Inférence efficace des modèles à blocs stochastiques et à blocs latents pour les graphes creux

Gabriel Frisch, Jean-Benoist Leger, Yves Grandvalet

▶ **To cite this version:**

Gabriel Frisch, Jean-Benoist Leger, Yves Grandvalet. Inférence efficace des modèles à blocs stochastiques et à blocs latents pour les graphes creux. 52èmes Journées de Statistiques de la Société Française de Statistique (SFdS - jds 2020), May 2020, Nice, France. hal-02973828

HAL Id: hal-02973828
https://hal.science/hal-02973828v1

Submitted on 21 Oct 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Inférence efficace des modèles à blocs stochastiques et à blocs latents pour les graphes creux

Gabriel Frisch, Jean-Benoist Leger & Yves Grandvalet

Université de Technologie de Compiègne, CNRS, Heudiasyc UMR 7253, Compiègne France. {gabriel.frisch,jbleger,yves.grandvalet}@hds.utc.fr

Résumé. Nous présentons un algorithme d'inférence variationnelle pour les modèles à blocs stochastiques et à blocs latents pour les graphes creux, qui tire parti de la rareté des arêtes pour passer à l'échelle sur un très grand nombre de nœuds. Cet algorithme est implémenté sous la forme d'un module python, appelé *sparsebm*, qui peut traiter des graphes comportant des millions de nœuds.
Mots-clés. Co-clustering, LBM, SBM, inférence variationnelle, graphe creux, matrice creuse

Abstract. We present a variational inference algorithm for the Stochastic Block Model and the Latent Block Model for sparse graphs, which leverages on the sparsity of edges to scale up to a very large number of nodes. This algorithm is implemented as a python package, called *sparsebm*, which can handle graphs with millions of nodes.
Keywords. Co-clustering, LBM, SBM, variationnal inference, sparse graph, sparse matrix

1 Introduction

Dans un graphe dense, l'adjacence des noeuds est traditionnellement représentée sous la forme d'une matrice d'adjacence. Ce choix est motivé par la simplicité de sa représentation mais aussi par la performance des calculs vectoriels fournis par les outils de programmation. Lorsque le degré moyen des nœuds est faible, les matrices d'adjacences comportent majoritairement des zéros et peuvent être qualifiées de « creuses ». Ces types de graphes apparaissent souvent dans les jeux de données issues de réseaux sociaux ou de systèmes collaboratifs. Par exemple, le jeu de données de LinkedIn [1] compte 3 millions d'utilisateurs avec un degré moyen de 8.3 par utilisateur. Le jeu de données MovieLens-25M [2], référence pour le filtrage collaboratif, peut être modélisé par un réseau bipartite composé de 120 000 nœuds utilisateurs et de 60 000 nœuds films, avec un degré moyen de 112. Dans ces contextes, la taille des matrices d'adjacence devient problématique pour les implémentations existantes des modèles à blocs stochastiques (SBM) ou des modèles à blocs latents (LBM). Nous montrons ici comment, en utilisant une représentation sous

1. Disponible sur https://lfs.aminer.cn/lab-datasets/multi-sns/aminer.tar.gz
2. Disponible sur https://grouplens.org/datasets/movielens/

forme de liste de l'adjacence des nœuds, il est possible de réaliser une inférence efficace du SBM et du LBM dans de très larges graphes creux.

Nous présentons dans un premier temps les modèles à blocs latents (LBM) et les modèles à blocs stochastiques (SBM) tels qu'initialement proposés par leurs auteurs. Nous comparons ensuite les inférences variationnelles originelles et celles utilisant des astuces calculatoires afin de réduire la complexité pour les graphes creux. Nous montrons enfin que, grâce à ces astuces, ces modèles peuvent être utilisés pour analyser des graphes comportant des centaines de milliers de nœuds pour peu qu'ils aient relativement peu d'arêtes. Les implémentations sont disponibles publiquement dans un module python [3] qui utilise l'accélération matérielle des GPUs.

2 Modèles

2.1 Modèle à blocs latents (LBM)

Le modèle à blocs latents (LBM) est un modèle probabiliste de co-clustering [3] qui permet de classifier simultanément les nœeuds d'un graphe bipartite, dont la matrice d'adjacence est notée $\boldsymbol{X}$. Le LBM repose sur l'hypothèse que cette matrice d'adjacence est structurée en blocs homogènes issus d'une double partition des lignes, en k_1 parties, et des colonnes, en k_2 parties, de cette matrice.

Comme introduit pour la première fois par Govaert et Nadif [2], nous considérons ici la version la plus simple du LBM, dans laquelle la matrice d'adjacence $\boldsymbol{X}$, de taille $n_1 \times n_2$, est binaire. Ses lignes correspondent aux n_1 nœuds de type (1) et ses colonnes aux n_2 nœuds de type (2) du graphe bipartite. La variable aléatoire X_{ij} associée à chaque paire de nœuds (i, j), respectivement de type (1) et de type (2), code la présence ou l'absence d'arête entre i et j : $X_{ij} = 1$ si l'arête est présente, et $X_{ij} = 0$ sinon.

On introduit $\boldsymbol{U}$ la matrice $n_1 \times k_1$ indicatrice de l'appartenance aux classes lignes et $\boldsymbol{V}$ la matrice $n_2 \times k_2$ indicatrice de l'appartenance aux classes colonnes. Nous avons $U_{iq} = 1$ si la ligne i appartient à la classe ligne q et $U_{iq} = 0$ autrement. De façon similaire, V_{jl} prend ses valeurs dans $\{0, 1\}^{k_2}$ et indique l'appartenance de la colonne j à la classe l.

Le LBM fait plusieurs hypothèses sur la forme et la dépendance des variables :

— Les appartenances aux classes des nœuds de type (1) et des nœuds de type (2) sont *a priori* indépendantes. Les variables latentes $\boldsymbol{U}$ et $\boldsymbol{V}$ sont *a priori* indépendantes : $p(\boldsymbol{U}, \boldsymbol{V}) = p(\boldsymbol{U})p(\boldsymbol{V})$.

— Les catégories des nœuds de type (1) sont indépendantes et identiquement distribuées selon une loi multinomiale $\mathcal{M}(1; \boldsymbol{\alpha})$ où $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{k_1})$ représente les proportions du mélange des classes en ligne.

3. Module python : sparsebm, disponible sur `https://pypi.org/project/sparsebm/`

— Les catégories des nœuds de type (2) sont indépendantes et identiquement distribuées selon une loi multinomiale $\mathcal{M}(1;\boldsymbol{\beta})$ où $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{k_2})$ représente les proportions du mélange des classes en colonne.
— La présence de l'arête entre le nœud i de type (1) et le nœud j de type (2), représentée par la variable aléatoire X_{ij} est considérée comme dépendant uniquement des classes des nœuds i et j. Autrement dit, le LBM suppose que tous les éléments d'un bloc suivent la même distribution de probabilité. La densité conditionnelle pour une observation X_{ij} du bloc ql s'écrit alors : $\phi(X_{ij};\pi_{ql}) = \pi_{ql}^{X_{ij}}(1-\pi_{ql})^{1-X_{ij}}$

La variable aléatoire $\boldsymbol{X}$ suit une densité mélange :

$$\begin{aligned} p(\boldsymbol{X};\boldsymbol{\theta}) &= \sum_{\boldsymbol{U},\boldsymbol{V}\in I\times J} p(\boldsymbol{U};\boldsymbol{\alpha})\; p(\boldsymbol{V};\boldsymbol{\beta})\; p(\boldsymbol{X}|\boldsymbol{U},\boldsymbol{V};\boldsymbol{\pi}) \\ &= \sum_{\boldsymbol{U},\boldsymbol{V}\in I\times J} \prod_{i,q} {\alpha_q}^{U_{iq}} \prod_{j,l} {\beta_l}^{V_{jl}} \prod_{i,j,q,l} \phi(X_{ij};\pi_{ql})^{U_{iq}V_{jl}} \quad , \end{aligned} \tag{1}$$

où I (resp. J) représente l'ensemble des partitions possibles pour les lignes (resp. colonnes) et $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\pi})$ représente le vecteur de paramètres du modèle.

2.2 Modèle à blocs stochastiques (SBM)

Le modèle SBM [4] peut être vu comme un LBM contraint à une seule partition des nœuds du graphe. Il ne s'applique donc plus aux graphes bipartite, mais il repose toujours sur l'hypothèse d'une structure de la matrice d'adjacence $\boldsymbol{X}$ en blocs homogènes. Le SBM fait plusieurs hypothèses sur la forme et la dépendance des variables :
— Les appartenances aux classes des nœuds sont indépendantes et identiquement distribuées selon une loi multinomiale $\mathcal{M}(1;\boldsymbol{\alpha})$ où $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{k_1})$ représente les proportions du mélange des classes.
— La présence de l'arête entre le nœud i et le nœud i', noté $X_{ii'}$ est considérée comme dépendant uniquement des classes des nœuds i et i'. Autrement dit, le SBM suppose que tous les éléments d'un bloc suivent la même distribution de probabilité. La densité conditionnelle pour une observation $X_{ii'}$ du bloc qq' s'écrit alors : $\phi(X_{ii'};\pi_{qq'}) = \pi_{qq'}^{X_{ii'}}(1-\pi_{qq'})^{1-X_{ii'}}$.

La variable aléatoire $\boldsymbol{X}$ suit une densité mélange :

$$p(\boldsymbol{X};\boldsymbol{\theta}) = \sum_{\boldsymbol{U}\in I} \prod_{i,q} {\alpha_q}^{U_{iq}} \prod_{i',q'} {\alpha_{q'}}^{U_{i'q'}} \prod_{i,q,i',q'} \phi(X_{ii'};\pi_{qq'})^{U_{iq}U_{i'q'}} \quad ,$$

où I représente l'ensemble des partitions possibles et $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\pi})$ représente le vecteur de paramètres du modèle.

3 Inférence

Afin d'inférer les paramètres du modèle pour réaliser le co-clustering, nous maximisons la vraisemblance de chaque modèle. L'estimateur du maximum de vraisemblance doit mener à $\widehat{\theta} = \arg\max_{\theta} p(\boldsymbol{X}; \theta)$. Cependant, la vraisemblance n'est pas calculable car elle implique un nombre de termes exponentiel dans la taille du graphe. Pour pallier cela, une reformulation variationnelle du critère est utilisée dans l'algorithme EM [1].

Inférence du modèle à blocs latents Nous présentons l'inférence variationnelle proposée par Govaert et Nadif [3] que nous adaptons afin d'utiliser la faible connectivité des nœuds pour accélérer les calculs. Le critère variationnel, borne inférieure de la vraisemblance est :

$$\mathcal{J}(q_\gamma, \boldsymbol{\theta}) = \mathcal{H}(q_\gamma) + \mathbb{E}_{q_\gamma}[\log p(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{V}; \boldsymbol{\theta})] \ , \tag{2}$$

avec q_γ représentant la distribution variationnelle, $\mathcal{H}$ l'entropie différentielle et $\mathbb{E}_{q_\gamma}$ l'espérance sous la distribution variationnelle. La distribution variationnelle est restreinte à un ensemble de distributions factorisables (approximation champ moyen ou *mean field* [5]) de la forme : $q_\gamma = \prod_i \mathcal{M}(1, \tau_i^{(U)}) \ \prod_j \mathcal{M}(1, \tau_j^{(V)})$, où $\gamma = (\tau^{(U)}, \tau^{(V)})$ est le vecteur des paramètres variationnels. En utilisant l'indépendance des variables latentes, le critère se réécrit sous la forme :

$$\mathcal{J}(q_\gamma, \boldsymbol{\theta}) = \mathcal{H}(q_\gamma) + \mathbb{E}_{q_\gamma}[\log p(\boldsymbol{U}; \boldsymbol{\alpha})] + \mathbb{E}_{q_\gamma}[\log p(\boldsymbol{V}; \boldsymbol{\beta})] + \mathbb{E}_{q_\gamma}[\log p(\boldsymbol{X}|\boldsymbol{U}, \boldsymbol{V}; \boldsymbol{\theta})] \ , \tag{3}$$

avec

$$\mathcal{H}(q_\gamma) = -\sum_{iq} \tau_{iq}^{(U)} \log \tau_{iq}^{(U)} - \sum_{jl} \tau_{jl}^{(U)} \log \tau_{jl}^{(V)} \tag{4}$$

$$\mathbb{E}_{q_\gamma}[\log p(\boldsymbol{U}; \boldsymbol{\alpha})] = \sum_{iq} \tau_{iq}^{(U)} \log \alpha_q \tag{5}$$

$$\mathbb{E}_{q_\gamma}[\log p(\boldsymbol{V}; \boldsymbol{\beta})] = \sum_{jl} \tau_{jl}^{(V)} \log \beta_l \tag{6}$$

$$\mathbb{E}_{q_\gamma}[\log p(\boldsymbol{X}|\boldsymbol{U}, \boldsymbol{V}; \boldsymbol{\theta})] = \sum_{ijql} \tau_{iq}^{(U)} \tau_{jl}^{(V)} (X_{ij} \log \pi_{ql} + (1 - X_{ij}) \log(1 - \pi_{ql})) \ . \tag{7}$$

Le calcul coûteux est celui de l'espérance de la log-vraisemblance conditionnelle de $\boldsymbol{X}$ (7), dont la somme implique tous les éléments de la matrice d'adjacence, entraînant une complexité calculatoire en $\mathcal{O}(n_1 n_2 k_1 k_2)$ où n_1, n_2 sont les dimensions de $\boldsymbol{X}$ et k_1, k_2 sont respectivement le nombre de classes des nœuds de type (1) et le nombre de classes des nœuds de type (2).

Il est cependant possible de réécrire l'équation (7) afin d'itérer uniquement sur les éléments non nuls de la matrice entraînant une complexité calculatoire en $\mathcal{O}(\#\{ij : X_{ij} \neq$

$1\}k_1k_2$) où $\#\{ij : X_{ij} = 1\}$ désigne le nombre d'entrées non nulles dans $\boldsymbol{X}$:

$$\mathbb{E}_{q_\gamma}[\log p(\boldsymbol{X}|\boldsymbol{U}, \boldsymbol{V}; \boldsymbol{\theta})] = \sum_{qlij:X_{ij}=1} \tau_{iq}^{(U)} \tau_{jl}^{(V)} (\log \pi_{ql} - \log(1 - \pi_{ql})) + \sum_{ql} \log(1 - \pi_{ql}) \left(\sum_i \tau_{iq}^{(U)} \right) \left(\sum_i \tau_{jl}^{(V)} \right) . \quad (8)$$

En utilisant l'équation (8), le critère variationnel $\mathcal{J}(q_\gamma, \boldsymbol{\theta})$ est calculé de manière efficace pour les graphes creux.

L'algorithme EM se réécrit en une double maximisation alternée du critère $\mathcal{J}(q_\gamma, \boldsymbol{\theta})$ par rapport à q_γ et par rapport à $\boldsymbol{\theta}$. Nous contrastons ci-dessous les mises à jour des paramètres telle que définie dans le LBM original d'une part, et telle que nous la réécrivons pour les graphes creux d'autre part. La complexité mémoire qui était en $\mathcal{O}(n_1 n_2)$ est réduite à $\mathcal{O}(\#\{ij : X_{ij} = 1\})$.

EM - version originale	EM - version creuse
Étape-E	*Étape-E*
Répéter	Répéter
$\tau_{iq}^{(U)} \propto \alpha_q \prod_l \pi_{ql}^{Q_{il}} (1 - \pi_{ql})^{R_l - Q_{il}}$	$\tau_{iq}^{(U)} \propto \alpha_q \prod_{jl} (1 - \pi_{ql})^{\tau_{jl}^{(V)}} \prod_l \frac{\pi_{ql}^{Q_{il}}}{(1-\pi_{ql})^{Q_{il}}}$
avec $Q_{il} = \sum_j \tau_{jl}^{(V)} X_{ij}$ et $R_l = \sum_j \tau_{jl}^{(V)}$	avec $Q_{il} = \sum_{j:X_{ij}=1} \tau_{jl}^{(V)}$
$\tau_{jl}^{(V)} \propto \beta_l \prod_q \pi_{ql}^{S_{jq}} (1 - \pi_{ql})^{T_q - S_{jq}}$	$\tau_{jl}^{(V)} = \beta_l \prod_{iq} (1 - \pi_{ql})^{\tau_{iq}^{(U)}} \prod_q \frac{\pi_{ql}^{S_{jq}}}{(1-\pi_{ql})^{S_{jq}}}$
avec $S_{jq} = \sum_i \tau_{iq}^{(U)} X_{ij}$ et $T_q = \sum_i \tau_{iq}^{(U)}$	avec $S_{jq} = \sum_{i:X_{ij}=1} \tau_{iq}^{(U)}$
jusqu'à convergence	jusqu'à convergence
Étape-M	*Étape-M*
$\alpha_q = \frac{\sum_i \tau_{iq}^{(U)}}{n_1}$ $\quad$ $\beta_l = \frac{\sum_j \tau_{jl}^{(V)}}{n_2}$ $\quad$ $\pi_{ql} = \frac{\sum_{ij} \tau_{iq}^{(U)} \tau_{jl}^{(V)} X_{ij}}{\sum_{ij} \tau_{iq}^{(U)} \tau_{jl}^{(V)}}$	$\alpha_q = \frac{\sum_i \tau_{iq}^{(U)}}{n_1}$ $\quad$ $\beta_l = \frac{\sum_j \tau_{jl}^{(V)}}{n_2}$ $\quad$ $\pi_{ql} = \frac{\sum_{ij:X_{ij}=1} \tau_{iq}^{(U)} \tau_{jl}^{(V)}}{\sum_i \tau_{iq}^{(U)} \sum_j \tau_{jl}^{(V)}}$

Inférence du Modèle à blocs stochastiques L'inférence et les astuces calculatoires sont similaires à celles utilisées pour le LBM. Nous ne donnons donc pas ici le détail du critère ou celui de la mise à jour des paramètres du modèle.

4 Expérimentations

Nous présentons les temps de calcul pour réaliser l'inférence du SBM avec des données simulées et réelles. Notre package *sparsebm* offre la possibilité d'utiliser l'accélération

matérielle fournie par un GPU. Le GPU utilisé pour ces expérimentations est un Tesla V100-SXM2-32GB. Le temps d'exécution de l'algorithme correspond à l'exécution de 100 itérations d'EM sur 100 initialisations aléatoires, suivie d'itérations jusqu'à convergence pour les 10 meilleurs résultats à l'issue de ces 100 itérations. La convergence est établie classiquement par la stagnation du critère optimisé.

TABLE 1 – Temps d'exécution du SBM : haut à gauche, pour un nombre de nœuds variable avec un degré moyen de 10 ; haut à droite, pour un nombre de nœuds fixe (10^4) et à degré moyen variable ; bas, pour des jeux de données réels. Le nombre de classes est toujours fixé à 4.

Nombre de nœuds	Temps d'exécution
$1 \cdot 10^3$	14.1 s
$5 \cdot 10^3$	16.6 s
$1 \cdot 10^4$	23.7 s
$5 \cdot 10^4$	1 min 51.3 s
$1 \cdot 10^5$	3 min 50.0 s
$5 \cdot 10^5$	20 min 30.4 s
$1 \cdot 10^6$	44 min 41.1 s
$5 \cdot 10^6$	266 min 42.0 s
avec un degré moyen de 10	

Degré moyen	Temps d'exécution
10	23.7 s
15	31.8 s
20	46.1 s
50	1 min 55.4 s
100	4 min 53.2 s
150	9 min 2.0 s
200	12 min 48.4 s
avec 10^4 nœuds	

Dataset	Nombre de nœuds	Degré moyen	Temps d'exécution
Flickr-medium [a]	$2 \cdot 10^5$	42.4	52 min 47.7 s
LinkedIn	$3 \cdot 10^6$	8.3	340 min 45.0 s

a. Disponible sur `https://www.aminer.cn/data-sna#Flickr-medium`

Références

[1] A. P. DEMPSTER, N. M. LAIRD et D. B. RUBIN : Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society : Series B (Methodological)*, 39(1):1–22, 1977.

[2] G. GOVAERT et M. NADIF : Block clustering with Bernoulli mixture models : Comparison of different approaches. *Computational Statistics & Data Analysis*, 52(6):3233–3245, February 2008.

[3] G. GOVAERT et M. NADIF : *Co-clustering : models, algorithms and applications*. ISTE Ltd, 2013.

[4] P. W. HOLLAND, K. B. LASKEY et S. LEINHARDT : Stochastic blockmodels : First steps. *Social Networks*, 5(2):109 – 137, 1983.

[5] T. S. JAAKKOLA : Tutorial on variational approximation methods. *In Advanced mean field methods : theory and practice*, p. 129–159. MIT Press, 2000.