Università commericiale Luigi Bocconi

Bachelor of Science in Mathematical and Computing Sciences for Artificial Intelligence

Stochastic block models for community detection in graphs

Relatore/
Supervisor:
Prof. Giacomo ZANELLA

Tesi di Laurea di/
Bachelor of Science thesis by:
Luca GANDOLFI
student ID no. 3164188

Academic Year 2023-2024

Abstract

This work aims to study and simulate in Python the Stochastic Block Model and some of its extensions. Firstly, an introduction to random graph theory and Network science is presented, with the purpose of having a preliminary overview of the field. Thereafter, the Stochastic Block Model is analyzed, both as a random graph model and as a method of doing community detection on graphs, and for this purpose, following a Bayesian approach, a Gibbs sampler is constructed to make inference on graphs. In the last two chapters two of the extensions of the Stochastic Block Model are studied, namely the MFM-Stochastic Block Model, preceded by an introduction to Mixture of Finite Mixtures and on Dirichlet Mixture Processes, and the Degree Corrected Stochastic Block Model. For the latter, differently from the first two, a Maximum Likelihood approach is followed, and a greedy algorithm is constructed.

All the models were simulated in Python and their implementation can be found at
https://github.com/LUKPROtm/Stochastic-Block-Model-for-community-detection-in-graphs

Contents

Chapter 1

Random graph theory and Network science

1.1 Introduction

The field of random graphs theory started with Pál Erdős (1913-1996) and Alfréd Rényi (1921-1970) in 1959-1961, with a series of influential papers: On random graphs I (1959) [1], On the evolution of random graphs (1960) [2], On the evolution of random graphs II (1961) [3] and On the strength of connectedness of a random graph (1961) [4]. At the beginning random graphs theory was used by Erdős and Rényi to prove deterministic properties of graphs using the Probabilistic method: for instance, if it is possible to show that a random graph has a certain property with positive probability, then it must exist a graph with that property. However, it was clear already that the study of random graphs could be used to model real-world networks, which is what we are interested in our work. In this chapter we mention some basic concepts of random graphs theory and of network sciences, in order to have an introduction to the topic and to be able to understand better the context of the main subject of this work: the Stochastic block model for community detection on graphs.

1.2 Random Graphs theory

1.2.1 Definitions

A random graph is a graph-valued random variable; it can be either described simply by a probability distribution over graphs or by a random process which generates the graph.
A random graph model is a family of probability distributions over graphs; examples of random graph models are the Erdős–Rényi random graph models, which are the main subjects

of this chapter and the various Stochastic Block models, which will be the main subjects of the next chapters.

The Erdős–Rényi random graph models are the building blocks of the study of networks and random graphs theory. There are two different Erdős–Rényi random graphs models: the uniform random graph, denoted $G(n, m)$, and the binomial random graph, denoted $G(n, p)$.

Definition 1.2.1 (Erdős–Rényi random graph model: uniform random graph)**.** *Let $\mathscr{G}_{n,m}$ be the family of all labelled undirected graphs with vertex set $V = \{1, 2, \ldots, n\}$ and exactly m edges, $0 \leq m \leq \binom{n}{2}$. Then, the uniform random graph $G(n, m)$ assigns the same probability to every element of $\mathscr{G}_{n,m}$. That is, $\forall g \in \mathscr{G}_{n,m}$*

$$\Pr(g) = \binom{\binom{n}{2}}{m}^{-1} \tag{1.1}$$

Equivalently, we start with an empty graph on V, and insert m edges in such a way that all possible $\binom{\binom{n}{2}}{m}$ choices are equally likely.

Definition 1.2.2 (Erdős–Rényi random graph model: binomial random graph)**.** *Let $\mathscr{G}_n$ be the family of all labelled undirected graphs with vertex set $V = \{1, 2, \ldots, n\}$. Fix $0 \leq p \leq 1$. Then the binomial random graph $G(n, p)$ is constructed such that, for all $\binom{n}{2}$ pairs of vertices $i, j \in V$, there is an edge between i and j with probability p, independently from all the other edges.*
Equivalently, we assign to each graph $g \in \mathscr{G}_n$ the probability

$$\Pr(g) = p^m (1-p)^{\binom{n}{2}-m} \tag{1.2}$$

where m is the number of edges of g.

Therefore, $G(n, m)$ fixes the number of edges, while $G(n, p)$ fixes the probability of each edge. Through the rest of the work, we will use mainly the $G(n, p)$ model and we will refer to it as the Erdős–Rényi random graph model.

1.2.2 Properties of the Erdős–Rényi random graph

We now briefly mention some properties of graph in generals and how they apply to Erdős–Rényi random graphs.

Degree distribution

Definition 1.2.3 (Degree)**.** *Given a graph $G = (V, E)$, the degree of the vertex $i \in V$, denoted k_i, is the number of edges connecting i to other vertices. That is, the degree k_i of a vertex i is defined as*

$$k_i = \#\{j \in V : \{i, j\} \in E\}. \tag{1.3}$$

In case of a directed graph, one has to consider two different degrees, the incoming degree k_i^{in} and the outgoing degree k_i^{out}; in this case the total degree k_i is the sum of the two.

Definition 1.2.4 (Degree distribution)**.** *Given a graph $G = (V, E)$, its degree distribution $p_0, p_1, ..., p_{max}$ is the distribution of the degrees of the vertices. In particular, if $|V| = n$, $p_k = \frac{n_k}{n}$, where n_k is the number of nodes with degree k.*

That is, if we pick a vertex at random, it will be of degree k with probability p_k. We denote the average degree $\langle k \rangle$, which is by definition the expectation of the degree distribution.
It is clear from the definition that the degree distribution of an Erdős–Rényi random graph $G(n, p)$ is a Binomial$(n-1, p)$, and therefore its (expected) average degree is $\langle k \rangle = p(n-1)$. In the large n and small p limit, that is when considering sparse graphs since $\langle k \rangle \ll n$ (which will be the case for real networks), the degree distribution of an Erdős–Rényi random graph is well-approximated by a Poisson distribution with parameter $\langle k \rangle$. This comes from the fact that we can view the Poisson(λ) distribution as the limiting distribution of a Binomial$(n, \frac{\lambda}{n})$ as n goes to ∞. Therefore we say that the Erdős–Rényi random graphs follow a Poisson distribution, that is, for $k = 0, ..., n-1$

$$p_k \approx \frac{\langle k \rangle^k e^{-\langle k \rangle}}{k!}. \tag{1.4}$$

Size of the largest component

A fundamental property of the $G(n,p)$ model is that it undergoes a phase transition for the size of the largest connected component. Indeed, fix an expected degree $\langle k \rangle$ and consider the sequence of random graphs $(G(n,p_n))_{n\in\mathbb{N}}$ with $p_n = \frac{\langle k \rangle}{n-1}$ so that the nodes of any random graph in the sequence have an expected degree of $\langle k \rangle$. For a given n, let L_n be the size of the largest connected component of $G(n,p_n)$. Then, as n goes to ∞, depending on $\langle k \rangle$, there could be different regimes (properties are with probability 1 as n goes to ∞):

1. **Subcritical regime**: $\langle k \rangle < 1$. In this regime $L_n = o(n)$, which means that all the connected components have a negligible size with respect to n.

2. **Critical regime**: $\langle k \rangle = 1$. Still $L_n = o(n)$, even though in absolute terms there is a jump in the size of L_n

3. **Supercritical regime**: $\langle k \rangle > 1$. The largest component contains a finite fraction of vertices, that is, $L_n = cn$ with $0 < c \leq 1$. We call such a component a giant component.

4. **Connected regime**: $\langle k \rangle > \log(n)$. The giant component absorbs all the vertices, $L_n = n$, the graph is connected.

Small world property

As we shall see later, often real networks show the small world phenomenon. Loosely speaking we say that a graph is a small world if the average distance between any two pair of vertices is much smaller than the total number of nodes n. This phenomenon is also called "six-degree separation" from the famous Milgram's experiment in 1967 [5], where he tried to estimate the average distance between people (where edges are social relationships), obtaining that in the sample studied on average people even living in very different places of the United States had a distance of 6 relationships, which is extremely small with respect to what one may think. More formally, we define

Definition 1.2.5 (Walk)**.** *Given $G = (V, E)$, a walk on G is a sequence of alternating vertices and edges that starts with a vertex and ends with a vertex such that consecutive vertices and edges in the sequence are incident to each other*

Definition 1.2.6 (Path)**.** *A path is a walk in which no edge is traversed more than once.*

We define the length of a walk as the number of edges in the walk. It is clear that, if G is connected, then there exists a finite-length path connecting any pair of vertices $i, j \in V$.

Definition 1.2.7 (Geodesic distance)**.** *Given $G = (V, E)$, for any pair of vertices $i, j \in V$, the geodesic distance of i and j is the length of the shortest path from i to j, that is, the minimum number of edges that one has to travel to go from i to j*

We denote by ℓ the mean geodesic distance in G. Since a priori G could be disconnected, thus having some nodes with ∞ distance, in the mean we condition on the fact that the geodesic distance is finite.

Definition 1.2.8 (Small World)**.** *Let $(G_n)_{n\geq 1}$ be a sequence of random graphs, with $G_n = (V_n, E_n)$ and $|V_n| = n$. We say that $(G_n)_{n\geq 1}$ is a small world if there exists a constant $K < \infty$ such that*

$$\lim_{n\to\infty} P\left(\ell_n \leq K \log n\right) = 1. \tag{1.5}$$

Furthermore, we say that $(G_n)_{n\geq 1}$ is an ultra-small world if $\forall \varepsilon > 0$,

$$\lim_{n\to\infty} P\left(\ell_n \leq \varepsilon \log n\right) = 1 \tag{1.6}$$

Theorem 1.2.1 (Chung and Lu., 2002)**.** *If $np_n \geq c > 1$ for some constant c, then for the $G(n, p_n)$ model ℓ is almost surely $(1 + o(1))\left(\frac{\log n}{\log np_n}\right)$ provided $\frac{\log n}{\log np_n} \to \infty$ as $n \to \infty$.*

Which means that the sequence of ER random graphs $(G(n, p_n))_{n\in\mathbb{N}}$ defined before satisfies the small world property if $\langle k\rangle > 1$, obtaining $\ell \sim \frac{\log n}{\log\langle k\rangle}$.[1]
The fact that, in the supercritical regime an ER random graph model is a small world as $n \to \infty$, that is the average geodesic distance is exponentially smaller than the number of vertices is something non-trivial at all. Our notion of distance is based on our experience on

[1] In particular, Fronczak et Al. 2004 [6] found the analytical solution, which is

$$\ell = \frac{\log n - \gamma}{\log\langle k\rangle} + \frac{1}{2} \tag{1.7}$$

where $\gamma \approx 0.57722$ is the Euler constant.

regular lattices, generally in 1D, 2D or 3D. For instance, in a regular d-dimensional lattice, l scales as $n^{\frac{1}{d}}$. This means that in a regular d-dimensional lattice the average geodesic distance between two vertices is much larger than the one in an ER random graph.

Clustering coefficient

Another important statistic of a graph is the clustering coefficient, which measures how much the neighbors of a given vertex link with each other.

Definition 1.2.9 (Local clustering coefficient)**.** *Given a graph $G = (V, E)$ the local clustering coefficient of a vertex $i \in V$ is defined as*

$$C_i = \frac{2M_i}{k_i(k_i - 1)} \tag{1.8}$$

where k_i is the degree of vertex i and M_i is the number of edges between the k_i neighbors of i.

Note that C_i is in $[0, 1]$, since $\frac{k_i(k_i-1)}{2}$ is the number of combinations without repetition of pairs of neighbours of vertex i. For a given vertex i, C_i is the probability that taking two of its neighbors at random, there is an edge between them. The degree of clustering of a graph is captured by the average clustering coefficient, denoted $\langle C \rangle$. The higher the average clustering coefficient, the more neighbors connect with each other.
From the definition of the $G(n, p)$ model, in particular from the independence assumption of each edge, the expected local clustering coefficient of each vertex of an ER random graph is $C_i = p = \frac{\langle k \rangle}{n-1}$. Therefore for a fixed $\langle k \rangle$, $C_i \to 0$ as $n \to \infty$. Thus, in a sparse ER random graph $\langle C \rangle \approx 0$.

1.3 Network science

Throughout this section, we will compare the ER random graph models with real networks and we will consider different and more accurate models to describe the latter. A parenthesis on the notation used: we will often use the words network, node and link to describe real networks even though they have the same meaning of graph, vertex and edge.
Some examples of real networks studied in the literature are described in 1.1:

Network	**Nodes**	**Links**	**Type**	n	m	$\langle k \rangle$
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.34
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Mobile-Phone Calls	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email addresses	Emails	Directed	57,194	103,731	1.81
Science Collaboration	Scientists	Co-authorships	Undirected	23,133	93,437	8.08
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908	83.71
Citation Network	Papers	Citations	Directed	449,673	4,689,479	10.43
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802	5.58
Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930	2.90

Table 1.1: Examples of real networks. Taken from [7]

Most real networks satisfy the following properties: they are small worlds, in the sense that $\ell \leq O(\log n)$, they have high clustering[2] and they are scale-free[3]. Moreover, in the majority of cases $1 < \langle k \rangle < \log n$, meaning that, if they were described by an ER random graph, they would be in the supercritical regime (but not in the connected one).

Clustering

As we have seen so far, in the supercritical regime the $G(n, p)$ model satisfies the small world property, however it does not have high clustering. Intuitively, for a network to have nodes with high local clustering coefficient means that, if a node i has an edge with both nodes j and k, then j and k are likely to be connected. It is quite natural to think that in real networks this is the case: for instance, if I am a friend of both i and j, then it is more likely that i and

[2]Differently from what discussed so far, in network science the phrase "small world" is often used not only to indicate the property of having a low average distance between vertices, but it also requires the high clustering property. However in this notation we will keep the two properties separated, calling only the first one small world.

[3]Actually, the universality of scale-free networks is controversial, see Broido and Clauset, 2019 [8], however, the discussion is still valid.

j are friends. In contrast, the ER model in its complete randomness does not take it into account. A notable extension of the Erdős–Rényi model which allows for high clustering is the Watts-Strogarz Model (Watts and Strogarz, 1998 [9]). This model interpolates between a ring lattice, which has high clustering but lacks the small world property, and an ER random graph, which lacks clustering but is a small world. It goes as follows (see Figure 1.1 for an illustration): start with a ring of nodes and connect all the nodes with their nearest neighbors and the second nearest ones. Then, rewire each edge with probability p (where p is a parameter of the model) and connect that to a randomly chosen node. Note that with $p = 1$ we go back to the ER random graph.

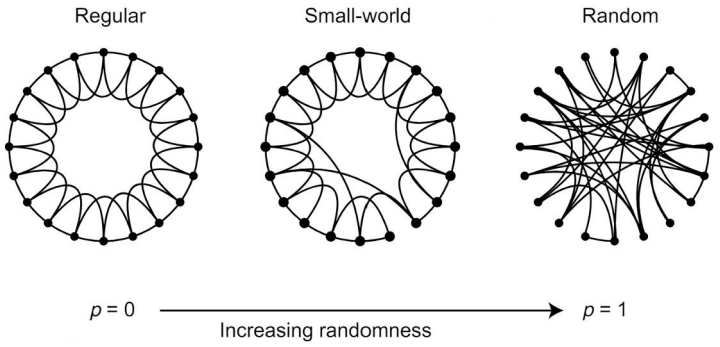


Figure 1.1: Illustration of the Watts-Strogarz Model. Image taken from [9].

Scale-Free property

The scale-free property is another important difference between the ER model and real networks

Most real networks are sparse, i.e. $\langle k \rangle \ll n$, therefore as before we will consider the Poisson distribution to be the degree distribution of the binomial random graph. However, most real networks do not follow a Poisson distribution. The first evident difference between the degree distribution of most of real networks and the Poisson distribution is that, in the latter, outliers are missing. Indeed, in a Poisson distribution, the tail decreases faster than exponentially. Using Stirling approximation

$$p(k) = \frac{e^{-\langle k \rangle} \langle k \rangle^k}{k!} \approx \frac{e^{-\langle k \rangle}}{\sqrt{2\pi k}} \left(\frac{e \langle k \rangle}{k} \right)^k \tag{1.9}$$

obtaining that for $k > e\ \langle k \rangle$, the term inside the parenthesis is smaller than one, and, as k increases, p(k) decreases more than exponentially. If we take the WWW network as an example (as studied by Albert et al., 1999 [10]), where each document (webpage) is a node and there is a link between two documents if there is a "link" from one to the other, it is clear that the variation in degree of the nodes is enormous: most of the webpages have only a few links, while there are pages, such as Google or Facebook, that have a degree with a completely different order of magnitude. If the WWW was described by a ER random graph, the existence of these so-called hubs (nodes with very high degrees), would not have been possible. The presence of hubs is verified for a large part of real networks.

The degree distribution of the WWW network is better approximated with a power law distribution, that is, a distribution of the form $p_k \sim k^{-\gamma}$. We call scale-free the networks which follow a power law with $2 < \gamma < 3$: the reason behind this name is related to the fact that, for $\gamma < 3$, the variance of the degree distribution is not finite and therefore it is impossible to define a scale within which the degrees of the nodes are, in contrast to Poisson networks, where the standard deviation is $\langle k \rangle^{1/2}$ and therefore nodes in the network have degree in the "range" $\langle k \rangle \pm \langle k \rangle^{1/2}$. Notice also that we are not interested in the case $\gamma < 2$, since in this regime also $\langle k \rangle$ is not finite.

In order to check whether a network is scale-free, it is useful to give a look at the so-called log-log plot, where $log(k)$ and $log(p_k)$ are on the x and y axis respectively. Taking the logarithm from the definition of the power law, we have $log(p_k) = \gamma log(k) + constant$, that is, a scale-free network should be well-approximated by a line on the log-log plot, while the parameter γ is the slope of that line. From this plot we can check that, for instance, the WWW network is indeed a scale-free network.

Remark 1.3.1 (Ultra-small world property). *Networks which follow a power law in the Scale-free regime ($2 < \gamma < 3$) not only are small words, but they actually are ultra-small worlds (Cohen et al. 2003 [11]), indeed in this case $\ell \sim \log\log n$. The presence of hubs is crucial in reducing the average geodesic distance between nodes because they connect a large number of small-degree nodes. This is a feature seen also in our life, as in the case of train stations (or airports): in general we have many small train stations and a few very connected ones, the hubs, and this structure dramatically reduces the average time needed to reach a destination.*

A network model that aims to explain the origin of the power law in real networks is the Barabási-Albert model (Barabasi, Albert 1999 [12]). The two main mechanisms underlying the model are: *Growth* and *Preferential attachment*. The first indicates that to obtain the final network we start with a smaller one (even empty) and at every time step we add a new node to it. The second implies that the newly added node will link more likely to nodes with high degree: in particular, each new node will link to m existing nodes, and each link connects to an existing node i with a probability proportional to k_i.
The final degree distribution obtained by the BA model is a power law with $\gamma = 3$.

Remark 1.3.2 (Degree Distribution of Real Networks). *Real networks very rarely follow a pure power law, indeed there are two main recurring deviations: Low-degree saturation which means that for $k < k_{sat}$ p_k is flat making the network have fewer small degree nodes than expected from a pure power law, and High-degree cutoff, which is a drop in p_k as soon $k > k_{cut}$, leading to fewer high degree nodes with degree greater than k_{cut}, and this can be due to inherent limitations in the number of links a node can have.*

Chapter 2
Stochastic Block Model

2.1 Introduction

We now turn our attention to the community detection task on graphs, that is, we want to find a way of grouping the vertices of a graph according to some criterion. There are two main classes of community detection methods for general data: model-based methods and non-model-based methods. In model-based methods one assumes there is a true model underlying the distribution of the data, while in non-model-based methods it is not assumed any underlying model but instead one groups elements according to some similarity measure. Among the model-based methods on graphs, the stochastic block model is the most used, while for non-model-based methods spectral clustering is the most relevant on graphs, see for instance von Luxburg, 2007 [13], even though another used option is hierarchical clustering, see for instance Girvan, 2002 [14] or Newman, 2004 [15]. More recent studies tried also to build clustering algorithms on graphs based on Graph Neural Networks, see Tsitsulin et al., 2023 [16]. Since non-model-based methods try to group data points based on some similarity measure, they would, in the graph scenario, group together nodes that are highly connected to each other, while, for model-based methods, this is not always the case, indeed the stochastic block model is able to recover structures with low intra-connectivity but sharing similar patterns and therefore, for this reason, it can be used with the broader scope of discovering the latent structure of the network.

2.1.1 Lineage of the Stochastic Block Model

The Stochastic Block Model was formalized by Holland et al. (1983 [17]) as a random graph model, while Wang and Wong (1987 [18]) were the first to apply it to real data but

assuming to know a priori the block structure, that is, the group membership. Snijders and Nowicki (1997 [19]) and (2001 [20]) studied a posteriori blockmodeling, which is what we are interested in: inference of the block structure, and they did it both in the case of the number of communities $k = 2$ and $k > 2$. In its simplest case of binary graphs (graphs with binary adjacency matrix), the model is referred to as *Bernoulli SBM*. Binary graphs have been studied by numerous models, among all, a relevant extension of the SBM on binary graphs is the *mixed membership models*, studied by Airoldi et al. (2008 [21]); Fu et al. (2009 [22]); Xing et al. (2010 [23]); Fan et al. (2013 [24]) and Li et al. (2015 [25]), which allows each node to belong to multiple communities; while in practice, the most used extensions are the *Degree-corrected SBM* of Karrer and Newmann (2011 [26]) and the *Microcanonical SBM* in its nested version (Peixoto, 2017 [27]). Some extensions focused on modelling the number of communities k, as the *MFM-SBM* of Geng et al. (2019 [28]), and in general, many more extensions have been studied.

In our analysis, we will discuss in Chapter 2 the original model proposed by Snijders and Nowicki, in particular in section 2.2 we introduce the basic model for undirected binary graphs and in section 2.3 we extend it to more general graphs and relations. In Chapter 3 we focus on the MFM-SBM, which allows us to include in the model the fact that we don't know a priori the number of communities, while in Chapter 4 we study the Degree-Corrected stochastic block model.

2.2 The Bernoulli Stochastic Block Model for undirected graphs

In this section we introduce the simplest Stochastic block model, the one called Bernoulli SBM for undirected graphs. In the next section we extend in its full generality and for this reason, we omit the discussion of how to make inference on the first as it will be easy to apply the general derivation to this specific case.

We have seen that in an Erdős–Rényi random graph, any pair of nodes has a probability of being connected which is denoted p and it is the same for every possible pair. Intuitively speaking, in the stochastic block model we assign a class at each node of the graph and the probability of having an edge between node i and j is not the same for every pair (i, j) as in the ER model, but depends on the classes of i and j. More formally, following the approach

of Snijders and Nowicki, 1997 [19] and 2001 [20], we consider n nodes and we define the set of possible classes $\mathcal{C} = \{1, ..., k\}$ where k is the number of classes. We define the attribute vector $X = (X_1, ..., X_n)$, where for every $i = 1, ..., n$, $X_i = x_i \in \mathcal{C}$ is the class of node i. We call Y the adjacency matrix of the graph and there is an edge between i and $j, i \neq j$ with probability $\eta(x_i, x_j) = \eta_{x_i,x_j}$, where η is the matrix of class-dependent edge probabilities. Note that we do not allow self-loops, therefore we impose $Y_{ii} = 0, \forall i = 1, ..., n$. Since we are considering undirected graphs, $\eta(c, d) = \eta(d, c), \forall c, d \in \mathcal{C}$. We thus have:

Definition 2.2.1 (Undirected Bernoulli Stochastic Block Model)**.** *An undirected Bernoulli stochastic block model is a family of probability distributions for an undirected colored graph G with vertex set $V = \{1, ..., n\}$ and color set $\mathcal{C} = \{1, ..., k\}$, defined as follow:*

- *The parameters are the vector $\pi = (\pi_1, ..., \pi_k)$ of class probabilities and the symmetric matrix η of class-dependent edge probabilities*
- *The attribute vector consists of i.i.d. random variables $X_1, ..., X_n$ where $\Pr(X_i = c) = \pi_c, \forall c \in \mathcal{C}$*
- *Conditional on X, for $i < j$ the edges Y_{ij} are independent Bernoulli$(\eta(X_i, X_j))$, and $Y_{ji} = Y_{ij}$*

2.3 The general Stochastic Block Model

We now extend the Bernoulli stochastic block model described before using:

- A symmetric set of observed pairs $\mathcal{N} \subseteq \mathcal{N}_0 = \{(i, j) \in \{1, ..., n\}^2 \mid i \neq j\}$ which allows for missing data.
- The dyad (Y_{ij}, Y_{ji}) will be our unit of analysis. We let α be the set of possible values of any Y_{ij}, while the set of possible values for (Y_{ij}, Y_{ji}) is denoted $\mathcal{A} \subseteq \alpha^2$ and it is called Alphabet of pairwise relations.

Note that with the Bernoulli SBM, we had $\mathcal{N} = \mathcal{N}_0$, $\alpha = \{0, 1\}$ and $\mathcal{A} = \{(0, 0), (1, 1)\}$ since we imposed $Y_{ij} = Y_{ji}$. For instance a directed graph would be described by $\alpha = \{0, 1\}$ and $\mathcal{A} = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ or a tournament by $\alpha = \{0, 1\}$ and $\mathcal{A} = \{(0, 1), (1, 0)\}$. For a signed directed graph we would have $\alpha = \{0, +, -\}$ and $\mathcal{A} = \{0, +, -\}^2$. Notice that we

always maintain the assumption that self-loops are not possible.
For every pair $(i,j) \in \mathcal{N}$ there is a pairwise relation $\mathbf{a} = (a_t, a_v) \in \mathcal{A}$ such that the relation from i to j is a_t and the relation from j to i is a_v. Since we are considering dyads and not single relations, we can explicitly model the mutual dependence of a_t with a_v.
Indeed, we extend the notion of η as following: we denote $\mathbf{Y}_{ij} = (Y_{ij}, Y_{ji})$, and we let:

$$\Pr(\mathbf{Y}_{ij} = \mathbf{a} \mid X = x) = \eta_a(x_i, x_j) \qquad (i,j) \in \mathcal{N} \quad \mathbf{a} \in \mathcal{A} \quad c, d \in \mathcal{C} \tag{2.1}$$

It follows from the definition that $\sum_{\mathbf{a}} \eta_{\mathbf{a}}(c,d) = 1$, $\forall c, d \in \mathcal{C}$.
By symmetry of $\mathcal{N}$ we have that if $\mathbf{a} \in \mathcal{A}$ then $\pi(\mathbf{a}) \in \mathcal{A}$, where π is the reflection operator defined as $\pi(a_t, a_v) = (a_v, a_t)$. Therefore we must have

$$\eta_{\mathbf{a}}(c,d) = \eta_{\pi(\mathbf{a})}(d,c). \tag{2.2}$$

Since it will be useful in the inference part, we now partition the alphabet into two subsets:

- The set of symmetric relations $\mathcal{A}_0 = \{\mathbf{a} \in \mathcal{A} \mid \pi(\mathbf{a}) = \mathbf{a}\}$
- The set of asymmetric relations $\mathcal{A}_1 = \{\mathbf{a} \in \mathcal{A} \mid \pi(\mathbf{a}) \neq \mathbf{a}\}$

For instance, for a directed graph, $\mathcal{A}_0 = \{(0,0),(1,1)\}$ and $\mathcal{A}_1 = \{(0,1),(1,0)\}$. Now, the set $\mathcal{A}_1$ is redundant, since we have in the set both $\mathbf{a}$ and $\pi(\mathbf{a})$. Therefore we partition $\mathcal{A}_1$ in two subsets $\mathcal{A}_{10}$ and $\mathcal{A}_{11}$ in a way that $\mathbf{a} \in \mathcal{A}_{10}$ implies $\pi(\mathbf{a}) \in \mathcal{A}_{11}$ (notice that there are many ways to define $\mathcal{A}_{10}$ and $\mathcal{A}_{11}$). We define

$$\mathcal{A}' \coloneqq \mathcal{A}_0 \cup \mathcal{A}_{10} \tag{2.3}$$

and we denote by $r = |\mathcal{A}|$, $r_0 = |\mathcal{A}_0|$, $r_1 = |\mathcal{A}_{10}| = |\mathcal{A}_{11}|$, so that $|\mathcal{A}'| = r_o + r_1$ and $r = r_0 + 2r_1$.

2.4 Bayesian inference

What we know and what we want to find

In general, given a graph and a known number of classes k, our goal will be to recover the true attribute vector x^* and the true parameters π^* and η^*. We follow a fully Bayesian approach, using a Gibbs sampler as a Bayesian estimator. A note on the notation used: uppercase letters stands for random variables, lower case letters for their values. We define the Bayesian setting:

- From the assumption $\Pr(X_i = c) = \pi_c, \forall c \in \mathcal{C}$ independently, we have that the joint distribution of X is

$$\Pr(X_1 = x_1, ..., X_n = x_n \mid \pi) = \pi_1^{m_1} ... \pi_k^{m_k} \tag{2.4}$$

  where $m_c = \sum_{i=1}^{n} \mathbb{1}(x_i = c)$ is the number of nodes with class c.

- We define the relation count $e_{\mathbf{a}}(c, d)$ which counts the total number of relations $\mathbf{a}$ between a node of class c and one of class d:

$$e_{\mathbf{a}}(c,d) = (1 + \mathbb{1}\{c = d\}\mathbb{1}\{\mathbf{a} \in \mathcal{A}_0\})^{-1} \times \sum_{(i,j)\in\mathcal{N}} \mathbb{1}\{\mathbf{y}_{ij} = \mathbf{a}\}\,\mathbb{1}\{x_i = c\}\,\mathbb{1}\{x_j = d\} \tag{2.5}$$

  note that the first term is used to divide by two in the case $\mathbf{a}$ is symmetric and $x_i = x_j$, since we want to avoid double counting.

- By definition of the SBM, the distribution of Y given X, Π and H is given by:

$$\Pr(y \mid x, \pi, \eta) = \left(\prod_{\mathbf{a}\in\mathcal{A}} \prod_{1\le c<d\le k} (\eta_{\mathbf{a}}(c,d))^{e_{\mathbf{a}}(c,d)} \right) \times \left(\prod_{\mathbf{a}\in\mathcal{A}'} \prod_{c=1}^{k} (\eta_{\mathbf{a}}(c,c))^{e_{\mathbf{a}}(c,c)} \right) \tag{2.6}$$

  Just to mention, in the case of an undirected graph, the expression can be written in a more compact form:

$$\Pr(y \mid x, \pi, \eta) = \prod_{i=1}^{n} \prod_{j=i+1}^{n} (\eta(x_i, x_j))^{y_{ij}} (1 - \eta(x_i, x_j))^{1-y_{ij}} \tag{2.7}$$

- We have that by Law of Total Probabilities and by independence of X on H

$$\Pr(y, x \mid \pi, \eta) = \Pr(y \mid x, \pi, \eta) \Pr(x \mid \pi, \eta) = \Pr(y \mid x, \pi, \eta) \Pr(x \mid \pi) \tag{2.8}$$

therefore, from Eq. (2.4) and Eq. (2.6) the joint distribution of $Y, X \mid \Pi, H$ is given by

$$\begin{aligned} \Pr(y, x \mid \pi, \eta) =& \pi_1^{m_1} \cdots \pi_k^{m_k} \\ & \times \left(\prod_{\mathbf{a} \in \mathcal{A}} \prod_{1 \leq c < d \leq k} (\eta_{\mathbf{a}}(c, d))^{e_{\mathbf{a}}(c,d)} \right) \\ & \times \left(\prod_{\mathbf{a} \in \mathcal{A}'} \prod_{c=1}^{k} (\eta_{\mathbf{a}}(c, c))^{e_{\mathbf{a}}(c,c)} \right) \end{aligned} \tag{2.9}$$

- We assume a known prior density function for Π, H called $f_{\Pi,H}(\pi, \eta)$. Then, inference of the class membership is based on the so-called *posterior predictive distribution*:

$$\Pr(x \mid y) = \int f_{\pi,H,X|Y}(\pi, \eta, x \mid y) d\pi d\eta \tag{2.10}$$

and inference on the parameters π, η is based on the so-called *posterior distribution*:

$$f_{\Pi,H|Y}(\pi, \eta \mid y) = \sum_x f_{\Pi,H,X|Y}(\pi, \eta, x \mid y) \tag{2.11}$$

- Our aim is to obtain the conditional distribution

$$f_{\Pi,H,X|Y}(\pi, \eta, x \mid y) \tag{2.12}$$

from which we can recover both the posterior predictive distribution and the posterior distribution. However, we are not able to compute it analytically, but we will construct a Gibbs sampler in order to sample from it.

Gibbs Sampling

Gibbs sampling is a MCMC algorithm introduced by Geman and Geman in 1984 [29], it is a technique utilized to approximate the posterior distribution (or more generally, a joint distribution) with any desired accuracy. This method sequentially draws each unknown random variable or vector, conditioned on the values of all other random variables. As an example, suppose we want to sample $(z_1, ...z_n) \in \mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_n$ from a joint distribution $p(z_1, ...z_n)$ and that we know how to sample any z_i from $p(z_i \mid z_{-i})$, where z_{-i} means all z's except z_i. Then Gibbs sampling consists of the following:

Algorithm 1 Gibbs sampler

```
Initialize z_i^0 arbitrarily for i = 1, ..., n
for t = 1, 2, ... do
    for i = 1, ..., n do
        sample z_i^t from p(z_i^t | z_1^t, ..., z_{i-1}^t, z_{i+1}^{t-1}, ..., z_n^{t-1})
    end for
end for
```

The Gibbs sampler defines a Markov chain $(Z_t)_{t\in\mathbb{N}}$ on $\mathcal{X}$ where at each step only one component is updated.

Proposition 2.4.1 (Detailed Balance)**.** $(X_t)_{t\in\mathbb{N}}$ *satisfies detailed balance with respect to the joint distribution* $p(z_1, ..., z_n)$.

Proof.

For simplicity consider the two-dimensional case. The extension is trivial. Suppose we are in a time step where we have to sample z_1*, then:*

$$\begin{aligned}
\mathrm{p}(z_1, z_2)\Pr((z_1, z_2) \to (y_1, z_2)) &= \mathrm{p}(z_1, z_2)\, p(y_1 \mid z_2) = \mathrm{p}(z_1, z_2)\frac{\mathrm{p}(y_1, z_2)}{\sum_{z_1'} \mathrm{p}(z_1', z_2)} \\
&= \mathrm{p}(y_1, z_2)\frac{\mathrm{p}(z_1, z_2)}{\sum_{z_1'} \mathrm{p}(z_1', z_2)} = \mathrm{p}(y_1, z_2)\, p(z_1 \mid z_2) \\
&= \mathrm{p}(y_1, z_2)\Pr((y_1, z_2) \to (z_1, z_2)).
\end{aligned}$$

When we have to sample z_2 *the proof would be almost the same.* □

Therefore, if the conditional distributions are such that the Markov chain is aperiodic and irreducible (which is often the case), the Gibbs sampler converges to the joint distribution

as $t \to \infty$.

Gibbs Sampler for the SBM

In our case, we want to be able to sample from $f_{\Pi,H,X|Y}(\pi, \eta, x \mid y)$. To do so, we consider (π, η) as a unique vector and we consider each x_i as a distinct element. That is, to do an analogy with the example of before, we would have $(z_1, ..., z_{n+1}) = ((\pi, \eta), x_1, ..., x_n)$, everything conditioned on y.
In order to apply the Gibbs algorithm, we need to be able to sample from:

a) $f_{\Pi,H|X,Y}(\pi, \eta \mid x, y)$

b) $\Pr(x_i \mid \pi, \eta, x_{-i}, y)$ for all $i = 1, ..., n$

For b) we have by definition of conditional probability

$$\Pr(x_i \mid \pi, \eta, x_{-i}, y) = \frac{\Pr(y, x \mid \pi, \eta)}{\Pr(y, x_{-i} \mid \pi, \eta)} \tag{2.13}$$

As seen before, we know how to compute $\Pr(y, x \mid \pi, \eta)$ (Eq.(2.9)), and, we can compute $\Pr(y, x_{-i} \mid \pi, \eta)$ by:

$$\Pr(y, x_{-i} \mid \pi, \eta) = \sum_{c'} \Pr(y, x_{-i}, X_i = c' \mid \pi, \eta) \tag{2.14}$$

From Eq.(2.13), Eq.(2.9) and Eq.(2.14) one can obtain:

$$\Pr(X_i = c \mid \pi, \eta, x_{-i}, y) = Q\pi_c \prod_{\mathbf{a} \in \mathcal{A}} \prod_{d=1}^{k} (\eta_{\mathbf{a}}(k, d))^{\mathrm{d}_{\mathbf{a}}(i,d)} \tag{2.15}$$

Where Q is a normalizing constant and where $\mathrm{d}_{\mathbf{a}}(i, d)$ is the number of nodes j such that $x_j = d$ and $y_{ij} = \mathbf{a}$:

$$\mathrm{d}_{\mathbf{a}}(i, d) = \sum_{j:(i,j) \in \mathcal{N}} \mathbb{1}\{y_{ij} = \mathbf{a}\} \mathbb{1}\{x_j = d\} \tag{2.16}$$

Thus we can compute expression b).

For a) we need a specification of the prior distribution $f_{\Pi,H}(\pi, \eta)$. If the number of vertices is large enough and the prior relatively flat, the choice of the prior does not have a large influence on the results. It is often reasonable to assume prior independence between π and η. In this analysis, we assume the prior for π to be a Dirichlet $\mathrm{Dir_k}(T, ..., T)$ where T is a hyper-parameter. The Dirichlet prior has a strong tendency to favour unequal classes when T is small, leading to the risk that in early stages the parameters will be trapped in a region where some of the class sizes are almost 0; for this reason, Snijders and Nowicki proposed to use $T = 100k$, which seemed to work well in practice.

Because of the redundancy 2.2 of the η parameter, we need to take more care in defining the prior for η. For $c < d$, $\eta(c, d) = (\eta_{\mathbf{a}}(c, d))_{\mathbf{a}\in\mathcal{A}}$ is an unconstrained r-dimensional vector (and the case $c > d$ can be obtained directly from the case $c < d$), while for $c = d$, this vector is subject to the constraint $\eta_{\mathbf{a}}(c, c) = \eta_{\pi(\mathbf{a})}(c, c), \forall \mathbf{a} \in \mathcal{A}_{10}$ which amounts to impose r_1 equality relations between the elements of $\eta(c, c) = (\eta_{\mathbf{a}}(c, c))_{\mathbf{a}\in\mathcal{A}}$. One can define a new vector

$$\eta^{(0)}_{\mathbf{a}}(c, c) = \begin{cases} \eta_{\mathbf{a}}(c, c) & (\mathbf{a} \in \mathcal{A}_0) \\ 2\eta_{\mathbf{a}}(c, c) & (\mathbf{a} \in \mathcal{A}_{10}) \end{cases} \tag{2.17}$$

which is a (r_0+r_1)-dimensional vector without redundant elements, and use this in the Gibbs sampler since there is a one-to-one map between $\eta(c, c)$ and $\eta^{(0)}(c, c)$.

For the choice of the prior, we still use a Dirichlet prior, which will be a r-dimensional Dirichlet $\mathrm{Dir_r}(1, ..., 1)$ for $\eta(c, d), c < d$ and a $(r_0 + r_1)$-dimensional Dirichlet $\mathrm{Dir_{r_0+r_1}}(1, ..., 1)$ for $\eta^{(0)}(c, c)$.

Remark 2.4.1 (Dirichlet distribution). *The Dirichlet distribution of order $K \geq 2$ with parameters $\alpha_1, \ldots, \alpha_K > 0$, denoted* $\mathrm{Dir_K}(\alpha_1, \ldots, \alpha_K)$*, has a probability density function with respect to Lebesgue measure on the Euclidean space $\mathbb{R}^{K-1}$ given by*

$$f\left(x_1, \ldots, x_K; \alpha_1, \ldots, \alpha_K\right) = \frac{1}{\mathbf{B}(\boldsymbol{\alpha})} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{2.18}$$

where $\{x_i\}_{i=1}^K$ belong to the standard $K-1$ simplex. The normalizing constant is the multi-

variate beta function, defined in terms of the gamma function as:

$$\mathbf{B}(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}, \quad \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K). \tag{2.19}$$

Thanks to Dirichlet priors, it is possible to use well-known results on the Bayesian analysis of multinomially distributed data to derive the posterior distribution of (π, η) used by the Gibbs sampler. The result we will use is the Conjugate Prior Theorem for Dirichlet Distributions, which in our case amounts to the following:

Proposition 2.4.2. *If the prior distribution of π is Dirichlet with parameters $(T_c)_{c\in\mathcal{C}}$ and the prior distribution of $\eta(c,d)$ is Dirichlet with parameters $E_{\mathbf{a}}(c,d)$, while π and the $\eta(c,d)$ are a priori independent, then the posterior distribution of (π,η), given the complete data (y,x), is given by independent Dirichlet distributions with parameters*

$$\begin{aligned} &(m_c + T_c)_{c\in\mathcal{C}} && \text{for } \pi \\ &(e_{\mathbf{a}}(c,d) + E_{\mathbf{a}}(c,d))_{\mathbf{a}\in\mathcal{A}} && \text{for } \eta(c,d), 1 \le c < d \le k \\ &(e_{\mathbf{a}}(c,c) + E_{\mathbf{a}}(c,c))_{\mathbf{a}\in\mathcal{A}'} && \text{for } \eta^{(0)}(c,c), 1 \le c \le k \end{aligned} \tag{2.20}$$

We can use this result to complete step a) of the Gibbs sampler, using as written before $T_c = 100k, \forall c$ and $E_{\mathbf{a}}(c,d) = 1, \forall \mathbf{a} \in \mathcal{A}, c, d \in \mathcal{C}$ with $c \le d$. To summarize, this is the final Gibbs Sampler we will use:

Algorithm 2 Gibbs sampler for SBM

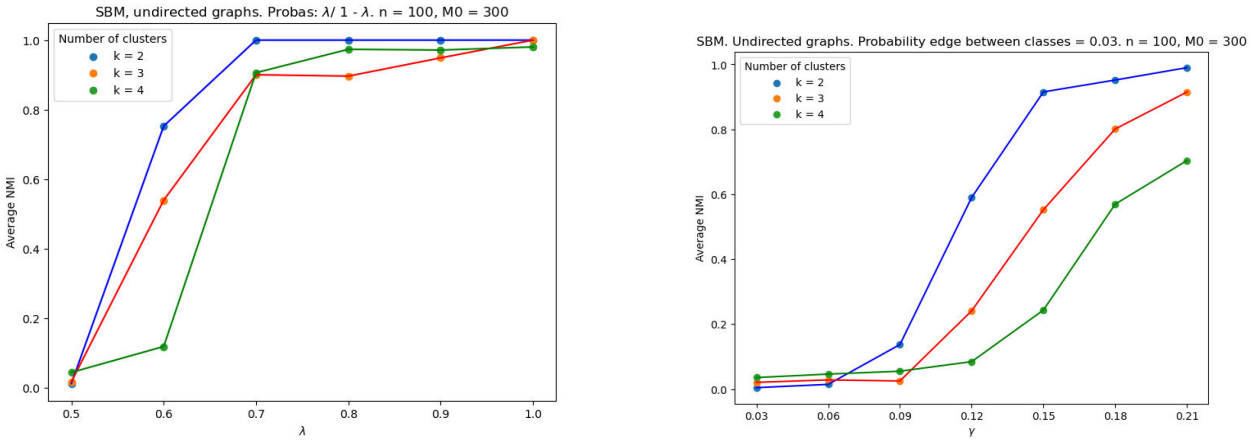
1: Initialize x_i^0 arbitrarily for $i = 1, ..., n$
2: **for** $t = 1, ..., t_{max}$ **do**
3: Sample $\pi = (\pi_1, ..., \pi_k)$ from $\mathrm{Dir_k}(m_1 + T, ..., m_k + T)$.
4: Sample $\eta(c,d)$, $\forall c, d : c < d$ from the r-dimensional $\mathrm{Dir_r}((1 + e_{\mathbf{a}}(c,d))_{\mathbf{a}\in\mathcal{A}})$
5: Sample $\eta^{(0)}(c,c), \forall c$ from the $(r_0 + r_1)$-dimensional $\mathrm{Dir_{r_0+r_1}}((1 + e_{\mathbf{a}}(c,c))_{\mathbf{a}\in\mathcal{A}'})$
6: **for** $i = 1, ..., n$ **do**
7: sample x_i^t from $\Pr(x_i^t \mid \pi, \eta, x_1^t, ..., x_{i-1}^t, x_{i+1}^{t-1}, ..., x_n^{t-1}, y)$
8: **end for**
9: **end for**

2.5 Simulations

We can proceed with the simulation of the Gibbs sampler to analyze its performance on synthetic data. The implementation of the SBM can be found in SBM_ext.py. Firstly I have created a class called *StochasticBlockModel_General* which generates random graphs with the distribution of the general SBM, given $n, \mathcal{N}, k, \pi$, an alphabet of relations $\mathcal{A}$ and the η matrix. Thereafter, the class *Gibbs_sampler_General* is used to perform the Gibbs sampling. It is initialized with the adjacency matrix Y, the set of observed relations $\mathcal{N}, k$ and $\mathcal{A}$.
Nowicki and Snijders proposed a method to improve convergence based on assigning initially $T = 10n$ and then decreasing it linearly to $T = 100k$ in M_0 iterations; in the meanwhile, one has to multiply the parameters of the Dirichlet $e_{\mathbf{a}}(c, d) + E_{\mathbf{a}}(c, d)$ by a factor ω which starts from $\frac{1}{n}$ and it is increased linearly to 1, provided that the resulting product is greater or equal than 1 (otherwise set it to 1). After the first M_0 iterations, the algorithm continues with other M_0 iterations with $T = 100k$. After that, it should start to check for convergence and stop when convergence is reached. It is possible to run the Gibbs sampler using the function *sample_improved(M_0)*, where the method used to improve convergence is implemented, but after the $2M_0$ iterations the function simply stops and returns the current configuration. It is also possible to run the Gibbs sampler without the improvement, using the function *sample(t_{max})*.
We can do some simple checks, and it is easy to see that the model works both in the case of high connectivity within the cluster and low connectivity between clusters, or vice versa: this is indeed what was anticipated in the introduction.
As a metric to understand the performance of the inference, I used the normalized mutual information (NMI), which is a value between 0 and 1, the higher the better, and tells us how much information we gain of the true classes given the knowledge of the predicted ones. The relationship between the NMI and the number of errors depends on the number of classes and on their prior probability. For instance, if $n = 100$, $k = 2$ and the two classes are equally probable, having 2 misclassified nodes out of 100 produces an NMI of around 0.86, while the same with $k = 3$ would produce an NMI of around 0.92 (thus higher as k increases).

Undirected graphs

The first analysis goes as follows: let $n = 100$, $k \in \{2, 3, 4\}, \pi_c = \frac{1}{k} \forall c \in \mathcal{C}$. We consider undirected graphs, and we set $\eta_{(1,1)}(c, c) = \lambda$ and $\eta_{(1,1)}(c, d) = 1 - \lambda$, so that there is an edge with probability λ within classes and $1 - \lambda$ between classes. Then we study how the NMI depends on λ. The results are shown in Figure 2.1a. Each data point is obtained from the average NMI of 7 different simulations (with different graphs). The total number of iterations for each simulation is set to 600, that is, $M_0 = 300$, even though Nowicki and Snijders considered values much larger ($M_0 = 5000$); therefore what I did is to check for fast convergence, even though with an increased number of iterations the performances could slightly improve.
As expected, the average NMI is around 1 for high values of λ, while it is 0 for $\lambda = 0.5$, since it is the case of no structure in the model. In all the cases $k = 2, 3, 4$ the model is able to recover quite well the underlying structure until $\lambda = 0.7$. With $\lambda = 0.6$ only in the case $k = 2$ it manages to maintain good performance. For symmetric reasons, the results would be exactly symmetric in the case $\lambda \in [0, 0.5]$.



(a) Simulation 1: λ / $1 - \lambda$ probabilities

(b) Simulation 2: Low expected degree

Figure 2.1: Results of simulations on undirected graphs

In this previous simulation we were considering highly connected graphs, however, real networks are often sparse. If we set the probability of edges between different classes fixed to 0.03, what is the probability γ of having an edge within classes needed to recover the block structure? The results are shown in Figure 2.1b. Note that the expected degree of each vertex is given by $0.03 * 50 + \gamma * 49 \approx \frac{3+100\gamma}{2}$. It is clear from the results that the lower ex-

pected degree of each vertex diminishes the performance, or at least, makes the convergence slower. For instance, in the case $\gamma = 0.12$ (expected degree = 7.5) the probability of having an edge within classes is 4 times higher than the one between classes, however, the NMI is much lower than the corresponding case $\lambda = 0.8$ in the example before. Increasing k in this case diminishes the performances, indeed with this model as k increases the expected degree decreases, making the convergence slower.

Undirected graphs

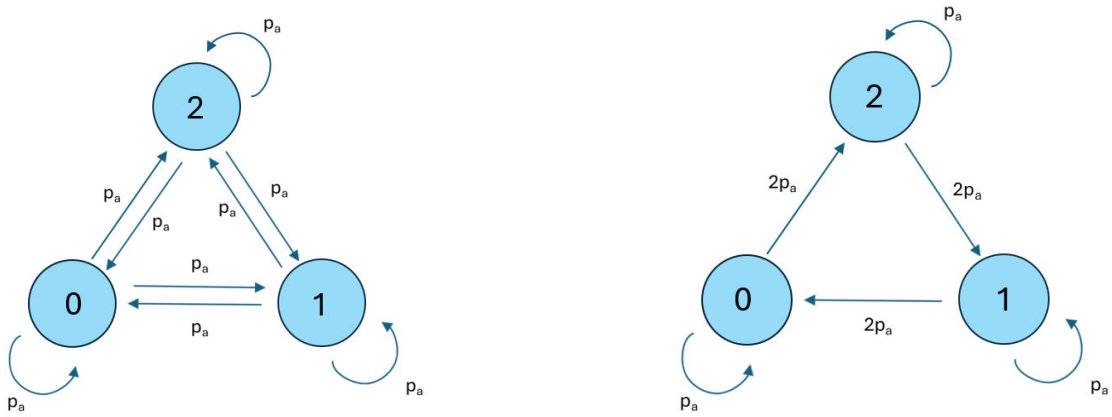
To verify the performance of the SBM with undirected graphs, I tried a model with $n = 70, k = 3$ and η given by :

$$\eta = \lambda\eta^{\text{planted}} + (1 - \lambda)\eta^{\text{random}} \tag{2.21}$$

that is, we interpolate linearly between η^{random} and η^{planted}. η^{random} represents a model such that there is a directed edge from node i to node j with probability p_a independently on the classes of i and j[1], while η^{planted} represents a sort of "rock, paper, scissor" model. It is such that the probability of having a directed edge from nodes of class 2 to 1, from 1 to 0 and from 0 to 2 is $2p_a$, the probability of one within the same class is p_a as before, and 0 else. Increasing λ is as if we shifted the probability of connecting from the class with which we "lose" to the one with which we "win". See Figure 2.2 for an illustration.

I used $p_a = 0.15$, so that the expected in and out degrees of each node are around 10, with the same hyper-parameters as before. The results of the simulations are shown in figure 2.3 The average NMI is around 0 for $\lambda \leq 0.2$, and it grows linearly until $\lambda = 0.7$, when it reaches 1, and remains 1 for $\lambda \in [0.7, 1]$.

[1]Note that it is possible to have at the same time an edge from i to j and one in the opposite direction.



(a) $\lambda = 0$ (b) $\lambda = 1$

Figure 2.2: Illustration of the extrema cases of the model.

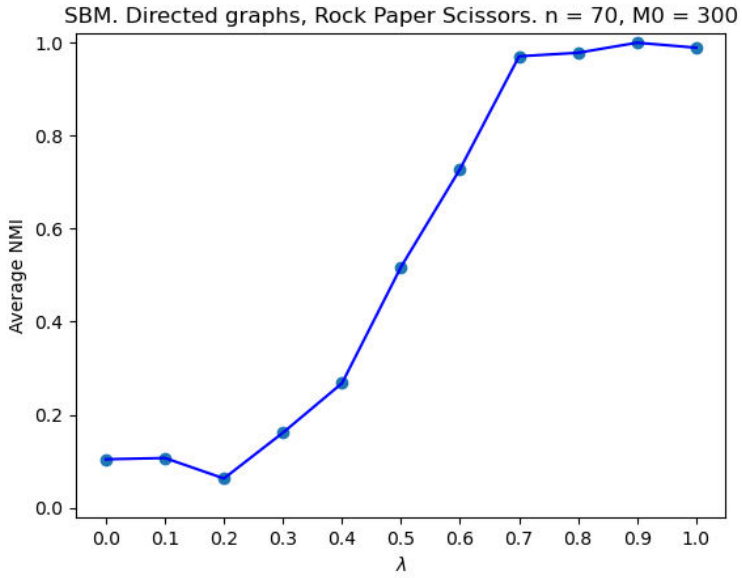


Figure 2.3: Rock, Paper, Scissors model

Chapter 3

MFM Stochastic block model

Models shown before assumed the knowledge of the true number of classes; in the case it is not given, one can estimate it a priori using some selection criteria, and after that apply the algorithm described before (or different ones). However, disregarding the uncertainty in the initial phase may result in inaccurate clustering, especially when the community structure is vague. Geng et al. (2019) [28] proposed to apply the methods for Mixture of finite models (MFMs) described by Miller and Harrison (2015) [30] which are based on the ones used for the Dirichlet process mixture (DPMs), in order to build a model that does not assume any known number of classes.
In this chapter we first introduce mixture models, afterwards we study the MFM Stochastic Block Model and we use the results from mixture models to build an inference algorithm for the MFM-SBM and lastly, we simulate the algorithm.

3.1 Mixture models

A mixture model is a statistical model that represents the probability distribution of data points as a combination of multiple probability distributions. Mixture models are often used in situations where the data is believed to arise from multiple underlying components and the goals are to estimate the parameters of these components and to assign each data point to the most likely component. They are widely used in clustering, density estimation, and model-based classification tasks. A common issue with finite mixtures (when the number of components is finite, call it k), is that it could be difficult to find the correct k. The most natural approach is called Mixture of finite mixtures (MFM): in a full Bayesian perspective, one treats k like any other unknown parameter and considers a prior on it. Several algorithms have been proposed for MFM, the most commonly used are based on reversible jump Markov

chain Monte Carlo. Even though reversible jump is a general technique, it is often complicated to use since it requires to design of good reversible jump moves, which can be nontrivial.
At the same time, infinite mixture models, such as Dirichlet Process Mixture (DPM), became popular and well-studied thanks to the fact that there is a general Markov chain Monte Carlo algorithm which is simple to implement and to adapt for specific cases. Miller and Harrison (2015) [30] found a way to rewrite the MFM model through various representations used for DPMs, letting one use the known methods of DPMs for MFMs.
First, we define the Mixture of finite mixtures (MFM) model rigorously:

Definition 3.1.1. *The MFM model is defined by the following hierarchical process:*

1. *Sample the number of components (classes):*
   $K \sim p_K$, *where* p_K *is the prior p.m.f. on* $\{1, 2, ...\}$

2. *Sample the prior probability distribution over components:*
   $\pi = (\pi_1, ..., \pi_k) \sim \mathrm{Dir_k}(\gamma, ..., \gamma)$ *given* $K = k$, *where* γ *is a parameter of the model*

3. *Sample the component for each data point:*
   $X_1, ..., X_n \overset{iid}{\sim} \pi$ *given* π

4. *Sample the parameters of each component:*
   $\theta_1, ..., \theta_K \overset{iid}{\sim} H$ *given* K, *where* H *is a prior on the parameter space* Θ

5. *Sample the data you observe:*
   $Y_i \sim f_{\theta_{X_i}}$ *independently for* $i = 1, ..., n$ *given* θ, X

Therefore the parameters of the model are the prior distribution on the number of components p_K, the parameter for the Dirichlet distribution γ, the parameter space Θ, the prior on Θ H and a parametric distribution f defined on the space of the space of the observed data.
Dirichlet mixture processes, since they are infinite mixture models, don't assume the existence of a finite number of components k: they assume instead that the number of components is infinite. We first write a representation of a DPM, called Chinese Restaurant Process (CRP), also called Blackwell–MacQueen urn process, which is probably easier to understand and also useful for our purposes, and afterwards we discuss for comparison with MFM the formal definition of DPM.

A CRP is described through the Chinese restaurant metaphor: suppose we are in a Chinese restaurant with infinite tables, where each table represents a cluster, labelled $1, 2, 3, ...$; the first customer is seated at the first table, that is $x_1 = 1$ and every time a new customer i arrives, he seats at the table c with probability

$$P\left(x_i = c \mid x_1, \ldots, x_{i-1}\right) \propto \begin{cases} |c|, & \text{at an existing table labeled } c \\ \alpha, & \text{if } c \text{ is a new table.} \end{cases} \tag{3.1}$$

where $|c|$ is the number of costumers at the table c.

The same probability distribution for $\{X_i\}_{i=1,\ldots,n}$ can be obtained also by defining a discrete time stochastic process $(\mathcal{C}_i)_{i=1,\ldots,n}$ where at each time i the value of the process is a partition $\mathcal{C}_i$ of the set $\{1, 2, ..., i\}$. The process is determined as follows (assuming $\alpha = 1$ for simplicity): at time $i = 1$ we set $\mathcal{C}_i = \{1\}$; then, at time $i + 1$, we either add the element $i + 1$ to one of the blocks c of the partition $\mathcal{C}_i$ with probability $\frac{|c|}{i+1}$ for each block, where $|c|$ is the size of the block, or we add the singleton $\{i + 1\}$ to the partition $\mathcal{C}_i$.

For comparison, we write the definition of the Dirichlet Process Mixture

Definition 3.1.2 (Dirichlet Process Mixture)**.** *The DPM model is defined by the following hierarchical process:*

1. *Sample the parameters for π:*
   $B_1, B_2, ... \overset{iid}{\sim} Beta(1, \alpha)$

2. *Sample the component for each data point:*
   $X_1, ..., X_n \overset{iid}{\sim} \pi$ *given* $\pi = (\pi_1, \pi_2, ...)$ *with* $\pi_c = B_c \prod_{d=1}^{c-1}(1 - B_d)$

3. *Sample the parameters of each component:*
   $\theta_1, \theta_2, ... \overset{iid}{\sim} H$*, where* H *is a prior on the parameter space* Θ

4. *Sample the data you observe:*
   $Y_i \sim f_{\theta_{X_i}}$ *independently for* $i = 1, ..., n$ *given* $\theta_{1:\infty}, X_{1:n}$

$B_1, B_2, ...$ are needed to construct π, the prior probability distribution over the classes. One can think of the intuition behind the B_js in a way similar to the geometric distribution: in order to be in class d you have to "fail" to be in class $1, ..., s - 1$ and then "succeed" to be

in d, where $\forall c = 1, 2, ..., B_c$ is the probability of being in class c given that you are not in classes $1, ..., c-1$.

A possible approach for our original problem, that is stochastic block model with an unknown number of components, or also for a more general clustering problem with an unknown number of clusters, would be to model the underlying process as a DPM and then make inference based on it. This is a possibility and from some point of view it looks very similar to an MFM model, however, there are two principal differences: firstly, the prior on the final number of clusters t is very different (here we use the following notation: clusters are the groups that we find, components the ones that the model has, which are infinite for a DPM); in an MFM one has complete control over the prior on the number of components k and as the sample size n increases, t converges to k almost surely. Instead, in a DPM, the prior on t has a particular parametric form and diverges with a $\log n$ rate as n goes to ∞ . Secondly, given the number of clusters t, the prior on the size of each cluster differs a lot between MFM and DPM; in MFM most of the prior mass is on partitions with cluster sizes of the same order of magnitude; in contrast, for DPM it is on partitions where cluster's sizes vary widely, having a few large clusters and many very small ones.

As a result, DPM tends to overestimate the number of clusters, creating new clusters with few elements. For this reason, MFM is preferable. Miller and Harrison proved that it is possible to write a Mixture of finite mixtures with a CRP representation (and also other representations of a DPM) and thus to adapt a Gibbs sampler used for Dirichlet Processes Mixtures to one for the MFM case.

The result is the following CRP representation of the MFM model:

1. Initialize with a single cluster consisting of element 1 alone:
   $\mathcal{C}_1 = \{\{1\}\}$

2. For $i = 2, 3, \ldots$, place element i in

   (a) an existing cluster $c \in \mathcal{C}_{i-1}$ with probability $\propto |c| + \gamma$

   (b) a new cluster with probability $\propto \frac{V_i(t+1)}{V_i(t)}\gamma$

   where $t = |\mathcal{C}_{i-1}|$.

and where the coefficient of partition distribution $V_i(t)$ is defined as

$$V_i(t) = \sum_{k=1}^{+\infty} \frac{k_{(t)}}{(\gamma k)^{(i)}} p(k) \tag{3.2}$$

where $k_{(t)} = k(k-1)\dots(k-t+1)$, and $(\gamma k)^{(i)} = \gamma k(\gamma k+1)\dots(\gamma k+i-1)$. (By convention, $x^{(0)} = 1$ and $x_{(0)} = 1$). With respect to the classical CRP, it is as if we were decreasing the rate of creation of new clusters/tables by a factor $V_i\left(|\mathcal{C}_{i-1}|+1\right)/V_i\left(|\mathcal{C}_{i-1}|\right)$, avoiding the creation of too many new clusters/tables.

Given a partition $\mathcal{C}$, for $c \in \mathcal{C}$ we denote $y_c = (y_i : i \in c)$ and the marginal likelihood $m\left(y_c\right) = \int_\Theta \left[\prod_{i\in c} f_\theta\left(y_i\right)\right] H(d\theta)$, with the convention that $m\left(y_\emptyset\right) = 1$, thus having:

$$p(y_{1:n} \mid \mathcal{C}) = \prod_{c\in\mathcal{C}} m(y_c) \tag{3.3}$$

When H is a conjugate prior for f_θ, such that the marginal likelihood $m\left(y_c\right)$ can be easily computed, the following Gibbs sampling algorithm can be used to sample from the posterior distribution on partitions $p\left(\mathcal{C} \mid x_{1:n}\right)$:

Algorithm 3 Gibbs sampler for a MFM in CRP representation

Initialize $\mathcal{C} = \{[n]\}$ ▷ i.e., one cluster.
for $\tau = 1, ..., \tau_{max}$ **do**
  for $i = 1, \dots, n$ **do**
    Remove element i from $\mathcal{C}$ and place it...
      in $c \in \mathcal{C}_{-i}$ with probability $\propto (|c|+\gamma)\frac{m(y_{c\cup i})}{m(y_c)}$
      in a new cluster with probability $\propto \gamma \frac{V_n(t+1)}{V_n(t)} m\left(y_i\right)$

  end for
end for

Where $\mathcal{C}_{-i}$ is the partition obtained by removing i and where $t = |\mathcal{C}_{-i}|$. This algorithm is a direct adaptation of a known algorithm for DPMs (Maceachern, 1994 [31]).

3.2 MFM Stochastic block model

We now consider the case of undirected graphs (which is exactly the case of the Bernoulli SBM). We Consider the usual notation defined before. Then we define our model as :

Definition 3.2.1 (MFM Stochastic Block Model, Bernoulli)**.**

$$
\begin{aligned}
&K \sim p_k, \text{ where } p_k \text{ is a p.m.f on } \{1, 2, \ldots\} \\
&\eta_{cd} = \eta_{dc} \overset{\text{ind}}{\sim} \text{Beta}(a, b), \text{ for } c, d = 1, \ldots, K, \\
&\pi \mid K \sim \text{Dir}_{\text{K}}(\gamma, \ldots, \gamma), \\
&X_i \mid \pi, K \sim \pi \text{ for } i = 1, ..., n \\
&Y_{ij} = Y_{ji} \mid X, \eta, K \overset{\text{ind}}{\sim} \text{Bernoulli}\left(\theta_{ij}\right), \quad \theta_{ij} = \eta_{x_i x_j}, \quad 1 \leq i < j \leq n.
\end{aligned}
$$

That is, we define a prior p_k for the number of classes K and then given the value of $K = k$, we assume that for each pair of classes (c, d), $\eta_{cd} = \eta_{dc}$ come from a Beta(a, b) prior (which is actually the special case of the Dirichlet when considering only two possible relations), with a, b hyper-parameters. π, the probability distribution over classes, comes from a Dirichlet prior with parameter γ (that in the standard SBM was called T), and finally for each pair of nodes (i, j), $i \neq j$ $Y_{ij} = Y_{ji}$ are given by a Bernoulli prior, with parameter $\theta_{ij} = \eta_{X_i X_j}$ as usual. This is exactly the same model as the undirected Bernoulli SBM, except that now K is not fixed but it is a parameter of the model. A default choice of p_k is a Poisson(1) distribution truncated to be positive, which is assumed through the rest of the chapter.
When we do a posteriori blockmodelling, we want to infer the unknown parameters k, η, x given what we know: Y (in this case we don't infer π). To do so we would like to apply the algorithm 3 described above to our specific model. However, the MFM Stochastic Block Model is actually different from a Bayesian Mixture Model, indeed if we consider the observed data Y_i to be the i-th row of the adjacency matrix Y, then in a Bayesian Mixture Model Y_i would depend only on X_i, but this is not true in the MFM-SBM since it depends also on the classes of the other nodes. Geng et al., using the same approach of Miller et al., adapted Algorithm 3 to the specific case of the MFM-SBM. In particular, we are in the case where H is a conjugate prior for f_θ and therefore it is easy to obtain the marginal likelihood m. The final algorithm is the following:

Algorithm 4 Collapsed sampler for MFM-SBM

Initialize x and η.
for $\tau = 1, ..., \tau_{max}$ **do**
  Update η conditional on x as
    $$\eta_{cd} \sim \text{Beta}\left(\bar{Y}_{[cd]} + a, r_{cd} - \bar{Y}_{[cd]} + b\right) \qquad c, d \in \{1, ..., k\}, c \leq d$$
  for $i = 1, \ldots, n$ **do**
    Update x_i from $\Pr\left(x_i = c \mid x_{-i}, Y, \eta\right)$
    $$\propto \begin{cases} [|c| + \gamma] \left[\prod_{j \neq i} \eta_{cx_j}^{Y_{ij}} \left(1 - \eta_{cx_j}\right)^{(1-Y_{ij})}\right] & \text{at an existing table c} \\ \gamma \frac{V_n(|\mathcal{C}_{-i}|+1)}{V_n(|\mathcal{C}_{-i}|)} m\left(Y_i\right) & \text{if c is a new table} \end{cases}$$
  end for
end for

Where k is the number of clusters formed by the current x,

$$\bar{Y}_{[cd]} = \sum_{(i,j):x_i=c, x_j=d, i \neq j} Y_{ij}, \tag{3.4}$$

r_{cd} is defined as:

$$r_{cd} = \sum_{(i,j):i \neq j} \mathbb{1}\left(x_i = c, x_j = d\right) \qquad c, d \in \{1, .., k\} \tag{3.5}$$

and

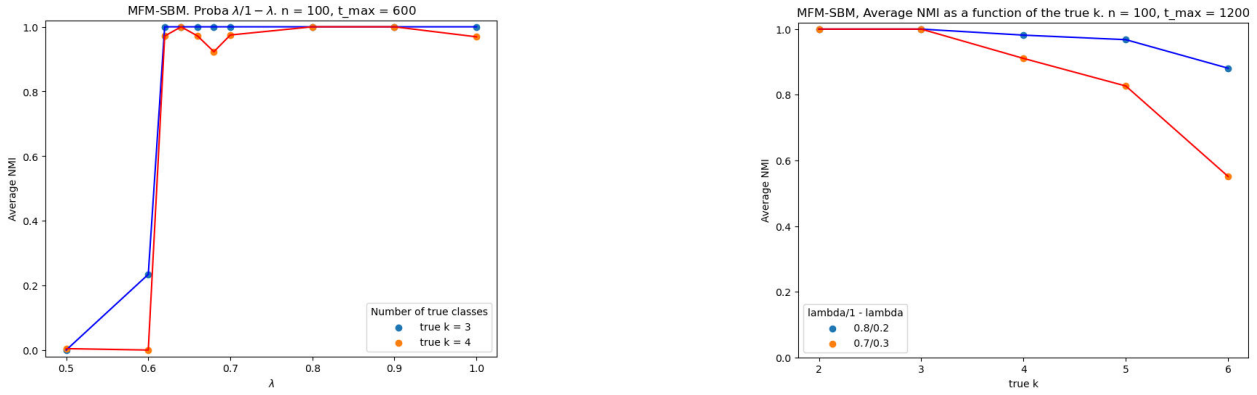
$$m\left(Y_i\right) = \prod_{d \in \mathcal{C}_{-i}} \left[\mathbf{B}(a, b)\right]^{-1} \mathbf{B}\left(\sum_{j \in d, j \neq i} Y_{ij} + a, |d| - \sum_{j \in d, j \neq i} Y_{ij} + b\right). \tag{3.6}$$

In this last equation $\mathbf{B}(a, b)$ is the two-dimensional Beta function defined in remark 2.4.1.

3.3 Simulations

Implementation of the MFM-SBM model can be found in MFM_SBM.py. I used the same generative SBM as before, in particular in the case of undirected graphs. The class *Collapsed_Gibbs_Sampler* implements the Collapsed Gibbs sampler used for the MFM-SBM. In order to initialize the Gibbs sampler, only the adjacency matrix Y is required; moreover, one can also choose to change the hyper-parameters, which are γ, a, b described before and k_{init},

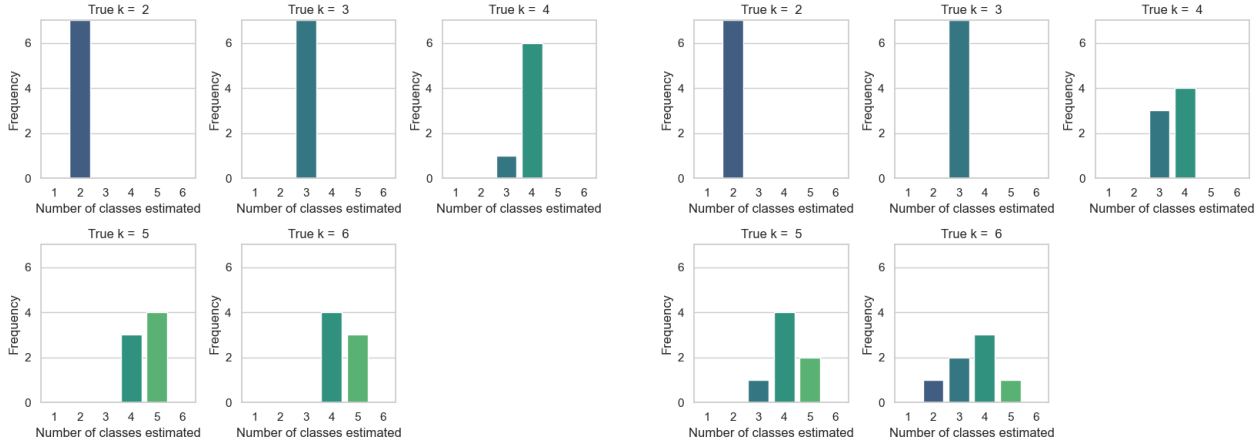
which is the number of classes to start with [1]. In this analysis I used $\gamma = 1, a = 1, b = 1$ and $k_{init} = 10$. The function *sample*(t_{max}) is used to run t_{max} iterations of the Gibbs sampler.
Figure 3.1a shows the NMI of the MFM-SBM in recovering the structure of a network with $k = 3$ (blue) and $k = 4$ (red) and probability of edges λ within the same class and $1 - \lambda$ between different classes. It seems that a phase transition occurs, with a critical point between 0.6 and 0.62
Instead, figures 3.1b and 3.2 show the performances of the MFM-SBM for a fixed λ but varying the number of true classes. The number of iterations is set to 600 in the first simulation and to 1200 in the second, since for larger ks more iterations are required.



(a) Simulation 1: Varying λ.

(b) Simulation 2: Varying the true k.

Figure 3.1: NMI of the simulations



(a) $\lambda = 0.8$

(b) $\lambda = 0.7$

Figure 3.2: Simulation 2: Predicted number of classes.

[1] indeed we are not required to start only with one class.

Chapter 4

Degree-corrected stochastic block model

4.1 Introduction

From the definition of the (generative) stochastic block model we have used until, in the case of undirected graphs, every node within a given class has the same degree distribution and this distribution is a Poisson binomial distribution, indeed the degree K of a node i of class c is given by the sum $K_1 + ... + K_k$, where k is the number of classes and each K_d represents the number of edges between the node i and nodes of class d, which is given by a Binomial distribution with parameter the number of nodes in t (minus 1 if $c = d$) and the probability that a node of class c has an edge with one of class d (which is η_{cd}). This distribution is similar in spirit as the Binomial (and the Poisson) distribution, in the sense that they are peaked around the mean and they do not allow high degree variability. As a result, the stochastic block model does not account for degree heterogeneity. This is particular problematic when we actually have a degree variability within classes: with the stochastic block models considered before we would end up dividing, for instance, between high-degree and low-degree nodes, and not between classes. As we have seen in Chapter 1, most real networks follow a power law, or similar, degree distribution, and thus have high variability of degrees. For this reason, the standard stochastic block model works poorly on real data. In this section we are going to solve this problem, following the approach of Karrer and Newman (2011) [26] and developing the so-called Degree-corrected stochastic block model. Note that, the MFM extension described in the past chapter, can be applied, with some minor changes, to the degree-corrected SBM (which was indeed proposed before). Karrer and Newman used another approach to do a posteriori blockmodelling, which is based

on Maximum likelihood estimation and a greedy algorithm (and not on Bayesian statistics and Gibbs sampling), therefore we will follow a different approach to what followed so far.

4.2 A new definition for the standard Stochastic Block Model

In order to simplify calculations, we change the definition of the stochastic block model: we now allow for multi-edges, that is, more than one edge connecting two nodes, and self-edges, that is edges with endpoints on the same node. Even though many real-world applications do not allow for such edges, allowing multi-edges and self-edges is something often done for random graph models for sparse networks since the difference vanishes as the size of the network n becomes large.

Now, since we are allowing multiple edges, instead of considering the probability of an edge between two nodes of classes c, d respectively, we consider the expected number of edges, called $\psi_{c,d}$. The actual number of edges between i and j will be given (independently) by a Poisson distribution with mean $\psi_{c,d}$. Notice that, in the limit of n large for sparse graphs, there is no difference between this model and the canonical one, since the expected number of edges becomes equal to the probability of an edge.

Let G be a realization of the model; we denote as usual by Y the adjacency matrix of G, where Y_{ij} is equal to the number of edges between node i and node j if $i \neq j$ and twice that number if $i = j$. We let $\omega_{c,d}$ be equal to $\psi_{c,d}$ if $c \neq d$ and equal to $2\psi_{c,d}$ if $c = d$, which is the expected value of Y_{ij} with $X_i = c$ and $X_j = d$. Thus, the probability of a graph G (which is equivalent to the probability of Y) given Ω, X is given by:

$$\Pr(G \mid \omega, x) = \prod_{(i,j):i<j} \frac{\left(\omega_{x_i x_j}\right)^{Y_{ij}}}{Y_{ij}!} \exp\left(-\omega_{x_i x_j}\right) \times \prod_i \frac{\left(\frac{1}{2}\omega_{x_i x_i}\right)^{Y_{ii}/2}}{(Y_{ii}/2)!} \exp\left(-\frac{1}{2}\omega_{x_i x_i}\right) \tag{4.1}$$

Which can be rewritten as

$$\Pr(G \mid \omega, x) = \frac{1}{\prod_{i<j} Y_{ij}! \prod_i 2^{Y_{ii}/2} (Y_{ii}/2)!} \times \prod_{cd} \omega_{cd}^{m_{cd}/2} \exp\left(-\frac{1}{2} n_c n_d \omega_{cd}\right), \tag{4.2}$$

where n_c is the number of vertices in class c and

$$m_{cd} = \sum_{ij} Y_{ij} \mathbb{1}(x_i = c)\mathbb{1}(x_j = d)$$

Following a maximum likelihood approach, we want to maximize Eq.(4.2) with respect to ω and x, which is equivalent to maximize its logarithm, the log-likelihood, which is given by, neglecting terms that do not depend on ω or x:

$$\log \Pr(G \mid \omega, x) = \sum_{cd} \left(m_{cd} \log \omega_{cd} - n_c n_d \omega_{cd}\right) \tag{4.3}$$

By first order conditions, we get that the optimal ω is given by

$$\hat{\omega}_{cd} = \frac{m_{cd}}{n_c n_d} \tag{4.4}$$

Now, with the optimal ω, the log-likelihood (4.3) becomes

$$\log \Pr(G \mid \hat{\omega}, x) = \sum_{cd} m_{cd} \log \left(m_{cd}/n_c n_d\right) - 2m \tag{4.5}$$

where m is the total number of edges in the network, thus a constant that we can neglect, obtaining

$$\mathcal{L}(G \mid x) = \sum_{cd} m_{cd} \log \frac{m_{cd}}{n_c n_d} \tag{4.6}$$

which we will call the unnormalized log-likelihood for the group assignment x. This is the final objective we want to maximize. The larger the objective for a given assignment x, the more probable x is: the one that maximizes Eq.(4.6) will be the most likely one, what we want to find. Eq.(4.6) has an information-theoretic interpretation: by adding and dividing by constant factors $(m, n, 2)$ we can rewrite it as

$$\mathcal{L}(G \mid x) = \sum_{cd} \frac{m_{cd}}{2m} \log \frac{m_{cd}/2m}{n_c n_d/n^2} \tag{4.7}$$

Suppose now to pick randomly an edge from the graph. Let A be the random variable representing the class of one of the two endpoints (taken at random) and B the one related to the class of the other endpoint. Then, $p_K(c,d) \coloneqq \Pr(A = c, B = d) = m_{cd}/2m$. At the same time, if we were placing edges at random without taking care of the classes, as if we were considering a multigraph version of the Erdős–Rényi random graph, we would (a priori, before sampling the actual graph) get $p_1(c,d) \coloneqq \Pr(A = c, B = d) = n_c n_d / n^2 m$ Therefore we can rewrite Eq.(4.7) as

$$\mathcal{L}(G \mid x) = \sum_{cd} p_K(c,d) \log \frac{p_K(c,d)}{p_1(c,d)} \tag{4.8}$$

which is the Kullback-Leibler between $p_K(c,d)$ and $p_1(c,d)$. What we have obtained is the following: for the standard Stochastic block model, the most likely assignment x is the one that maximizes $\mathrm{KL}(p_K \parallel p_1)$, which can be thought as, loosely speaking, the assignment that maximizes the surprise compared to the so-called null model, which in this case is the Erdős–Rényi random graph. However, using the Erdős–Rényi random graph as a null model is problematic for real networks, because it produces Poisson-distributed networks, which are very unrealistic, as discussed in Chapter 1.

4.3 Degree-corrected SBM

Our aim is to incorporate the degree heterogeneity of nodes in our model. To do so, we define the degree-corrected stochastic block model as follows:

Definition 4.3.1 (Degree-corrected SBM)**.** *Consider the standard Stochastic block model defined before. The degree-corrected stochastic block model with parameters $\theta = (\theta_1, ..., \theta_n)$ is defined in the exact same way, except that for any pair of nodes i, j, with $i \leq j$, Y_{ij} is now given by a* Poisson$(\theta_i \theta_j \omega_{X_i X_j})$*. In particular, the parameter of the Poisson does not depend only on the classes of i and j, but also on i and j.*

With this definition the probability of a graph G given Θ, X and Ω becomes:

$$\Pr(G \mid \theta, \omega, x) = \prod_{(i,j):i<j} \frac{\left(\theta_i \theta_j \omega_{x_i x_j}\right)^{Y_{ij}}}{Y_{ij}!} \exp\left(-\theta_i \theta_j \omega_{x_i x_j}\right) \times \prod_i \frac{\left(\frac{1}{2}\theta_i^2 \omega_{x_i x_i}\right)^{Y_{ii}/2}}{(Y_{ii}/2)!} \exp\left(-\frac{1}{2}\theta_i^2 \omega_{x_i x_i}\right). \tag{4.9}$$

Since the θ parameters are arbitrary up to a multiplicative constant for each class, that would be absorbed into ω, we can impose for every class c: $\sum_{i:x_i=c} \theta_i = 1$. With this constraint we can simplify Eq.(4.9) and, with a similar argument as before and ignoring constant terms, one gets that the loglikelihood is given by:

$$\log \Pr(G \mid \theta, \omega, x) = 2 \sum_i k_i \log \theta_i + \sum_{cd} \left(m_{cd} \log \omega_{cd} - \omega_{cd}\right) \tag{4.10}$$

hence obtaining that it is maximized for

$$\hat{\theta}_i = \frac{k_i}{\kappa_{x_i}}, \quad \hat{\omega}_{cd} = m_{cd} \tag{4.11}$$

where k_i is the degree of node i and κ_c, which is called stub, is defined as the sum of the degree of vertices in class c. That is,

$$\kappa_c = \sum_d m_{cd} = \sum_{i:x_i=c} k_i. \tag{4.12}$$

Let us denote with $\langle . \rangle$ the average value over an ensemble of graphs with the same parameters. A remarkable property of this model is that it preserves both the expected number of edges between classes c and d :

$$\sum_{(i,j):x_i=c, x_j=d} \langle Y_{ij} \rangle = m_{cd} \qquad\qquad c, d \in \mathcal{C} \tag{4.13}$$

and the expected degree of each node i:

$$\sum_j \langle Y_{ij} \rangle = k_i \qquad\qquad i = 1, ..., n \tag{4.14}$$

In contrast, the standard SBM preserves only the expected number of edges between classes, since the average of the degree of each node will be the same among a class.
Back to the loglikelihood, substituting the optimal θ and ω it one can obtain:

$$\mathcal{L}(G \mid x) = \sum_{cd} m_{cd} \log \frac{m_{cd}}{\kappa_c \kappa_d} \tag{4.15}$$

Note that the only difference between this degree-corrected log-likelihood and the standard one, is that, instead of having the term $n_c n_d$ one has $\kappa_c \kappa_d$, that is, instead of having the number of members of each class, one has the stubs of those classes.
We can proceed in the same way as before with the information-theoretic interpretation: up to constant terms and multiplications by constants, we can rewrite the log-likelihood as

$$\mathcal{L}(G \mid x) = \sum_{cd} \frac{m_{cd}}{2m} \log \frac{m_{cd}/2m}{(\kappa_c/2m)\,(\kappa_d/2m)} \tag{4.16}$$

Now, consider a model in which we fix the expected degree of each node to be the degree of that node in the observed graph. Then, the joint p.m.f. of A, B (defined above) becomes

$$p_{degree}(c, d) \coloneqq \Pr(A = c, B = d) = \frac{\kappa_c}{2m} \frac{\kappa_d}{2m}. \tag{4.17}$$

Consequently, Eq.(4.16) can be written as $\mathrm{KL}(p_K \parallel p_{degree})$. As a result, loosely speaking, the most likely assignment x is the one that maximizes the surprise with respect to the null model used for p_{degree}, which is a model that incorporates the degree of each vertex.

4.4 The greedy algorithm

A useful property of the log-likelihood is that it is easy to compute its change when we modify the assignment of a single node i from class c to any class d. Let's define $a(z) \coloneqq 2z \log(z)$ and $b(z) \coloneqq z \log(z)$, with the convention $a(0) = b(0) = 0$. Then it is not difficult to check that we can write the change in the log-likelihood given by changing the class of i from (c,

d) to d as:

$$\begin{aligned}\Delta\mathcal{L} = &\sum_{t\neq c,d}\left[a\left(m_{ct}-k_{it}\right)-a\left(m_{ct}\right)+a\left(m_{dt}+k_{it}\right)-a\left(m_{dt}\right)\right]\\ &+a\left(m_{cd}+k_{ic}-k_{id}\right)-a\left(m_{cd}\right)\quad+b\left(m_{cc}-2\left(k_{ic}+u_i\right)\right)-b\left(m_{cc}\right)\\ &+b\left(m_{dd}+2\left(k_{id}+u_i\right)\right)-b\left(m_{dd}\right)-a\left(\kappa_c-k_i\right)+a\left(\kappa_c\right)-a\left(\kappa_d+k_i\right)+a\left(\kappa_d\right)\end{aligned} \tag{4.18}$$

where:

- k_{it} is the number of edges from vertex i to vertices in class t excluding self-edges;
- u_i is the number of self-edges of vertex i.

This computation can be done in $O(K + \langle k\rangle)$ and thus finding the d that maximizes the change can be done in $O(K(K + \langle k\rangle))$. Since those computations can be done quickly, it's possible to implement local vertex switching algorithms, such as Monte Carlo methods like the Metropolis-Hastings algorithm. However, those are slow to converge in general, and Karrer and Newman proposed instead a greedy algorithm which gives better results.
The algorithm goes as follows:

- initialize at random x
- Repeat until there is no increase in the objective function:
  - let the set of available nodes be the set of all nodes
  - repeat until you move every node:
    - select among all the available nodes and all possible classes d the move $i \leftarrow d$ (node i becomes of class d) that gives the largest increase in the objective (or least decrease, since it is not allowed to remain in the same class), obtained from Eq.(4.18) and apply it
    - remove the node i previously selected from the set of available nodes
  - Among all the configurations of x scanned throughout the inner loop, select the one with the highest log-likelihood, obtained from Eq.(4.15) and let x be equal to that one (that is, start the new loop with this configuration)

Algorithm 5 Greedy algorithm for degree corrected SBM

initialize at random x
repeat
    Available_nodes = $\{1, ..., n\}$
    while Available_nodes is non-empty **do**
        apply the move $i \leftarrow d$ that leads to the largest $\Delta\mathcal{L}$, with $i \in$ Available_nodes, $d \neq x_i$
        remove i from Available_nodes
    end while
    Let the new configuration x be the one with the highest objective among all the ones scanned throughout the inner loop
until there is no increase in the objective function
return x of the last but one iteration (the one with the largest objective)

Which can be summarized as follows:

The idea is that, at every iteration, we move every vertex at most exactly once, and among all the configurations scanned, we select the one with the highest score.

We note that the authors suggested running more simulations with different random seeds and taking the assignment x which gives the best score. This is a common fact/issue in deterministic algorithms which depend on a stochastic initial condition.

4.5 Simulations

Implementation of the Degree Corrected SBM can be found in DC_SBM.py. The class *GreedyAlgorithm* can be used to run the greedy algorithm on an adjacency matrix Y given a known number of classes k. To run the algorithm it is sufficient to use the function *infer(n_samples)*, where *n_samples* is the number of simulations to run (the best configuration is kept). The classes *Multigraph_SBM_noDC* and *Multigraph_SBM_DC* can be used to generate non-degree corrected and degree corrected SBMs respectively, without constraints. The classes *Separated_groups*, *Core_Periphery*, *Hierarchical* can be used to generate the models described later.

We test the greedy algorithm on synthetic data. How to generate them? Given a number of nodes n and a class assignment x (or equivalently a number of classes k with an associate probability for each class), the parameters of the (generative) DC-SBM are θ and ω. Suppose we want to fix the expected degree of each node, then we can generate a network with this constraint by setting for $i = 1, .., n \quad \theta_i$ from Eq.(4.11), where k_i is in this case the expected

degree of i and the stub is given by Eq.(4.12). For the choice of ω we have more freedom instead, provided that equations (4.11) and (4.12) are respected, which is equivalent to impose $\forall c \in \mathcal{C}, \sum_{d\in\mathcal{C}} \omega_{cd} = \kappa_c$. Following the approach of Karrer and Newman, we can let ω be a convex combination of two different structures, in this form:

$$\omega = \lambda\omega^{\text{planted}} + (1-\lambda)\omega^{\text{random}} \tag{4.19}$$

where ω^{random} represents a fully random network, so that $\omega_{cd}^{\text{random}}$ is the expected value of m_{cd} in a random graph preserving the degree sequence but without a block structure. $\omega_{cd}^{\text{random}}$ is thus given by $\frac{\kappa_c\kappa_d}{2m}$.
Instead, ω^{planted} is chosen to create the group structure. A possible example, which we call a separated groups structure, is:

$$\omega^{\text{planted}} = \begin{pmatrix} \kappa_1 & 0 & 0 & 0 \\ 0 & \kappa_2 & 0 & 0 \\ 0 & 0 & \kappa_3 & 0 \\ 0 & 0 & 0 & \kappa_4 \end{pmatrix}. \tag{4.20}$$

With this choice of ω^{planted}, when $\lambda = 1$ all edges are confined within communities, with no connections between different communities. Conversely, when $\lambda = 0$, edges are distributed randomly, maintaining the degree sequence. More complicated choices of ω^{planted} are also possible, as for instance a core-periphery structure (left) (with $\kappa_1 \geq \kappa_2$) or a hierarchical structure (right) (with $A \leq \kappa_1, \kappa_2$)

$$\omega^{\text{planted}} = \begin{pmatrix} \kappa_1 - \kappa_2 & \kappa_2 \\ \kappa_2 & 0 \end{pmatrix}, \omega^{\text{planted}} = \begin{pmatrix} \kappa_1 - A & A & 0 \\ A & \kappa_2 - A & 0 \\ 0 & 0 & \kappa_3 \end{pmatrix} \tag{4.21}$$

We can thus study how the Degree Corrected stochastic block model is able to recover those structures as λ varies and eventually compare it with the Standard Stochastic Block model. For instance, in figure 4.1 is displayed the average NMI as a function of λ for a Separated

Groups model with $n = 150, k = 2$ and expected degrees chosen randomly from $5, 10, 12, 20$. The curve is similar to the ones obtained by Karrer and Newman. Instead of considering

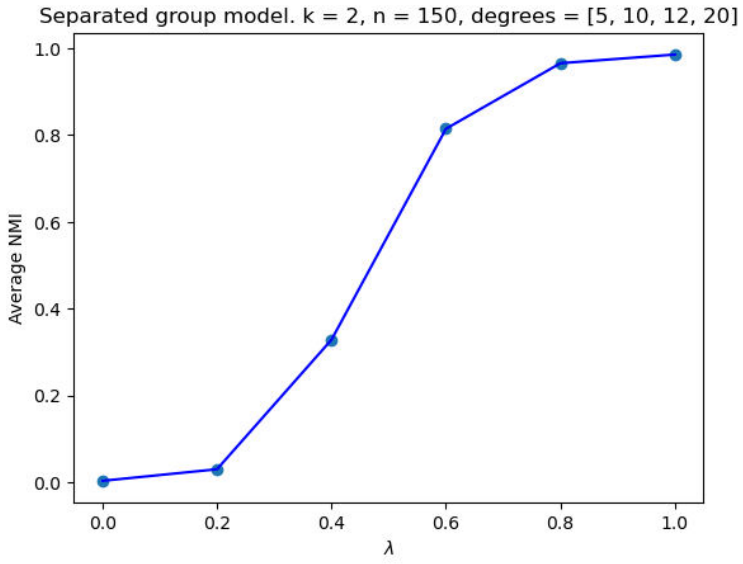
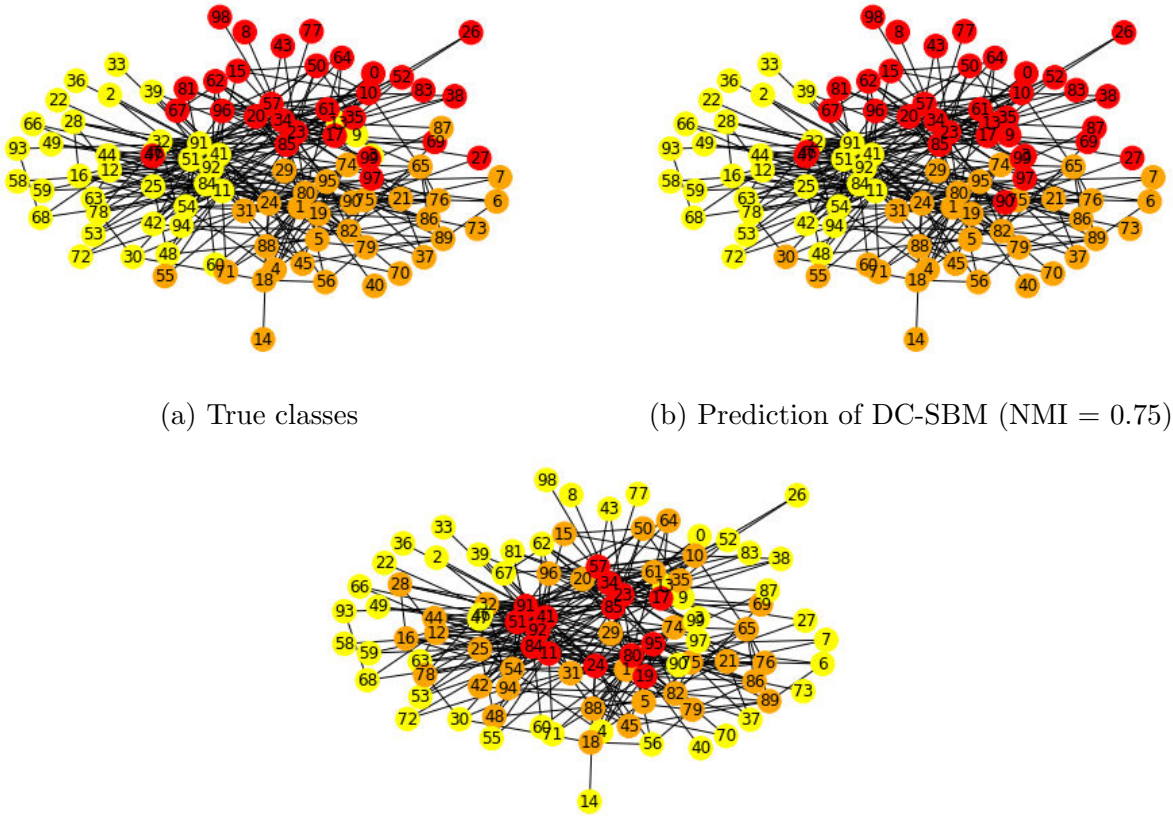


Figure 4.1

multigraphs, as discussed so far, it is possible to impose the same conditions required in Chapter 2 for binary graphs, setting to zero the diagonal elements of the adjacency matrix and cutting at 1 the values in Y greater than 1. Since we are in the sparse regime the correction is not relevant and with this new adjacency matrix, we can compare the results of the Degree Corrected SBM with the SBM defined in Chapter 2. For instance, figure 4.2 shows the results of the inference on a Separated Group model with $n = 100, k = 3, \lambda = 0.8$ and with a degree distribution following a power law with $\gamma = 2$. As expected, we see that the DC-SBM is able to find the true structure in such a network, while the standard SBM cannot: indeed it divides nodes only according to their degree.

As a final analysis we can study the performances of the DC-SBM on the three models described before as a function of λ, and compare them to the ones of the standard SBM. Results are shown in figure 4.3. For all of them $n = 100$, $n\_samples = 7$ and $M_0 = 300$. In the case of the Separated groups model the expected degree of each node was chosen randomly between 5 and 15 independently of the class, while in the Core Periphery model it was chosen between 5 and 20 if the node was in the core group, from 5 and 10 if it was in the periphery group. For the Hierarchical model, it was chosen from 5,10 and 20 independently of the class, and moreover, A was set to 60. Generally, the DC-SBM performs much better

than the standard SBM, however, the standard SBM is still able to reach relatively decent results, probably because the degree variation is moderated, in contrast with the scale-free network of figure 4.2.

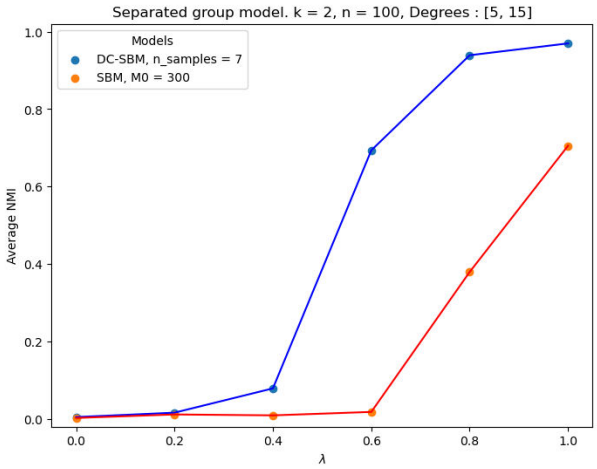


(a) True classes

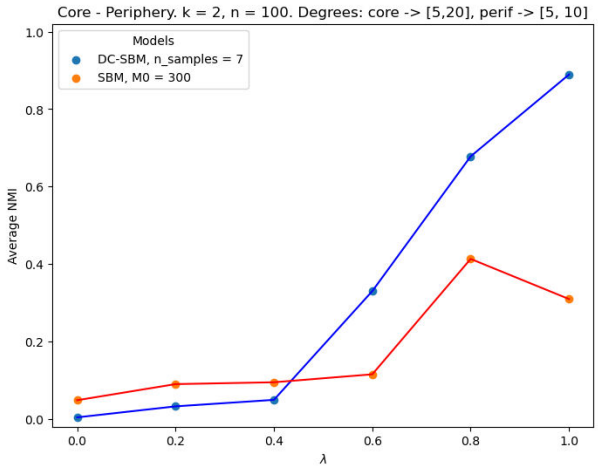
(b) Prediction of DC-SBM (NMI = 0.75)

(c) Prediction of standard SBM (NMI = 0.02)

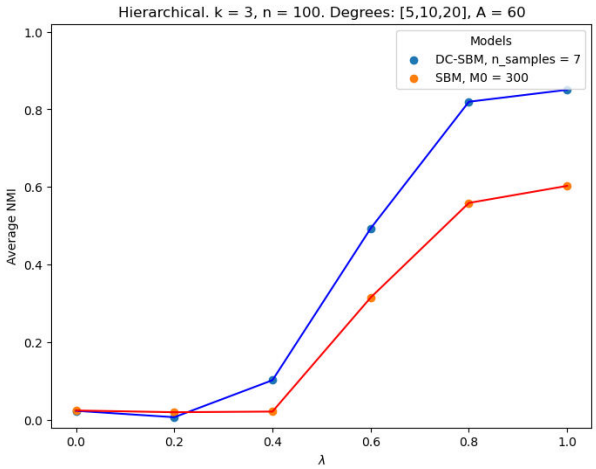
Figure 4.2: Scale free network, comparison of results



(a) Separated groups model



(b) Core-Periphery model



(c) Hierarchical model

Figure 4.3: Comparison of DC-SBM with the standard SBM in the three models described above

Bibliography

[1] P Erdös and A Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[2] Paul Erdos and Alfred Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.*, 5:17–61, 1960.

[3] Paul Erdos and Alfred Renyi. On the evolution of random graphs. *Bull. Inst. Internat. Statist.*, 38:343–347, 1961.

[4] Paul Erdos and Alfred Renyi. On the strength of connectedness of a random graph. *Acta Math. Acad. Sci. Hungar.*, 12:261–267, 1961.

[5] Stanley Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[6] Agata Fronczak, Piotr Fronczak, and Janusz A. Hołyst. Average path length in random networks. *Physical Review E*, 70(5), November 2004.

[7] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge University Press, Cambridge, 2016.

[8] Anna D Broido and Aaron Clauset. Scale-free networks are rare. *Nature Communications*, 10(1):1017, March 2019.

[9] D J Watts and S H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[10] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Diameter of the world-wide web. *nature*, 401(6749):130–131, 1999.

[11] Reuven Cohen and Shlomo Havlin. Scale-free networks are ultrasmall. *Phys. Rev. Lett.*, 90:058701, Feb 2003.

[12] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.

[13] Ulrike von Luxburg. A tutorial on spectral clustering, 2007.

[14] M Girvan and M E J Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. U. S. A.*, 99(12):7821–7826, June 2002.

[15] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), June 2004.

[16] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel MÃ$\frac{1}{4}$ller. Graph clustering with graph neural networks. *Journal of Machine Learning Research*, 24(127):1–21, 2023.

[17] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[18] Carolyn J Anderson, Stanley Wasserman, and Katherine Faust. Building stochastic blockmodels. *Social Networks*, 14(1):137–161, 1992. Special Issue on Blockmodels.

[19] Tom Snijders and Krzysztof Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14:75–100, 01 1997.

[20] Krzysztof Nowicki and Tom A B Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American statistical association*, 96(455):1077–1087, 2001.

[21] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(65):1981–2014, 2008.

[22] Wenjie fu, Le Song, and Eric Xing. Dynamic mixed membership blockmodel for evolving networks. page 42, 06 2009.

[23] Eric P. Xing, Wenjie Fu, and Le Song. A state-space mixed membership blockmodel for dynamic network tomography. *The Annals of Applied Statistics*, 4(2), June 2010.

[24] Xuhui Fan, Longbing Cao, and Richard Yi Da Xu. Dynamic infinite mixed-membership stochastic blockmodel, 2013.

[25] Wenzhe Li, Sungjin Ahn, and Max Welling. Scalable mcmc for mixed membership stochastic blockmodels, 2015.

[26] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011.

[27] Tiago P. Peixoto. Nonparametric bayesian inference of the microcanonical stochastic block model. *Physical Review E*, 95(1), January 2017.

[28] Junxian Geng, Anirban Bhattacharya, and Debdeep Pati. Probabilistic community detection with unknown number of communities. *Journal of the American Statistical Association*, 114(526):893–905, 2019.

[29] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[30] Jeffrey W. Miller and Matthew T. Harrison. Mixture models with a prior on the number of components, 2015.

[31] Steven Maceachern. Estimating normal means with a conjugate style dirichlet process prior. *Communications in Statistics-simulation and Computation - COMMUN STATIST-SIMULAT COMPUT*, 23:727–741, 01 1994.