Gibbs Sampling Estimation Method for Stochastic Block Models

Candidate Number 1024186

Mathematical Institute, University of Oxford, Radcliffe Observatory Quarter, Woodstock Road, Oxford OX2 6GG, UK

This manuscript was compiled on March 31, 2021

In this paper, we present a Markov chain Monte Carlo (MCMC) method for fitting a Stochastic Block Model (SBM) to a network with a previously assumed number of communities. The method consists of Gibbs sampling and computing the maximum likelihood estimators of the model parameters. We find that the method finds the correct fit for networks drawn from SBM universe when the number of classes is relatively small ($C \leq 4$) and generates reasonable fits on the real world data. We emphasize the advantages of being computationally efficient and address the drawbacks of limited applicability and correctness of the algorithm.

Community Detection | Stochastic Block Model

Network analysis has seen a great advance due to the increase of computer power available. In order to describe relational data, that is the type of data where we have interaction between data points, we use networks/graphs. Each data point is represented by a node which contains the information about the data point itself and edges, connecting two nodes, which contain the information about the interaction between the nodes.

Today, networks analysis is probably one of the most widely applied area of mathematics. It is used in physics to model complex systems (REF), social sciences to model interactions between people (REF) and even in marketing in order to properly display ads to the relevant customers (REF).

One of the largest problems in the field is the problem of community detection. It is often useful to partition the nodes into several groups of densely connected nodes within the groups and with fewer number of connections between nodes from different groups. These groups might have an interpretation such as grouping countries in a continent or they might be totally artificial. In any case, the partition could show interesting patterns that are not obvious to humans. Also, they can be used for making predictions about new data points based on their connections to the other nodes.

In the simplest setting, we could think of a network consisting of several disconnected parts. The partition of such a network can be done in complexity linear in the sum of the number of nodes and edges (1). The problem gets more complicated as we start connecting nodes from different groups. There are two main ways to formulate this problem: as an optimization problem (2) where we define a quality function and try to minimize/maximize it or as a statistical inference problem (3), (4).

In this paper, we take the later approach. Specifically, we study Stochastic Block Models (SBM) which is a class of randomly generated networks exhibiting a block-like structure. Stochastic Block Models are generated using internal parameters which describe the probabilities of nodes being in a certain group and connecting nodes from two groups. Our goal is that, given a new network, we find an SBM which best describes the data. That is finding the parameters which maximize the likelihood of seeing such a network. For simplicity, we assume that the number of groups is known. This is a rather strong assumption and there is a lot of research devoted to finding the optimal criteria for determining this number (5).

We mainly build on (3) which describes a Gibbs Sampling approach to finding the model parameters. This method has very nice properties of being scalable and efficient when dealing with larger networks where the optimization methods fail due to the large computational complexity. The shortcoming is that the method is not guaranteed to work because the process of finding the optimal parameters could possibly find a suboptimal solution. We identify and try to build intuition on why and when this happens in some simple cases. We first study our approach on networks drawn from SBM universe and later apply to Dolphin Social Network (6).

Notation

We define a *graph* as a tuple $G = (V, E)$. Where V is a set of vertices and E is a set of edges. The set of vertices is finite and we label $|V| = N$ and index the vertices from 1 to N. We can, then, describe an edge joining vertices i and j as (i, j). We can also add a value to the edge in which case the graph is *weighted* and the edge is described by three numbers (i, j, w) meaning that the edge joins vertices i and j assigning the value w to the connection. The graph is called *undirected* if for every edge (i, j, w) we also have an edge (j, i, w), otherwise it is called *directed*. We define the adjacency matrix to be a matrix A such that $A_{ij} = w$ if there is an edge (i, j, w) and 0 otherwise. In case the graph is undirected we have $A_{ij} = 1$ if there is a connection between i and j.

In our analysis we consider both directed and undirected weighted graphs with a restriction that the weights can only take values from a finite set.

A. Stochastic Block Model. A Stochastic Block Model is the simplest randomly generated network that has block (community) structure. It is widely studied in the literature and has many interesting properties (5), (7).

We consider a general weighted network where the weights take values from a finite set $\mathcal{W} = \{w_i\}_{i=1}^{W}$. We assume that

Significance Statement

In this paper, we present a method for fitting a network to a Stochastic Block Model with presumed number of classes. The method consists of Gibbs sampling and computing the maximum likelihood estimations of the model parameters.

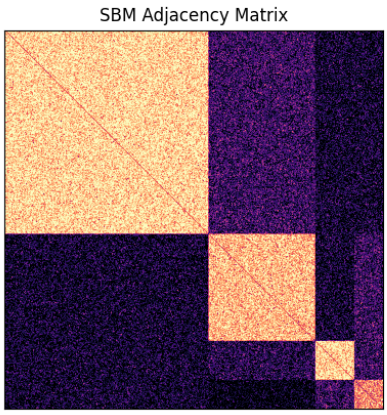

Fig. 1. Adjacency matrix of the SBM example.

there are C groups and we label them from 1 to C and define a *colour* of a node as the index of the group it belongs to. We define an SBM as such a network with a corresponding number C, parameter vector θ_i and a tensor η_{ijk}. θ_i is a probability that a randomly selected node corresponds to the i-th group where i takes values from 1 to C. η_{ijk} is the probability of having two nodes, one having a colour i and the other having a colour j, connected with an edge weighted by w_k. Here i and j run from 1 to C and k runs from 1 to W. In case we have an undirected graph we impose $\eta_{ijk} = \eta_{jik}$ and redefine the coefficient as being the probability of having edges (i, j, w_k) and (j, i, w_k) at the same time. Similarly, if the graph is unweighted $\mathcal{W} = \{0, 1\}$ and it is enough to define η_{ij}, omitting the index k, as a probability of connecting nodes i and j. We shall immediately note the following constraints:

$$\sum_{i=1}^{C} \theta_i = 1 \qquad [1]$$

and

$$\sum_{k=1}^{W} \eta_{ijk} = 1 \qquad [2]$$

First condition says that a node must belong to one of the groups and the second one saying that the value of the connection between two nodes must take a value from $\mathcal{W}$.

As an example for an unweighted, directed SBM with $N = 1000$, $C = 4$, $\vec{\theta} = (0.5, 0.3, 0.1, 0.1)$ and

$$\eta = \begin{pmatrix} 0.9 & 0.2 & 0.1 & 0.1 \\ 0.1 & 0.85 & 0.1 & 0.2 \\ 0.1 & 0.15 & 0.9 & 0.2 \\ 0.1 & 0.05 & 0.2 & 0.75 \end{pmatrix}$$

the adjacency matrix is display in Figure 1.

One should note that SBM looses it's block-like structure, if η is not defined properly. For example, if all η elements are the the same we effectively have only one group. Therefore, we restrict our attention to the η values that do exhibit a clear block structure as the one in Figure 1. In the literature, this is known as a recovery problem and the research produced a number of theorems and inequalities concerning the classification of nodes given the arbitrary values of parameters θ and η. (7)

1. Theory

In this section, we derive main formulae related to the probability distributions over the relevant values and parameters.

Let x_i be the colour of the node i. Our primary goal is to, given the adjacency matrix A, estimate the values x_i. In this case, x_i can be considered as a latent variable and there are Expectation-Maximization (EM) formulations of this problem. (4)

Note that, given θ and η, we can write a conditional distribution of $\{x_i\}_{i=1}^N$ and A as:

$$P(\{x_i\}_{i=1}^N, A|\theta, \eta) = (\prod_{i=1}^{C} \theta_i^{m_i}) \\ (\prod_{1\le i\ne j\le C} \prod_{1\le k\le W} \eta_{ijk}^{m_{ijk}}) \\ (\prod_{1\le i\le C} \prod_{1\le k\le W} \eta_{iik}^{m_{iik}}) \qquad [3]$$

Where m_i counts the number of nodes with colour i from given $\{x_i\}_{i=1}^N$ and m_{ijk} counts the number of edges between nodes from class i to j with the k-th weight from given A. Of course, in the undirected case the second product term should be modified to $i < j$. Using Bayes theorem we can write:

$$P(\eta, \theta|\{x_i\}_{i=1}^N, A) = \frac{P(\{x_i\}_{i=1}^N, A|\theta, \eta)P(\eta, \theta)}{P(\{x_i\}_{i=1}^N, A)} \qquad [4]$$

Where $P(\eta, \theta)$ is a prior probability distribution on these parameters and can be taken to be uniform, if we have no prior information. Here, we make a small modification compared to (3). Based on 4 we can write the maximum likelihood estimators as:

$$\hat{\theta}_c = \frac{\sum_{i=1}^N I\{x_i = c\}}{N} \qquad [5]$$

$$\hat{\eta}_{ijk} = \frac{m_{ijk}}{m_i m_j} \qquad [6]$$

for $i \ne j$ and:

$$\hat{\eta}_{iik} = \frac{m_{iik}}{m_i(m_i - 1)} \qquad [7]$$

Since the equation 3 is a multivariate distribution on the values of x_i it is very difficult to sample from it. Therefore, we adopt a classical Gibbs sampling approach (8). In order to write the probability distribution of a particular x_i given the values of $\{x_j\}_{i\ne j}$, we first define:

$$d(i, c, k) = \sum_{j=0}^{N} I\{A_{ij} = w_k\} I\{x_j = c\} \qquad [8]$$

This simply counts the number of connections of the node i to the nodes of colour j with an edge weighted by w_k. Then the distribution used for Gibbs sampling can be written as:

$$P(x_i = c|A, \theta, \eta, \{x_j\}_{i\neq j}) = Q\theta_c \prod_{1\leq k\leq c} \prod_{1\leq h\leq W} \eta_{ckh}^{d(i,k,h)} \quad [9]$$

Here Q is a normalization constant that can be easily computed.

2. Algorithm

Our algorithm is mainly based on the one proposed in (3). It is an iterative approach where we, in alternate steps, perform Gibbs sampling to find the colours of the vertices and compute the values of MLE for θ and η. The algorithm is summarised below:

Algorithm 1 Fitting SBM

1: Initialize $\{x_i^0\}_{i=1}^N$
2: **for** $p \leftarrow 1, M$ **do**
3: $\quad\theta^p \leftarrow \hat{\theta}(\{x_i^p\}_{i=1}^N, A)$
4: $\quad\eta^p \leftarrow \hat{\eta}(\{x_i^p\}_{i=1}^N, A)$
5: $\quad$**for** $i \leftarrow 1, N$ **do**
6: $\quad\quad$Sample x_i^p from $P(x_i|A, \theta^p, \eta^p, \{x_j^p, x_k^{p+1}\}_{j<i, i<k})$
7: **return** $\{x_i^M\}_{i=1}^N$, θ^M, η^M

There are a few points here that need to be discussed.

Firstly, we begin by randomly initializing the colours. However, one could also choose to initialize the parameters θ and η first. We see no trivial reasons why one approach should be chosen before the other in a general case. If one chooses to initialize θ and η first, steps 3 and 4 should be put below steps 5 and 6.

Secondly, it is important to emphasise that the authors of (3) did not use MLE for estimating the parameters. In each step, they sample the parameters from 4. We, empirically, find that using MLE works just as well and for the sake of simplicity and efficiency we decided to stick with that. It would be interesting to study the implications that using MLE instead of sampling has on the convergence properties.

Finally, we have not specified what is M. That is the number of iterations that we want to perform. Typically this is $O(1)$ and we will discuss it once more in the result section below.

A. Computational Complexity. Here, we address the computational complexity of all the steps. To compute θ^p we need to compute m_i which is $O(N)$. To compute η^p we need to compute m_{ijk} for which we need to visit all the edges which is $O(|E|)$ or in case of a dense network $O(N^2)$. The final step is Gibbs sampling where we first have to compute $d(i, c, k)$, here we need to visit all the nodes which is $O(N)$. After that, we need to compute the terms in equation 9 which takes $O(CW)$ for each class. To compute the normalization constant we have to repeat that for every class which makes the complexity $O(C^2W)$. In the end, we have to repeat this for every node so in total the complexity of one iteration of Gibbs sampling is $O(MN^2 + MNC^2W)$ in the worst case where $|E| \sim N^2$. If we assume that $M = O(1)$ and that $N >> C, W$, the total computational complexity is $O(N^2)$. This seems to be a great achievement compared to the modularity based methods (9) that run in the expected $O(n^2 log(n))$ and the worst $O(n^3)$ times. However, this comes at a certain cost of the algorithm not always producing the correct solution which we discuss in the following sections.

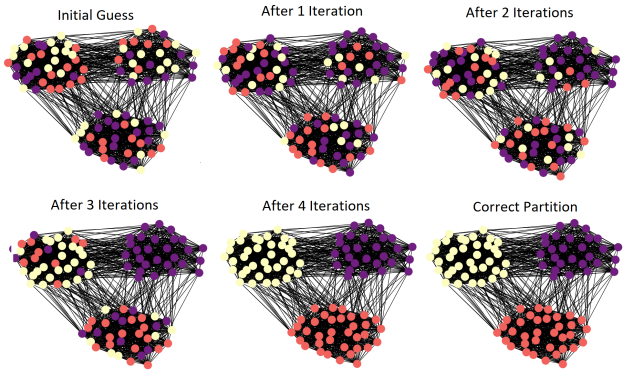

Fig. 2. A Successful Run.

B. Where the Algorithm Breaks Down. Our algorithm has a few weak points. The obvious ones are equations 5 and 6 which do not work anymore when one of m_i equals 0 or 1. We call these type 1 and type 2 errors. Obviously, this goes towards an undesired situation where the initial number of colours shrink by 1.

To better understand this consider an undirected, unweighted graph only two colours and where $\eta_{11} = \eta_{22} = 0.9$ and $\eta_{12} = \eta_{21} = 0.1$. Suppose that, at some point, a certain node of colour 2 in connected only to nodes of colours 1. In that case, flipping the node's colour is favourable because it increases the likelihood of such a situation by a factor $\frac{\eta_{11}}{\eta_{12}}$ raised to the power of how many connections the node has. Furthermore, the node need not to be connected only the nodes of different colours. It is enough that the overall likelihood goes in the favour of flipping for it to be likely to flip its colours even when it should not. Luckily, this only depends on the initialization and running the algorithm again will most certainly fix this issue.

Less often, the algorithm could find a sub-optimal solution where no flipping is favourable. This is partially suppressed due to the fact that we sample x_i^p values. However, we have seen examples where even with the sampling, the algorithm gets trapped between two or three sub-optimal partitions.

3. Results

In this section we present the results of the experiments we have performed. We mainly focused on testing our algorithm on the simulated data in order to better understand its properties. Finally, we give an example how to algorithm could be applied to the real Dolphin Social Network.

A. Experiments on Synthetic SBMs. We have performed a large number of experiments across wide ranges of parameters. For simplicity we focused on undirected and unweighted networks. In Figure 2, we show what a successful algorithm run looks like. In just 4 iterations the algorithm is able to correctly classify the nodes. In Figures 3 and 4 we see examples of type 1 and 2 errors.

The displayed experiments were all run with $N = 100$, $C = 3$, $\theta = (0.5, 0.3, 0.2)$ and

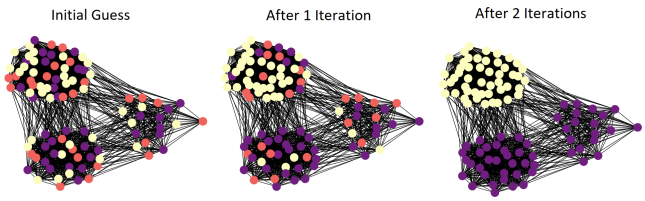

Fig. 3. Type 1 Error.

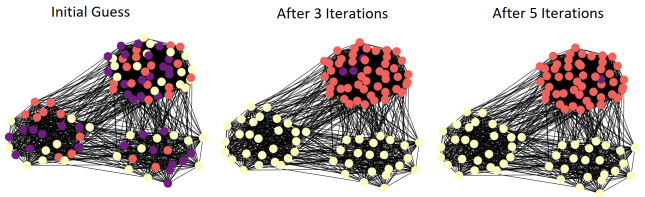

Fig. 4. Type 2 Error.

$$\eta = \begin{pmatrix} 0.9 & 0.2 & 0.1 & 0.1 \\ 0.1 & 0.85 & 0.1 & 0.2 \\ 0.1 & 0.15 & 0.9 & 0.2 \\ 0.1 & 0.05 & 0.2 & 0.75 \end{pmatrix}$$

The algorithm performs nearly always perfectly when $C = 2$. When $C = 3$ it is not rare that it needs to be run more than once. For $C > 4$ the algorithm almost always fails. This happens because the probability of seeing type 1 or type 2 errors is very large due to the larger number of clusters. Therefore, it remains an open question how should one proceed in these cases.

We tested the computational complexities as a function of number of nodes. We empirically validated that the worst case cost per iteration is bounded by $O(N^2)$. The tables 1 and 2 show this in cases when $C = 2$ and $C = 3$.

N	t[s]	$(\frac{N}{N_1})^2$	$\frac{t}{t_1}$
100	0.10	1	1.0
300	0.83	9	8.3
500	2.35	25	23.5
700	4.63	49	46.3
1000	9.12	100	91.2

Table 1. Computation cost per iteration when $C = 2$

N	t[s]	$(\frac{N}{N_1})^2$	$\frac{t}{t_1}$
100	0.38	1	1.0
300	1.95	9	5.1
500	5.73	25	15.1
700	10.0	49	26.3
1000	20.7	100	54.5

Table 2. Computation cost per iteration when $C = 3$

Finally, we would like to make a comment on the ability of finding the correct solution depending on the model parameters. The algorithm works the best when there is a clear difference between diagonal and off-diagonal elements of η. Also, in cases when two colours have similar probabilities of attaching to all different colour groups the algorithm may confuse them and errors of type 1 and 2 are likely to occur. Note that it is not necessary for the model that the diagonal elements are much larger than the off-diagonal ones. The algorithm works fine even when the situation is completely opposite. Similarly if some entries of θ are much smaller compared to the others, those classes are likely to vanish through type 1 and 2 errors.

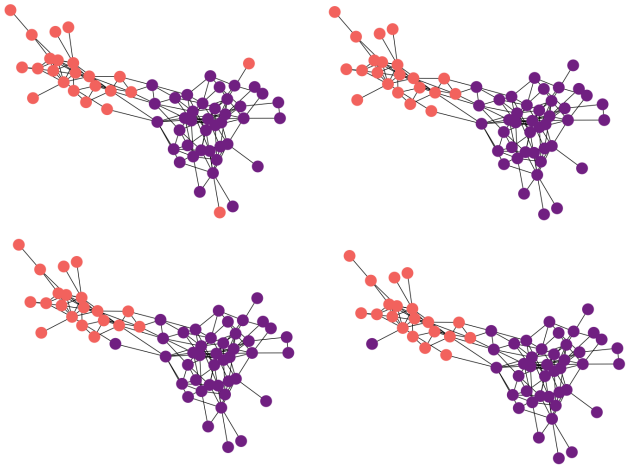

Fig. 5. 4 Consecutive Iterations on Dolphin Social Network Near the Optimal Partition.

B. Dolphin Social Network. We choose to test our algorithm on a rather simple and small real network. There are many problems that arise once we depart from the SBMs and start applying the algorithm to the real world data. Therefore, we show all the inconveniences that occur on this example.

The first question is how to choose the number of colours. As we did not want to focus on this, rather difficult question, we took advantage of the fact that we can visualize this network and roughly estimate that the number of classes should be 2. However, there are criteria developed that could solve this issue in more complex cases (5).

Another drawback of the algorithm is the fact that it takes more iterations to converge when the data is not as ideal as it is in the case of SBMs. It took the algorithm around 30 iterations to reach what we think is the optimal partition of this network. These was to be expected as the existence of such simple probabilistic dynamics postulated by SBM is a very strong assumption.

Finally, there is a question of convergence. In case of the real networks we do not have predefined values of colours. Therefore, we do not know when the algorithm is done. In fact, the algorithm is not likely to stop changing the partition even when it has already found the correct one. This is well illustrated in Figure 5. There is always a non-zero probability that a node within one colour group will flip its colour. To overcome this issue, we propose that the algorithm is run for a number of iterations and that it computes either likelihood of a partition or a value of some quality function that measures how good the partition is. Once we see that our measure does not change significantly or oscillates around some value, we simply pick the best possible one the algorithm has computed so far.

4. Conclusion

To conclude, we have presented an efficient Gibbs sampling (MCMC) method for fitting an SBM to a given network. Our algorithm is mainly based on (3) with a modification that it uses MLEs of parameters rather than sampling them from complicated posterior distributions. We explored the efficiency and common problems of this method. We conducted experiments on both simulated SBM networks and real Dolphin Social Network. We show, both theoretically and empirically, that the worst-case cost per iteration is $O(N^2)$ and that the algorithm completes the task in $O(1)$ iterations. This is seemingly better than that of modularity based methods $O(N^3)$, but comes at a price that it is applicable only when the number of classes is small $C \leq 4$. We illustrated the problems of choosing the number of classes, increased computational cost and the indefiniteness of convergence when dealing with real data on Dolphin Social Network. The subject of the further research could be: a more rigorous analysis of the convergence and the extension to handling the larger number of classes.

Materials and Methods

The entire code is written in Python using standard Python packages. SBM generator as well as all functions are written from scratch apart from visualization and loading of .gml files for which we used NetworkX package. Dolphin Social Network data was retrieved from this page.

1. (2021) Retrieved from https://en.wikipedia.org/wiki/connected-component_labeling.
2. Reichardt J, White DR (2007) Role models for complex networks. *The European Physical Journal B*.
3. Nowicki K, Snijders TAB (2007) Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*.
4. Nowicki K, Snijders TAB (1997) Estimation and prediction for stochastic block models for graphs with latent block structure. *Journal of Classification*.
5. Lee C, Wilkinson DJ (2019) A review of stochastic block models and extensions for graph clustering. *Applied Network Science*.
6. (2021) Retrieved from http://www-personal.umich.edu/mejn/netdata/.
7. Abbe E (2017) Community detection and stochastic block models: Recent developments. *Information Theory Newsletter*.
8. Lynch SM (2007) Introduction to applied bayesian statistics and estimation for social scientists.
9. Vinícius da Fonseca Vieira, Carolina Ribeiro Xavier NFFE, Evsukoff AG (2014) Performance evaluation of modularity based community detection algorithms in large scale networks. *Mathematical Problems in Engineering*.