Éléments de processus stochastiques

Méthodes de Monte Carlo par chaînes de Markov pour la détection de communautés dans un graphe

Prof. Pierre Geurts

Encadrants : Vân Anh Huynh-Thu, Yann Claes

Année académique 2021-2022

Le travail est à réaliser par groupe de **trois** étudiants. Il sera encadré au moyen de séances de travaux dirigés qui auront lieu les mardis de 10h30 à 12h30 au local Raikem (B31). Seules les questions posées lors de ces séances ou sur le forum de discussion sur Ecampus donneront lieu à des réponses de la part des encadrants. Nous encourageons vivement les étudiants à se documenter eux-mêmes sur les sujets abordés dans ce projet. Toute source utilisée devra évidemment être citée dans le rapport.

Le rapport et le code source sont à remettre via la plateforme de soumission de Montefiore pour **le 10 mai 2022 à 22h00** au plus tard.

Contexte général et objectifs

L'objectif général du projet est de développer un algorithme permettant de résoudre le problème de détection de communautés dans un (grand) graphe. Pour construire ce système, on se basera sur la méthode de Monte Carlo par chaînes de Markov (MCMC pour *Markov chain Monte Carlo* en anglais).

Le projet est divisé en deux parties. La première partie du projet a pour but de vous familiariser avec les chaînes de Markov et la méthode MCMC. La deuxième partie, qui constitue le cœur du projet, vise à mettre en œuvre concrètement la méthode MCMC pour résoudre le problème de détection de communautés.

Définitions et notations

Dans cette section, nous fournissons les définitions et notations minimales nécessaires pour la première partie du projet. Nous vous encourageons néanmoins à consulter d'autres références pour obtenir plus de détails à propos de la méthode MCMC.

Chaînes de Markov. Soit une suite de variables aléatoires $\{X_1, X_2, \ldots, X_t, \ldots\}$. Cette suite définit un modèle (ou une chaîne) de Markov d'ordre 1 ssi, pour tout $t \geq 1$, la distribution conjointe de t premières variables peut se factoriser comme suit :

$$\mathbb{P}(X_1, X_2, \ldots, X_t) = \mathbb{P}(X_1) \prod_{l=2}^{t} \mathbb{P}(X_l | X_{l-1}).$$

Dans cette partie, on supposera que le modèle de Markov est discret et on notera sans restriction $\{1, \ldots, N\}$ l'ensemble des valeurs possibles des variables X_i, appelées les états.

Méthode de Monte Carlo. La méthode de Monte Carlo de base permet d'estimer la valeur de l'espérance $E\{h(X)\}$ d'une fonction h d'une variable aléatoire X en utilisant la moyenne empirique

$$\hat{I} = \frac{1}{n} \sum_{i=1}^{n} h\left(x^{(i)}\right),$$

où les $x^{(i)}$ ont été générés aléatoirement *i.i.d.* selon la densité p_X. Lorsque que la variance de $h(X)$ est finie, $\hat{I}$ converge presque sûrement vers $E\{h(X)\}$, avec une erreur standard qui décroit en $1/\sqrt{n}$ avec la taille de l'échantillon. Des versions plus générales de la loi des grands nombres permettent de garantir la convergence de cette méthode dans certains cas où les $x^{(i)}$ ne sont pas indépendantes ou lorsque la variance de $h(X)$ n'est pas finie.

Méthodes MCMC. Les méthodes MCMC consistent à simuler une chaîne de Markov ergodique dont la distribution stationnaire est égale à p_X. Dans ce projet, nous utiliserons l'algorithme de Metropolis-Hastings. Cet algorithme utilise une fonction q appelée *densité de proposition* qui, en pratique, est telle qu'il est possible de générer des échantillons selon cette densité.

Algorithm 1 Algorithme de Metropolis-Hastings.

Set starting state $x^{(0)}$,$t = 1$
while convergence not reached **do**
  Generate $y^{(t)} \sim q(y|x^{(t-1)})$
  $\alpha \leftarrow \min\left\{1, \frac{p_X(y^{(t)})}{p_X(x^{(t-1)})} \frac{q(x^{(t-1)}|y^{(t)})}{q(y^{(t)}|x^{(t-1)})}\right\}$
  Generate u uniformly in $[0, 1[$
  if $u < \alpha$ **then**
    $x^{(t)} \leftarrow y^{(t)}$
  else
    $x^{(t)} \leftarrow x^{(t-1)}$
  end if
  $t \leftarrow t + 1$
end while

1 Première partie : chaînes de Markov et algorithme MCMC

La première partie du projet permet de se familiariser avec les chaînes de Markov et de comprendre comment l'algorithme MCMC fonctionne en l'appliquant sur un problème simple d'échantillonnage selon une distribution binomiale.

1.1 Chaînes de Markov

Soit la chaîne de Markov à 4 états définie par la matrice de transition Q suivante :

$$Q = \begin{pmatrix} 0 & 0.1 & 0.1 & 0.8 \\ 1 & 0 & 0 & 0 \\ 0.6 & 0 & 0.1 & 0.3 \\ 0.4 & 0.1 & 0.5 & 0 \end{pmatrix},$$

où $Q_{ij} = \mathbb{P}(X_{t+1} = j | X_t = i)$.

Questions :

1. Calculez (numériquement) les quantités suivantes pour des valeurs de t croissantes :
   (1/4, 1/4, 1/4, 1/4) — $\mathbb{P}(X_t = x)$, où $x = 1, 2, 3, 4$, en supposant que le premier état est choisi au hasard, i.e. l'état initial est tiré dans une loi uniforme discrète.
   (0, 0, 1, 0) — $\mathbb{P}(X_t = x)$, où $x = 1, 2, 3, 4$, en supposant que l'état initial est toujours 3,
   — Q^t, c'est-à-dire la t-ième puissance de la matrice de transition.
   Représentez l'évolution des deux premières grandeurs sur un graphe (en traçant une courbe par état). Discutez et expliquez les résultats obtenus sur base de la théorie.
Théorique+graphe 2. En déduire la distribution stationnaire π_∞ de la chaîne de Markov définie par $[\pi_\infty]_j = \lim_{t\to\infty} \mathbb{P}(X_t = j)$.
3. Générez une réalisation aléatoire de longueur T de la chaîne de Markov. Calculez pour chaque état le nombre de fois qu'il apparaît dans la réalisation divisé par la longueur de la réalisation. Observez l'évolution de ces valeurs pour chaque état lorsque T croît.
4. Que concluez-vous de cette expérience ? Reliez ce résultat à la théorie.

Théorique

1.2 Méthode MCMC : analyse théorique dans le cas fini

Les méthodes MCMC peuvent être appliquées aussi bien à des variables continues que discrètes, prenant leurs valeurs dans un ensemble fini ou infini et il en existe de nombreuses variantes. Cependant, dans le cadre de ce travail nous nous focaliserons sur l'application de cette méthode dans le cas où la variable cible est discrète à valeurs finies, et nous nous limiterons à l'étude de l'algorithme de Metropolis-Hastings.

Questions :

1. Etant donné une matrice de transition Q et une distribution initiale π_0 d'une chaîne de Markov invariante dans le temps qui satisfont les équations de balance détaillée, c'est-à-dire :
$$\forall i, j \in \{1, \ldots, N\} : \pi_0(i)[Q]_{i,j} = \pi_0(j)[Q]_{j,i}, \tag{1}$$
montrez que π_0 est une distribution stationnaire de la chaîne de Markov. Dans quel(s) cas celle-ci est-elle unique ?
2. Dans le cas où X est discrète, démontrez que l'application de l'algorithme de Metropolis-Hastings en remplaçant p_X par une fonction f telle que $\forall x : f(x) = cp_X(x)$, où c est une constante, génère une chaîne de Markov qui satisfait les équations de balance détaillée avec $p_X(x)$ comme distribution stationnaire. Quelles autres conditions la chaîne doit-elle respecter pour que l'algorithme de Metropolis-Hastings fonctionne ?
Indice : commencez par écrire les probabilités de transition en prenant en compte les différents cas possibles (rejet ou acceptation) et vérifiez que les équations de balance détaillée sont satisfaites.

1.3 Méthode MCMC : illustration sur un exemple simple

Avant d'utiliser la méthode pour résoudre le problème de détection de communautés, nous allons l'appliquer sur un problème simple, afin de mieux comprendre comment elle fonctionne et comment l'appliquer.

Questions : Soit la distribution binomiale définie sur les entiers $\{0, 1, 2, 3, \ldots, K\}$ par :

$$\mathbb{P}(X = k) = C_K^k p^k (1-p)^{K-k},$$

où p est la probabilité de succès. On souhaite appliquer l'algorithme de Metropolis-Hastings pour échantillonner des valeurs suivant cette loi, avec comme paramètres $p = 0.3$ et $K = 10$. Pour ce faire, on propose d'utiliser la distribution de proposition suivante :

$$q(y|x) = \begin{cases} r & \text{si } x = 0 \text{ et } y = 0, \\ (1-r) & \text{si } x = K \text{ et } y = K, \\ r & \text{si } 0 < x \leq K \text{ et } y = x - 1, \\ (1-r) & \text{si } 0 \leq x < K \text{ et } y = x + 1, \\ 0 & \text{sinon.} \end{cases}$$

où r est un paramètre $\in]0, 1[$.

1. Sur base de l'analyse du point précédent, montrez que pour ce choix de distribution de proposition l'algorithme de Metropolis-Hastings permettra bien de générer des échantillons selon la distribution p_X.
2. Générez une réalisation suffisamment longue de la chaine et étudiez la convergence de la moyenne et de la variance des valeurs ainsi générées vers les valeurs théoriques attendues en fonction de la longueur de cette réalisation. Testez deux valeurs de r : 0.1 et 0.5.
3. Tracez un histogramme des fréquences d'apparition de chaque valeur dans la réalisation et comparez cet histogramme avec la distribution théorique, pour les deux mêmes valeurs de r.

2 Deuxième partie : détection de communautés dans un graphe par algorithmes MCMC

L'objectif de cette seconde partie est de développer un algorithme pour partitioner les nœuds d'un graphe en sous-groupes, appelés des communautés, en se basant sur l'hypothèse qu'il y a plus d'arêtes dans le graphe entre des nœuds d'une même communauté qu'entre des nœuds de communautés différentes.

Il existe plusieurs manières de formaliser ce problème et autant d'algorithmes plus ou moins efficaces pour le résoudre [For10]. Dans ce projet, on abordera ce problème à l'aide de l'algorithme de Metropolis-Hastings. L'idée générale de l'approche sera la suivante :

— On fera l'hypothèse que les communautés et le graphe ont été générés aléatoirement à partir d'un modèle probabiliste particulier, appelé modèle à blocs stochastiques (qu'on dénotera SBM plus loin, pour "Stochastic Block Model" [Abb18]).
— Sur base de ce modèle et du théorème de Bayes, on calculera la probabilité d'une partition particulière des nœuds en communautés après avoir observé le graphe.
— On utilisera l'algorithme Metropolis-Hastings pour échantillonner selon cette distribution, ce qui nous permettra entre autres d'identifier la partition en communauté la plus probable.

Plus de détails sont donnés ci-dessous sur le modèle et l'approche avant de préciser les questions qui vous seront posées, qui sont divisées en trois parties.

SBM —> Modèles à blocs stochastiques. Soient :

- N le nombre de nœuds du graphe,
- K le nombre de communautés,
- $p = (p_1, \ldots, p_K)$ une distribution discrète sur les communautés ($\forall i, p_i \in [0,1]$ et $\sum_{i=1}^{K} p_k = 1$),
- W une matrice $K \times K$ symétrique dont les entrées $W_{i,j} \in [0,1]$ représentent des probabilités de connexion entre nœuds de différentes communautés,,
- $x \in \{1, \ldots, K\}^N$ un vecteur indiquant la communauté de chaque nœud,
- $G \in \{0,1\}^{N \times N}$ une matrice d'adjacence symétrique encodant un graphe non dirigé et sans boucle ($G_{ii} = 0\ \forall i$) défini sur ces nœuds.

Une paire (x, G) est générée selon un modèle $SBM(N, K, p, W)$, si :

- chaque composante x_i est générée aléatoirement *i.i.d.* de la distribution p,
- deux nœuds i et j ($i \neq j$) sont connectés avec une probabilité $W_{x_i x_j}$, c'est-à-dire uniquement en fonction de leurs communautés respectives et indépendamment des autres nœuds ou arêtes du graphe.

On ne s'intéressera dans ce projet qu'aux modèles SBM dits de partition plantée, c'est-à-dire tels que $W_{ii} = A\ \forall i$ et $W_{ij} = B\ \forall i, j : i \neq j$ et assortatif, c'est-à-dire tels que $A > B$. On notera par la suite un tel modèle $SBM(N, K, p, A, B)$.

Détection de communautés. Le problème de détection de communautés dans ce contexte revient au problème de la détermination du vecteur de communautés x à partir uniquement de l'observation d'un graphe G, sous l'hypothèse que (x, G) est tiré d'un modèle $SBM(N, K, p, A, B)$. Même si ça constitue une hypothèse très forte en pratique, pour simplifier le problème, on supposera par la suite que les paramètres du modèle SBM ayant généré le graphe, c'est-à-dire p, A et B, sont connus.

Soit un graphe G, on notera par la suite x^* le vecteur de communautés ayant servi à sa génération et $\hat{x}$ une estimation de ce vecteur. Pour mesurer la qualité d'une estimation, on utilisera la concordance $A(x^*, \hat{x})$ entre x^* et $\hat{x}$ définie de la manière suivante :

$$A(x^*, \hat{x}) = \max_{\pi \in S_K} \frac{1}{N} \sum_{i=1}^{N} 1(x_i^* = \pi(\hat{x}_i)), \tag{2}$$

où S_K est l'ensemble des permutations π sur $\{1, \ldots, K\}$. La maximisation sur S_K permet de tenir compte du fait qu'il n'est possible de déterminer les communautés qu'à une permutation près, les indices de communautés étant interchangeables.

Il existe une littérature très abondante sur les modèles SBM et la détection de communautés, notamment plusieurs résultats théoriques sur l'identifiabilité des communautés sur base du graphe (voir [Abb18] pour une synthèse de ces résultats). Un résultat intéressant du domaine est d'avoir montré qu'il existait un phénomène de transition de phase : sous certaines conditions, en dessous d'un certain écart entre A et B, il devient impossible de détecter les communautés. C'est évidemment impossible lorsque $A = B$ mais ça le reste même quand A et B commencent à différer. Plus formellement, il a été montré que lorsque N tend vers l'infini et que A et B décroissent avec N comme a/N et b/N respectivement, où a et b sont deux constantes, les communautés peuvent être détectées (sous-entendu mieux que par une méthode purement aléatoire) si et seulement si :

$$(a - b)^2 > 2(a + b). \tag{3}$$

Les conditions $A = a/N$ et $B = b/N$ expriment le fait que le degré moyen des nœud reste constant même lorsque la taille du graphe augmente. On vérifiera ce phénomène dans la section 2.2.

Metropolis-Hastings pour la détection de communautés. Pour estimer le vecteur de communautés, on pourrait directement rechercher l'estimation $\hat{x}$ qui maximise la probabilité $\mathbb{P}(x|G)$ mais le problème serait trop complexe comme on le montrera dans la section 2.1. A la place, on utilisera dans ce projet une solution approchée basée sur l'algorithme de Metropolis-Hastings. Cet algorithme sera utilisé pour échantillonner des solutions candidates selon la distribution $\mathbb{P}(x|G)$ et on identifiera parmi toutes les solutions générées celle qui maximise la probabilité $\mathbb{P}(x|G)$[1]. On utilisera le même algorithme dans la section 2.2 pour vérifier l'existence du seuil de transition (3).

Pour implémenter l'algorithme MH, une distribution de proposition simple, notée q_s dans la suite, consiste simplement à choisir aléatoirement un nœud du graphe à en changer la communauté aléatoirement. D'autres approches plus efficaces peuvent néanmoins être envisagées, que vous pourrez explorer dans le contexte de la question de la section 2.3.

2.1 Etude théorique

Avant d'entamer l'implémentation de l'algorithme de détection, on vous demande ici de l'étudier au préalable de manière théorique.

Questions :

1. En utilisant le théorème de Bayes, explicitez la distribution $\mathbb{P}(x|G)$, d'abord dans le cas du modèle général $SBM(N, K, p, W)$ et ensuite dans le cas du modèle simplifié $SBM(N, K, p, A, B)$.
2. Expliquez pourquoi le calcul de $\mathbb{P}(x|G)$ est impossible pour des valeurs de N élevées et pourquoi ce n'est pas un problème pour utiliser l'algorithme de Metropolis-Hastings.
3. Justifiez la validité de la distribution de proposition q_s discutée plus haut.
4. Déduisez de $\mathbb{P}(x|G)$ l'expression de la probabilité d'acceptation α de l'algorithme de Metropolis-Hastings pour cette distribution de proposition.

2.2 Analyse expérimentale

Dans cette section, on se propose d'étudier empiriquement la qualité de l'identification des communautés en fonction des paramètres A et B. On se placera dans le cas de deux communautés ($K = 2$) et d'une distribution de communautés $p = (1/2, 1/2)$ uniforme. Pour se mettre dans les conditions de l'inégalité (3), on prendra $A = a/N$ et $B = b/N$.

Questions :

1. Prenez une valeur N aussi grande que possible, fixez le degré moyen $\frac{a+b}{2}$ à une valeur donnée (par exemple 3) et étudiez l'évolution de la concordance moyenne :

$$E_{(x^*,G)\sim SBM(N,2,p,a/N,b/N)}\{E_{\mathbb{P}(x|G)}\{A(x^*, x)\}\} \tag{4}$$

1. Cette utilisation de l'algorithme de Metropolis-Hastings pour résoudre un problème d'optimisation s'appelle l'algorithme de recuit simulé (*simulated annealing*).

en fonction du rapport $\frac{b}{a}$ dans $[0, 1]$. La première espérance est une espérance sur différentes paires (x^*, G), qui vous demandera d'implémenter un générateur de graphes selon le modèle SBM. La seconde est une espérance sur la distribution $\mathbb{P}(x|G)$ qu'on peut estimer par l'algorithme de Metropolis-Hastings. Il vous appartient de choisir les paramètres de cet algorithme pour obtenir la courbe la plus pertinente possible.

2. Analysez la ou les courbes obtenues en fonction de ce qui est prédit par l'inégalité (3) et en comparant avec la concordance moyenne qu'on obtiendrait en choisissant les communautés de manière purement aléatoire.

2.3 Application à un grand graphe

Lors d'une des dernières séances de cours, un graphe réel vous sera fourni et vous devrez appliquer l'algorithme de Metropolis-Hastings que vous aurez développé pour obtenir la meilleure estimation possible du vecteur de communautés sous-jacent. Les soumissions de tous les groupes seront évaluées sur base de la concordance avec les communautés réelles et les meilleurs groupes obtiendront un bonus de points.

Afin de préparer votre code, la procédure sera la suivante :

— Le graphe sera représenté sous la forme d'une liste d'arêtes. Des fichiers dans différents formats et pour différents langages vous seront fournis. Un exemple de fichier sera disponible sur Ecampus.

— Comme à la section précédente, la valeur de K sera fixée à 2. Le vecteur à nous envoyer devra contenir une valeur 1 ou 2 pour chaque nœud. Un exemple de fichier attendu vous sera également fourni sur Ecampus.

— La taille du graphe ne sera pas précisée à l'avance mais vous pouvez vous attendre à un graphe de plus de 5000 nœuds.

— Nous vous préciserons les valeurs de A et B, ainsi que la distribution p. Le graphe étant un graphe réel, la valeur de A sera calculée sur base des communautés comme la moyenne des probabilités de connexions au sein des deux communautés et elle pourra donc différer entre les deux communautés.

En préparation de cette séance, nous vous encourageons à tester votre code sur des grands graphes, que vous pouvez générer vous-mêmes ou que vous pouvez récupérer d'autres sources. Le temps d'exécution de l'algorithme sera limité à la durée de la séance et donc, il est important de prendre en compte les temps de calcul. En particulier, il peut valoir la peine d'analyser l'impact des paramètres de l'algorithme MH (nombres d'itérations, état initial, etc.) sur les performances et d'explorer d'autres distributions de proposition que celle proposée plus haut, qui pourraient converger plus vite vers la bonne solution.

Questions : Dans votre rapport, on vous demande de préciser et de justifier vos choix d'implémentation, potentiellement via des expériences réalisées sur des graphes aléatoires. Les résultats de la compétition seront divulgués avant la remise du rapport final de manière à ce que vous puissiez les prendre en compte pour la finalisation de votre rapport.

3 Rapport et code

Vous devez nous fournir un rapport au format pdf contenant vos réponses, concises mais précises, aux questions posées ainsi que le code que vous avez utilisés pour y répondre, le tout

sous forme d'archive zip. La longueur attendue du rapport est entre 15 et 30 pages (figures et bibliographie incluse, police de taille 11, pas de code dans le rapport).

Vous pouvez réaliser le projet en utilisant Matlab, Python, ou C. Si vous souhaitez utiliser un autre langage, demandez au préalable l'accord des encadrants. Quel que soit le langage, l'algorithme de Metropolis-Hastings doit cependant être implémenté par vos soins. Aucun outil existant solutionnant le problème considéré ne peut être utilisé. Vous pouvez néanmoins utiliser des outils de manipulation de graphes. En cas de doute, contactez toujours les encadrants.

4 Références

Toutes les références, les données, et les codes relatifs au projet seront disponibles sur Ecampus. Des conseils et réponses aux questions fréquentes seront également rassemblées sur cette page au fur et à mesure. Un forum de discussion a également été ouvert. Veillez à consulter régulièrement Ecampus tout au long du projet.

Références

[Abb18] Emmanuel Abbe. Community detection and stochastic block models : Recent developments. *Journal of Machine Learning Research*, 18(177) :1–86, 2018.

[For10] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5) :75 – 174, 2010.