Community Detection on Stochastic Block Model

Markov chains and Algorithmic Applications

Francesco Borg, Roberto Ceraolo, Arturo Cerasi
School of Computer and Communication Sciences, EPFL Lausanne, Switzerland

***Abstract*—Given a Stochastic Block model with two communities, the goal of this project is to test the performances of various algorithms, based on Markov Chain Monte Carlo simulation, in identifying the two communities. Several simulations using Metropolis-Hasting, Houdayer and a Mixed Metropolis-Houdayer algorithms have been run. For suitable values of the parameters, all the algorithms were able to identify the two communities, with different convergence rates. We observed two main insights: the convergence rate strongly depends on the number of nodes of the network (Houdayer performs the best in large networks settings). And the ability to correctly categorize the nodes into the two communities depends on the ratio of the probabilities of inter-cluster over intra-cluster connection, with a threshold point, above which the algorithms struggle to converge.**

I. Introduction

The Stochastic Block Model is a rather simple model used to describe a specific type of graphs. In particular it characterizes those in which two or more communities are present, where two members of the same community are more likely to be connected rather than two members of different communities. It can be thought of as two high-school classes and their Facebook network: we have two communities (the two classes), and each member of such classes is more likely to be friend with a member of his own community rather than being friend to a person from the other class. Our goal here is, given a Stochastic Block Model in which only two communities are present, to test different algorithms on the task of identifying correctly the nodes belonging to the two communities, by only observing their connections. More in detail, the three algorithms compared are: the Metropolis-Hastings algorithm, the Houdayer algorithm, and a mixed version of the two. From a high level perspective, these algorithms allow us to sample from a certain probability distribution. In the community detection problem, given a ground truth (an assignment of each node to a community), after a large number of steps on a Markov Chain, the aim is to sample from a distribution on nodes which closely approximate the original ground truth. Denoting the ground truth by $\boldsymbol{x}^* = \{\pm 1\}^N$ and a proposed estimate $\hat{\boldsymbol{x}}$, the validity of a solution is assessed calculating the overlap between the two, which is given by:

$$q_n = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{x}_i x_i^*\right|$$

More specifically, we are interested in assessing the average overlap for our algorithms over prior distribution on $\boldsymbol{x}^*$ and over the random graph G:

$$\bar{q}_n = \mathbb{E}_{\boldsymbol{x}^*}[\mathbb{E}_{\{e_{ij}\}|\boldsymbol{x}^*}[q_N]]$$

Where e_{ij} is the edge connecting node i and j. We will follow the Bayesian approach and therefore our estimate will be the posterior mean. Hence using Bayes rule the posterior distribution, which in practice corresponds to the distribution $\boldsymbol{\pi}$ we try to sample from with the MCMC algorithms, will be:

$$\mathbb{P}(\boldsymbol{x}|\{e_{ij}\}) = \frac{\mathbb{P}(\{e_{ij}\}|\boldsymbol{x})\mathbb{P}(\boldsymbol{x})}{\sum_{\boldsymbol{x}}\mathbb{P}(\{e_{ij}\}|\boldsymbol{x})\mathbb{P}(\boldsymbol{x})}$$

Which after some simplification can be rewritten as:

$$\mathbb{P}(\boldsymbol{x}|\{e_{ij}\}) \propto \frac{\Pi_{1\le i<j\le N} e^{h_{ij}x_i x_j}}{\sum_{\boldsymbol{x}\in\{\pm1\}^N}\Pi_{1\le i<j\le N} e^{h_{ij}x_i x_j}} \qquad (1)$$

where

$$h_{ij} = \frac{1}{2}\left[e_{ij}\ln\frac{a}{b} + (1-e_{ij})\ln\frac{1-\frac{a}{N}}{1-\frac{b}{N}}\right]$$

Furthermore, it can be shown that a Bayesian estimator is the best possible estimator. We now introduce some notation to improve clarity, in the following sections we will refer to:

1) N as the number of nodes in the observed graph;
2) a, $b > 0$, with $a < b$ when divided by N they represent respectively the probability to have an edge that connects to another node in the same cluster ($\frac{a}{N}$) and to a node in the other cluster ($\frac{b}{N}$).

So that $\mathbb{P}(e_{ij} = 1 | x_i^* x_j^* = +1) = \frac{a}{N}$ and $\mathbb{P}(e_{ij} = 1 | x_i^* x_j^* = -1) = \frac{b}{N}$;

3) $d = \frac{(a+b)}{2}$ the average degree of the graph;
4) $r = \frac{b}{a}$ as the ratio between the two probabilities, a key parameter for the experiments

II. Algorithms

A. Metropolis-Hasting

The first algorithm that will be used is the Metropolis-Hasting algorithm. Starting by assigning each node to one of the two community ($x_i \in \{-1, +1\}, \forall i \in S$), the base chain ψ will consists of picking a node uniformly at random and flipping its value (i.e. changing the community assignment). From the Metropolis-Hasting definition it follows that the acceptance probability for each move will be $a_{ij} = \min(1, \frac{\pi_j}{\pi_i})$, where $\boldsymbol{\pi}$ in this case represents the posterior distribution. From (1) we can see that the computation of the acceptance probability reduces to calculating the ratio between two products of exponentials. In order to reduce the computational complexity of the algorithm (without incurring in any loss), two major simplifications were implemented. First of all, the denumerators of the two distributions are the same, as they are the summations over all possible values of $\mathbf{x}$ of the products of the same exponentials (see (1)) so when computing the ratio they cancel out. Moreover, there are also many factors that cancel out between the two numerators of π_j and π_i. Since at the numerator we use the proposed chain after one flip and at the denumerator we use the current state of the chain, the factors that differ are the ones involving only the flipped index. Let i be the proposed move, $\boldsymbol{x}^{(t)}$ the current chain, $\boldsymbol{x}^{(t+1)}$ the chain with the flipping applied that we need to decide whether to accept or reject. In formulae:

$$\frac{\Pi_{1 \le i < j \le N} e^{h_{ij} x_i^{(t+1)} x_j^{(t+1)}}}{\Pi_{1 \le i < j \le N} e^{h_{ij} x_i^{(t)} x_j^{(t)}}} = \frac{e^{\sum_{i \neq j} h_{ij} x_i^{(t+1)} x_j^{(t+1)}}}{e^{\sum_{i \neq j} h_{ij} x_i^{(t)} x_j^{(t)}}} \quad (2)$$

After several steps of the chain, the obtained distribution should be close to the graph's ground truth.

B. Houdayer

The second algorithm used will be the Houdayer algorithm. After having randomly initialized two parallel configurations $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$, we define the *local overlap* to be $y_i = x_i^{(1)} x_i^{(2)}$, computed at every node i. Thus defining the clusters on the observed graph, which corresponds to the connected sets of nodes having the same overlap value. A node i for which $y_i = -1$ is picked uniformly at random. Considering the subgraph of the observed graph in which all the nodes have label $y_i = -1$, we flip the value of each node connected to the one chosen (i.e. in the same cluster) in both configurations $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$. Then a single Metropolis step is performed on each configuration independently. After an arbitrary number of steps the overlap between each of the two spin configurations and the ground truth is computed and then averaged.

C. The Mixed Metropolis-Houdayer algorithm

The third and last algorithm tested is a mixed Metropolis-Houdayer algorithm. Here simply the Houdayer step is performed every n_0 steps, instead of on every step.

III. Theoretical considerations

There are two theoretical elements worth considering before moving on to experiments. Firstly, finding the theoretical value for the ratio $\frac{a}{b}$ at which the phase transition occurs. Secondly, understanding whether a process defined through the Houdayer move (and that move only) is an ergodic Markov Chain or not.

A. Phase Transition

When considering the Stochastic Block Model, one can show that communities cannot be detected if the values for a and b follow this inequality:

$$(a-b)^2 \ \le \ 2(a+b) \quad (3)$$

Hence one can expect that there are some critical values of b and a, which can be proxied by their ratio r_c, past which, it is possible to observe a drastic (how drastic depends on N and t) change in the overlap value. In order to find the critical value r_c, we consider the associated equation to 3 and we write it in terms of $r_c = f(d)$. Finding the values of a and b as functions of d and r yields:

$$a = \frac{2d}{r+1} \quad b = \frac{2dr}{r+1}$$

Substituting in the equation associated to 3 we get:

$$(\frac{2d - dr}{r+1})^2 \ \le \ 2(\frac{2d - dr}{r+1})$$

From which we arrive to our result, valid for $d > 1$:

$$\frac{d+1}{d-1} - 2\sqrt{\frac{d}{(d-1)^2}} \ \le \ r \le \ \frac{d+1}{d-1} + 2\sqrt{\frac{d}{(d-1)^2}}$$

It can be easily seen that the rightmost term will be greater than 1 for any $d > 1$. If we consider the cases in which $d > 1$, we are left with the left inequality, as r will never be greater than 1, since we are in the assortative case, i.e. $b < a$. Evaluating the critical value of r_c for instance, when $d = 3$, we get: $r \geq 2 - \sqrt{3} = 0.268$

B. Ergodicity of Houdayer

Let's consider the chain given by the Houdayer move, without the Metropolis flip. We claim that such chain is not ergodic. As explained above, at each epoch we switch all the nodes connected in a certain cluster on which the two chains "disagree" (i.e. the local overlap $y_i = -1$). Given any node on which the two chains have different label values at the beginning, if it gets flipped, it will happen in both chains, so the two chains will disagree again. Hence, if at the beginning there is at least one point on which the two chains disagree, then the chain will never stay the same in two consequent epochs, as at least the index corresponding to such node will be flipped. Hence, there cannot be self loops. This hints at a possible absence of aperiodicty. This fact, per se, does not completely exclude ergodicity, as it could theoretically be aperiodic even without self loops. Nevertheless, the chain is periodic with period $d = 2$. First of all, we notice that the local overlap array y is constant, since nodes on which the two spins disagree are flipped simultaneously, and so the two spins will always disagree on those. Secondly, every time we choose a cluster we flip the nodes corresponding to that cluster. The only way the chain can go back to the initial state is by choosing that cluster again and flipping it all, hence the period is 2.

There are two corner cases worth mentioning: when the y chain is full of ones (the two chains agree on everything) and when it is full of -1s (the two chains are different in every node).

1) When $y_i = 1$, for every i, the Houdayer move as defined, does not change anything, so the algorithm cannot run. Fortunately, the chain cannot reach that condition by itself, as the move will always switch both spins simultaneously, so where $y_i = -1$, it will remain so. We assume the situation in which $y_i = 1$, for every i is not a possible initial configuration, otherwise this simplified version of the Houdayer algorithm cannot work (it would always stay in the same state).
2) When $y_i = -1$ for every i, at every epoch the Houdayer move changes completely both spin configurations, so it alternates just 2 possible overall configurations, one in which one spin is composed of only 1s and the other only -1s and the other possible configuration is vice versa. So it is still periodic (we assume this cannot be an initial configuration).

Finally, one last consideration: if we considered the Houdayer case together with the Metropolis step on the two chains, it could happen that through the metropolis flipping the two spin configurations become equal, hence the y chain gets only ones, and so in this case the Houdayer flip would not happen. At that point if the Metropolis step rejects the moves on both configurations, everything stays the same, we get a self- loop, gaining aperiodicy and hence ergodicity.

IV. Results

A. Studying Convergence as a function of time

We want to start the experiments by assessing which are the different convergence times of the three proposed algorithms. In order to do so we define some fixed parameters for the three algorithms to yield comparable results. In particular we set the number of nodes $N = 100$, the number of steps for which every algorithm is trained $t = 10000$, average degree of the graph $d = 3$ and $r = 0.0169$ (following from $a = 5.9$ and $b = 0.1$).

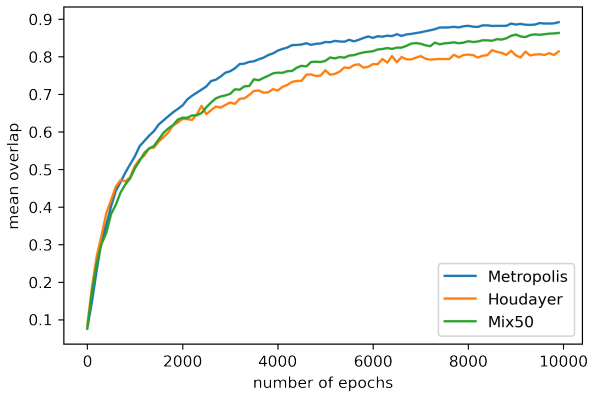

Fig. 1. Avg convergence times in graphs with N=100

From the plotted graph (displaying average results over 100 runs of the algorithms) we can clearly see how the Metropolis algorithm is the one with the fastest convergence for such parameters: with a small number of epochs it is difficult to make a clear distinction between the three as they all poorly predict the two communities. However, as the number of epochs increases we can see how the mean overlap yielded by the Metropolis-Hasting is quicker to reach high values. Although, this does not imply that with different parameters configurations (for

example with a very high N) we would obtain the same results. Indeed we have been able to run a simulation on a larger amount of nodes ($N = 1000$) for a greater amount of epochs ($t = 50000$), still keeping $r = 0.0169$ but averaging the results of just 10 simulations due to the high computational complexity.

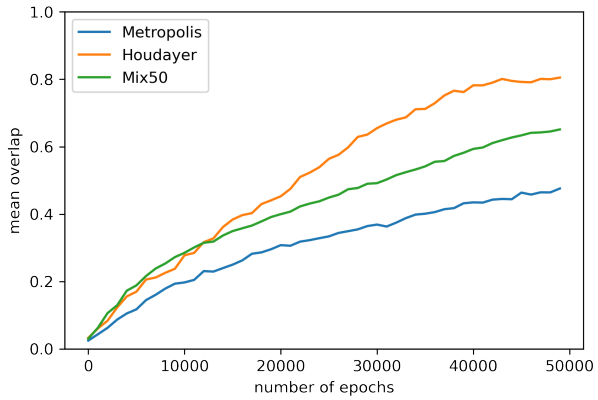

Fig. 2. Avg convergence times in graphs with N=1000

The results confirm what we could have expected: the advantage of cluster moves contained in the Houdayer algorithm begins to make a difference for large size graphs, we can clearly see an inverted trend with respect to the previous experiment where only 100 nodes were used. We can conclude that the performance in terms of convergence time strongly depends on the number on nodes of the graph. For a smaller graph, made by relatively few nodes, the Metropolis-Hasting algorithm provide an adequate solution in relatively few iterations. Instead, as the number of nodes increases, the Metropolis algorithm, for a fixed number of iterations will detect the two communities, less precisely. In this scenatrio, Houdayer algorithm is the fastest to converge.

B. Limiting Performance as a function of r

In this section we present the experiments we performed to verify whether there are differences in the overlap performance of the various algorithm, for various values of $r \in (0, 1]$, holding the parameters fixed. In particular, we compared the three algorithms, choosing $n_0 = 50$ for the Mixed one. In Figure 3 below we present the average overlap results of the three algorithms, over 10 runs (we had to limit the number of runs due to computational complexity), made of 10000 epochs, for 50 different values of $r \in (0, 1]$:

As we expected the performance of all three algorithms are extremely comparable. The difference in terms of performance between the three algorithms should

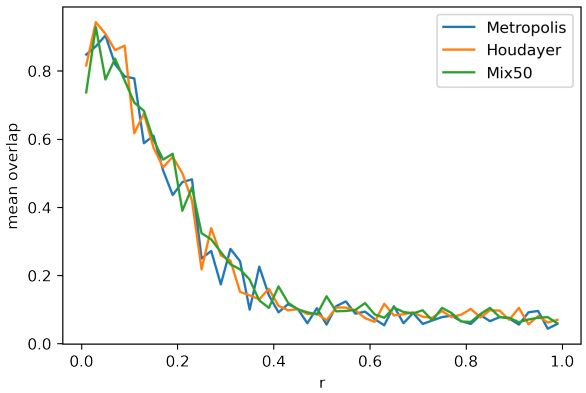

Fig. 3. Avg overlap for different values of r, over 10 runs

only be perceivable for large sizes of the graph. Indeed, here for N fixed to 100 due to the computing power requirements the are no noticeable differences, instead we notice a rapid increase of performance for r below the critical value estimated in the section III-A.

In addition, we also ran the same experiment for some selected values of r, and computed the average over 100 runs. We can notice from the plot below that in general there are no big differences between the average final overlaps of the different algorithms. This graph also confirms the slightly faster convergence time of the Metropolis algorithm for $a = 5.9$ and $b = 0.1$ and $N = 100$ (corresponding to $r = 0.0169$) noticed in section IV-A.

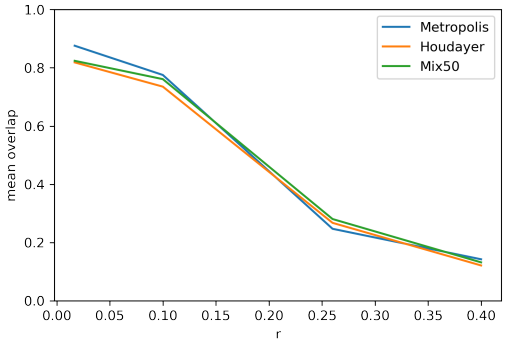

Fig. 4. Caption

C. Phase Transition for various values of N

We ran various experiments to test the hypothesis made about the critical threshold of values r_c for various values of N. Our observations seem to confirm the hypothesis found in theory about the phase transition around the critical value of r_c. Given that the value of r_c is independent of the optimization algorithm used

(as seen in Section IV-B) and taken into account the high computational cost of the Houdayer (and mixed) algorithm we decided to study the phase transition for networks of various sizes (N) exclusively using Metropolis. In particular here below we plot the average results obtained across 10 separate runs, of Metropolis algorithm applied on 10000 epochs, to networks of sizes $[30, 100, 500, 1000]$ for a range of $r \in (0, 1]$

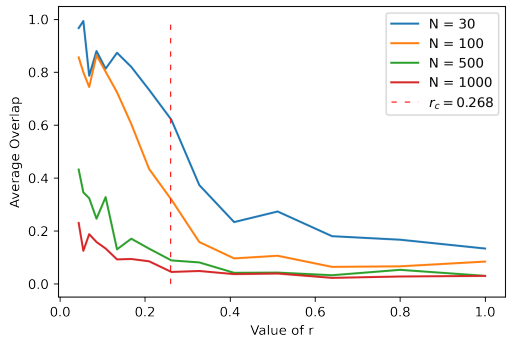

Fig. 5. Caption

We can notice that the average overlap for all values of N start to significantly increase around the theorical critical value $r_c = 0.268$. Clearly the number of epochs used in our setting (chosen to be constant to make results comparable) is not sufficient to visualize the full transition phase on larger size networks, as they would require a much higher (and computationally expensive) number of epochs to reach good overlap values.

V. Conclusions

Finally, from our simulations we can conclude that:

1) the theoretical value of the transition phase, r_c, that we calculated can be said to be reflected empirically from the experiments, even without being in an ideal setting with $N \to \infty$ and $t \to \infty$, we can notice a substantial drop of average overlap value around r_c;
2) moreover, the convergence time of the algorithms is strongly influenced by the number of nodes present in the graph, for values of N in the order of 100 we found that the Metropolis Algorithm is the fastest to converge, instead, for values of N around 1000, the Houdayer algorithm appears to be the best one.