \sum_{i,i' \in C_r} d(i, i')$ and $n_r = |C_r|$. The proposed approach is to standardize $log(W_k)$ by comparing it to its expectation under a reference null distribution, i.e. a distribution with no a priori clusters. Hence, it leads to the following function:

$$Gap_n(k) = E^{\star}_n\{log(W_k)\} - log(W_k). \tag{15}$$

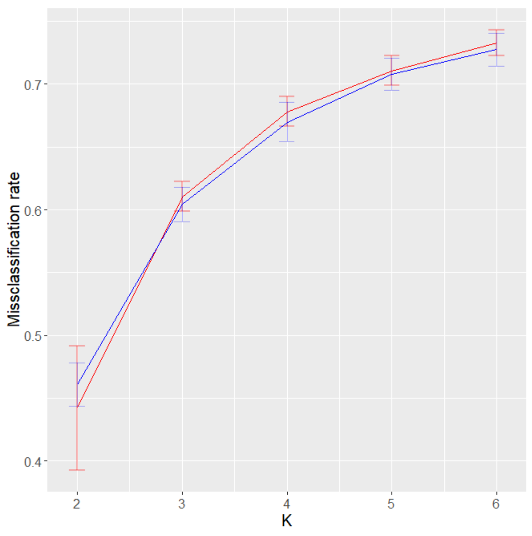
Then, the optimal choice k would be the one which maximizes $Gap_n(k)$.

One issue remaining with this approach is choosing the reference distribution. Hopefully, this issue is discussed in the article of Tibshirani et al. [2001].

C Numerical simulations: Figures

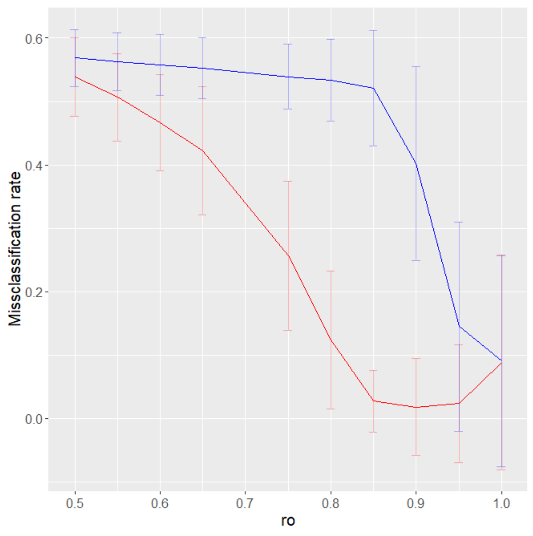


(a) Missclassification rate between layers

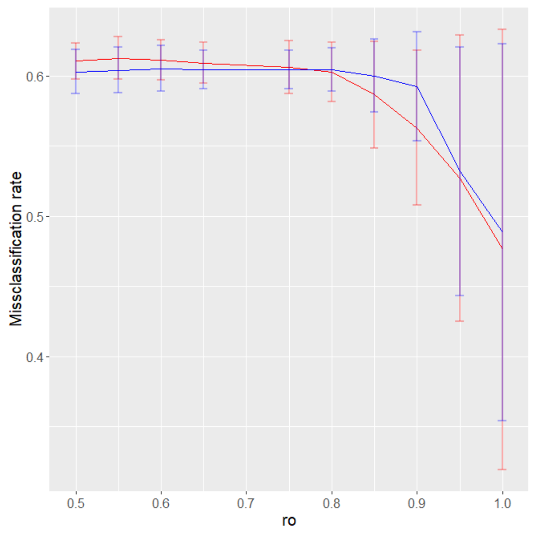


(b) Missclassification rate within layers

Figure 9: As in 3, ALMA is represented by the blue curve and TWIST is represented by the red curve. We observe that as K grows, the missclassification error for nodes and layers are. Again, TWIST is better than ALMA concerning the missclassification between layers. However, ALMA seems a little bit better to classify the individuals in this context.



(a) Missclassification rate between layers



(b) Missclassification rate within layers

Figure 10: As expected, if we increase the edge density, the approaches converge to the truth. We can explain it by recall that ρ is also the parameter of the *Bernoulli* density. Hence, as ρ tends to 1, the variance tends to 0. So, the adjacency matrix is equal to its expectation, which is what we approximate.

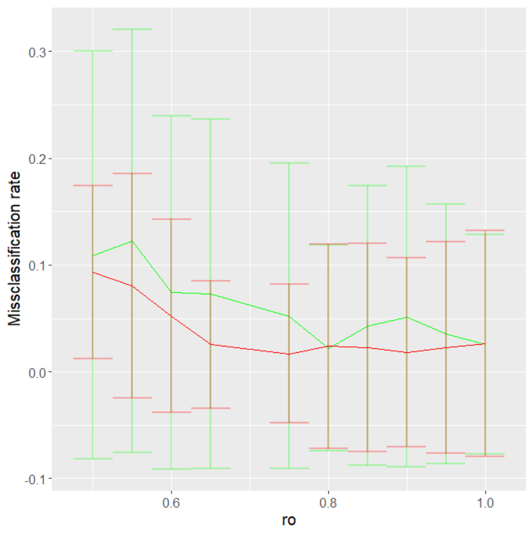


Figure 11: The green curve corresponds to the least square approach and the red one corresponds to TWIST. Clearly, TWIST is better than the least square approach when we vary ρ over the range $[0.5, 1]$. However, when $\rho = 1$, i.e. every individual interacts with everyone in its community, the both methods have the same mean and standard deviation.

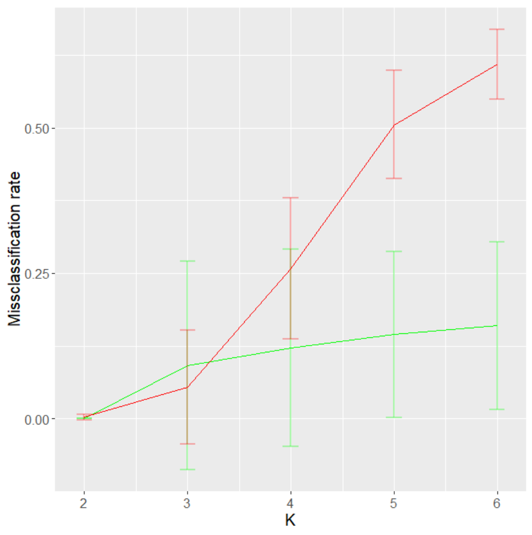
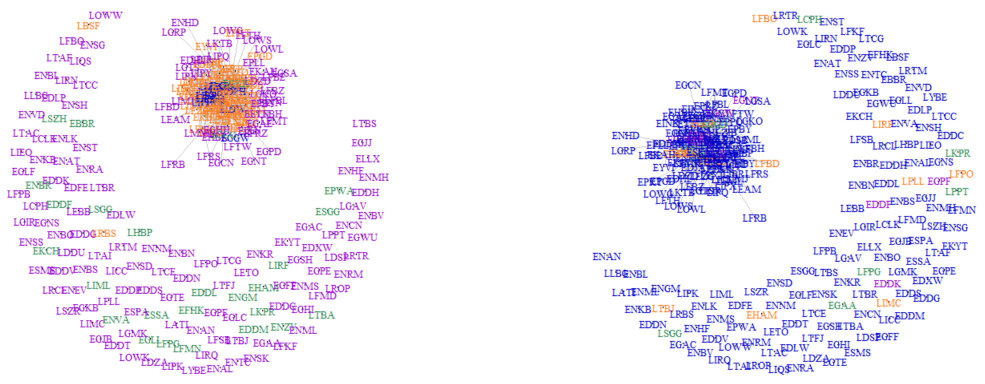


Figure 12: The green curve corresponds to the least square approach and the red one corresponds to TWIST. Least square approach outperforms clearly TWIST here.

D Networks



(a) TWIST

(b) ALMA

Figure 13: The full network of *Ryanair*, with the community memberships of the two different approaches.