Mini-project: Application of Monte-Carlo methods to community detection

This mini-project aims to apply variants of the Metropolis-Hasting algorithm to one of the simplest models of the community detection problem and to compare the performance and properties of these methods.

1 Definition of the Stochastic Block Model (SBM) model

To begin with an example, consider a portion of the Facebook network corresponding to students in a high school, with edges corresponding to users who are friends. Knowing the graph of connections, the goal is to recover a division of students corresponding to high-school classes. The signal is derived from the assumption that students in the same class are more likely to be friends and thus connected than students from different classes. The Stochastic Block Model, which will be introduced, is a widely studied simple model for such a situation. We only consider two classes here, but the model can be generalized for more classes. We follow standard terminology in what follows: classes are called "communities", and the Facebook network or graph of connections is called the "observation graph".

First, we explain how a random "observation graph" G is generated. G has N nodes $i \in \{1, \dots, N\}$, each belonging to one of the two communities. The variable denoting to which community node i belongs is $x_i^* \in \{-1, +1\}$. A vector $\boldsymbol{x}^* \in \{\pm 1\}^N$ is generated with i.i.d. elements uniformly distributed in $\{\pm 1\}$: $\mathbb{P}(x_i^* = +1) = \mathbb{P}(x_i^* = -1) = \frac{1}{2}$ for all $i \in \{1, \dots, N\}$. For N large enough, we have $\frac{1}{N}\sum_{i=1}^N x_i^* \approx 0$, so the two communities have (roughy) the same number of nodes. The vector $\boldsymbol{x}^* \in \{\pm 1\}^N$ is latent (in other words, not observed or hidden). An edge between two nodes i, j in the graph is represented by the variable $e_{ij} \in 0, 1$, where $e_{ij} = 1$ means that this edge exists in the graph. The edges are generated independently as follows:

$$\begin{aligned} \mathbb{P}(e_{ij} = 1 | x_i^* x_j^* = +1) = \frac{a}{N} = 1 - \mathbb{P}(e_{ij} = 0 | x_i^* x_j^* = +1) \quad \forall i < j \in \{1, \dots, N\} \\ \mathbb{P}(e_{ij} = 1 | x_i^* x_j^* = -1) = \frac{b}{N} = 1 - \mathbb{P}(e_{ij} = 0 | x_i^* x_j^* = -1) \quad \forall i < j \in \{1, \dots, N\} \end{aligned} \tag{1}$$

where a, b are two $O(1)$ positive numbers. We assume that $a > b$. This is called the assortative case and means that agents in the same community have a stronger probability to get connected than agents in different communities (the case $a < b$ is called the disassortative case). In mathematics, this model of an observation graph is called an inhomogeneous Erdös-Rényi sparse random graph.

Given an instance of the observation graph $\mathbf{G}$ (or equivalently given the edge-variables $\{e_{ij}\}$ for $1 \leq 1 < j \leq N$), the goal is to find a vector $\hat{\boldsymbol{x}} \in \{\pm 1\}^N$ as close as possible to the vector $\boldsymbol{x}^*$. One can show that, in the limit $N \to \infty$, the communities cannot be detected if

$$(a - b)^2 \leq 2(a + b) \tag{2}$$

To assess our estimate $\hat{\boldsymbol{x}}$, we define a quantity called *overlap* as

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

The quantity $q_N \in [0,1]$ and measures the quality of the estimate $\hat{\boldsymbol{x}}$. Note that we need the absolute value in the definition of the overlap, since we can only find the communities and not their associated values $\{\pm 1\}$. Indeed, both $\boldsymbol{x}^*$ and $-\boldsymbol{x}^*$ results in the same distribution for the generated graph.

We are interested in finding the average (over prior distribution on $\boldsymbol{x}^*$, and over the random graph G) overlap:

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

In the limit $N \to \infty$, this quantity can have sudden behaviour changes as a function of a and b. These changes indicate phase transitions (here eq.(2)).

1.1 Posterior distribution

Following the *Bayesian* approach, we choose our estimate to be the posterior mean. Using Bayes rule, the posterior distribution reads:

$$\begin{aligned}\mathbb{P}\big(\boldsymbol{x}|\{e_{ij}\}\big) &= \frac{\mathbb{P}\big(\{e_{ij}\}|\boldsymbol{x}\big)\mathbb{P}(\boldsymbol{x})}{\sum_{\boldsymbol{x}}\mathbb{P}\big(\{e_{ij}\}|\boldsymbol{x}\big)\mathbb{P}(\boldsymbol{x})} \overset{(a)}{=} \frac{\prod_{1\le i<j\le N}\mathbb{P}(e_{ij}|x_ix_j)\prod_{i=1}^N\mathbb{P}(x_i)}{\sum_{\boldsymbol{x}}\prod_{1\le i<j\le N}\mathbb{P}(e_{ij}|x_ix_j)\prod_{i=1}^N\mathbb{P}(x_i)} \\ &\overset{(b)}{=} \frac{\prod_{1\le i<j\le N}\mathbb{P}(e_{ij}|x_ix_j)}{\sum_{\boldsymbol{x}\in\{\pm1\}^N}\prod_{1\le i<j\le N}\mathbb{P}(e_{ij}|x_ix_j)}\end{aligned}$$

In (a) we used the independence of edges and labels of the nodes, and in (b) we used the fact that $\mathbb{P}(x_i)$ is uniform on $\{\pm 1\}$. Exploiting the fact that $e_{ij} \in \{0,1\}$ and $x_i \in \{\pm 1\}$, we can write

$$\begin{aligned}\mathbb{P}(e_{ij}|x_ix_j) &= \big(\frac{a-b}{2N}x_ix_j + \frac{a+b}{2N}\big)^{e_{ij}}\big(1 - \frac{a-b}{2N}x_ix_j - \frac{a+b}{2N}\big)^{1-e_{ij}} \\ &= \exp\Big\{e_{ij}\ln\big(\frac{a-b}{2N}x_ix_j + \frac{a+b}{2N}\big) + (1-e_{ij})\ln\big(1 - \frac{a-b}{2N}x_ix_j - \frac{a+b}{2N}\big)\Big\} \\ &= \exp\left\{e_{ij}\big(\frac{1}{2}x_ix_j\ln\frac{a}{b} + \frac{1}{2}\ln ab - \ln N\big) + (1-e_{ij})\Big(\frac{1}{2}x_ix_j\ln\frac{1-\frac{a}{N}}{1-\frac{b}{N}} + \frac{1}{2}\ln\big(1-\frac{a}{N}\big)\big(1-\frac{b}{N}\big)\Big)\right\} \\ &\overset{(c)}{\propto} \exp\left\{\frac{1}{2}x_ix_j\Big[e_{ij}\ln\frac{a}{b} + (1-e_{ij})\ln\frac{1-\frac{a}{N}}{1-\frac{b}{N}}\Big]\right\}\end{aligned}$$

To find (c), we dropped the terms independent of x_ix_j since these terms cancel when we plug the numerator and denominator in the posterior distribution. We finally have

$$\mathbb{P}\big(\boldsymbol{x}|\{e_{ij}\}\big) \propto \frac{\prod_{1\le i<j\le N}e^{h_{ij}x_ix_j}}{\sum_{\boldsymbol{x}\in\{\pm1\}^N}\prod_{1\le i<j\le N}e^{h_{ij}x_ix_j}}, \quad h_{ij} = \frac{1}{2}\Big[e_{ij}\ln\frac{a}{b} + (1-e_{ij})\ln\frac{1-\frac{a}{N}}{1-\frac{b}{N}}\Big] \tag{3}$$

This distribution is the Gibbs distribution of the *Ising* model on a complete graph with N nodes and the *Hamiltonian* (or energy, or cost function, to be minimized) is

$$\mathcal{H}(\boldsymbol{x}) = -\sum_{1\le i<j\le N}h_{ij}x_ix_j \tag{4}$$

The Bayesian estimator is

$$\hat{x}_i = \mathbb{E}_{\boldsymbol{x}|\{e_{ij}\}}[x_i] \equiv \langle x_i\rangle \tag{5}$$

The performance measure of this estimator is then obtained by plugging it the expression for $\bar{q}_N$ above. It can be shown that the Bayesian estimator (5) is the best possible in the sense that it maximizes q_N among all possible estimators.

2 MCMC Algorithms

You will investigate the behavior of different MCMC algorithms to estimate the empirical overlap $q_N(t) = \frac{1}{N}\sum_{i=1} x_i(t)x_i^*$ where $x_i(t)$ is the state of the MCMC chain at time t. This empirical overlap will have to be averaged over many MCMC runs, graph instances, and maybe over a suitable time interval (all this is up to you to decide and investigate). The algorithms to study are:

- *The standard Metropolis algorithm.*
- *The Houdayer algorithm*:

  This is a variant of the Metropolis algorithm in which instead of single flips, we have multiple flips. Here, we will use a version of the Houdayer algorithm adapted to the community detection problem. The algorithm is initialized with a pair of spin configurations $(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$ and at each step, it works as follows:

  1. For the pair of configurations $(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$, compute for every node i the value of $y_i = x_i^{(1)}x_i^{(2)}$ (this is called the "local overlap" at node i.)

  2. The overlap computed in the previous step defines the clusters *on the observed graph* $\mathbf{G}$, which are the connected parts (including connections on the observed graph $\mathbf{G}$ only) having the same overlap value. Select at random a node for which $y_i = -1$ and flip the cluster to which it belongs in both configurations.

  3. Perform a single flip Metropolis algorithm on each of the new pairs independently.

  Implementation Hint: For the Houdayer move, we suggest using the NetworkX package to perform the graph operations you need.

- *The Mixed Metropolis-Houdayer algorithm*:

  This algorithm is like Houdayer's algorithm, except that instead of performing the Houdayer move in each step, you do it in some portion of steps (every n_0 steps).

The Houdayer algorithm belongs to the class of "cluster algorithms" where the move involves suitable clusters of variables. This allows to make "non-local" moves in the state space and to (sometimes) better explore the landscape. For the ferromagnetic Ising model, this improves the performance and/or the mixing times. In this project, one of your goals will be to assess if this is also the case - or not - for the SBM.

3 Specific questions and indications

Take as parameters the average degree of the graph $d = \frac{1}{2}(a+b)$ and $r = \frac{b}{a} \in (0, 1]$.

- (*Phase transition*) From (2), find the critical value r_c (as a function of d) at which the phase transition occurs.

You have seen in the class that the Metropolis chain is ergodic. We want to check if this is the case for the Houdayer algorithm (without the last step). Consider the chain on the pair of configurations $(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$, at each step we do a Houdayer move (steps 1 & 2).

- (*Ergodicity of Houdayer*) Is the chain on the pair configurations constructed from Houdayer move ergodic?

  (*Hint:* Compare the sum of energy of two configurations $\mathcal{H}(\boldsymbol{x}^{(1)}) + \mathcal{H}(\boldsymbol{x}^{(2)})$ before and after each move.)

For the simulations, you are free to fix parameters as you wish and explore. However here are a few guidelines and reasonable values: $d = 3$, N ranges between 100 and 1000, make about 100 experiments to compute the empirical average of the overlap.

Explore and compare the above algorithms and in particular:

- Investigate the average overlap as a function of time. Compare the convergence time of the various algorithms (a meaningful measure of time is given by the number of steps of each algorithm).
- Plot the average (over experiments) of $\lim_{t\to+\infty} q_N(t)$ as a function of $\frac{b}{a} \in (0, 1]$. Compare the limiting performance of different algorithms.
- Try to pinpoint the phase transition. Ideal phase transitions occur for $N \to +\infty$, so you will necessarily find a smooth curve for finite N. By looking at the evolution of curves for many N values, you may be able to get a rough indication of the phase transition point.

For the standard Metropolis algorithm, you should be able to increase N to 1000 to get the plot. However, the original Houdayer algorithm for N large has a considerable running time, so you are required to provide the plots for N as large as it has a reasonable running time. On the other hand, the advantage of cluster moves begins to make a difference for large sizes, so it is suggested to compare the algorithms for large N for some ratios of b/a. The mixed algorithm has the advantage of cluster moves, but with fewer computations than the original Houdayer algorithm, so you should be able to try the algorithm for larger sizes.

4 Deadlines

- You should work in teams of 3. Please send to Farzad, Nicolas and Olivier an email with the composition of your team and find also a *name* for your team. Please do this by **Tuesday, Nov 30, 23h59**.
- We expect a report from each team with approx 4-5 pages describing your approach and results. Please do not plot a million graphs, but just the ones that matter! The report is due on **Monday, Dec 20, 23h59**.
- A competition will take place on **Thursday, Dec 23, 12h15** (= usual class schedule), where you will have the opportunity to try your algorithm(s) on a real dataset. At least one member of each team should be present in the room to be allowed to participate to the competition, but we of course encourage you to be all present, so that you will have more fun working in teams!