Deep Multi-CoClustering des données multivues et multi-modales

Reda Khoufache

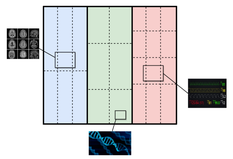
Dirigé par Mustapha Lebbah[1] et Hanene Azzag[2]

1-Université Paris-Saclay, UVSQ, DAVID, Versailles
2-Université Sorbonne Paris Nord, LIPN UMR CNRS 7030

Contexte

Le multi-coclustering est **un algorithme Bayesien non paramétrique** qui combine deux couches :

- la première couche regroupe ensemble certaines variables dans une même vue.
- la deuxième couche qui applique un coclustering pour chaque vue.



Axes de recherches

MCC distribué et fédéré : Développer une version **scalable** du Multi-Coclustering qui puisse être déployée sur des architectures modernes en «***clusters***» (edge computing).

Figure 1. Calcul distribué et fédéré

Deep multi-Coclustering

- Tirer profit des réseaux de **neurones profonds** et leur capacité d'extraction des caractéristiques.
- Proposer un outil unifié qui réalise de façon **jointe l'apprentissage profond** des représentations et la tache du multi-Coclustering.

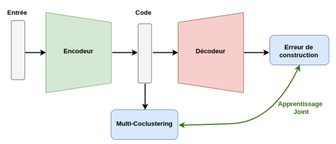


Figure 2. Architecture d'un deep multi-Coclustering basé sur les auto-encodeurs

Multi-Coclustering topologique : Etendre le Multi-CoClustering au modèle qui regroupe les variables partageant la même partition d'observations en utilisant les modèles topologiques à base de modèles de mélange.

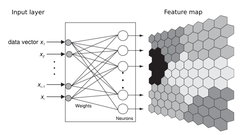


Figure 3. **Carte topologique** de Kohonen (SOM: Self-Organizing Map)

Inférence distribuée pour les DPMM dans l'appressage fédéré

[Khoufache et al. SIAM SDM'2024]

- Dans ce travail, nous présentons une méthode d'inférence distribuée (DisCGS) pour les modèles de mélange de processus de Dirichlet (DPMM).
- Notre approche, basée sur l'échantillonneur de Gibbs qui est une méthode de Monte-Carlo par chaînes Markov (MCMC), est conçue pour être exécutée sur un environnement distribué, notamment dans le contexte de l'apprentissage fédéré horizontal.

Processus de Dirichlet pour les modèles de mélanges

$$
\begin{aligned}
x_i \mid \{z_i = k, \theta_k\} &\overset{\text{i.i.d.}}{\sim} f\left(x_i, \theta_k\right), \forall i \in \{1, \cdots, n\} \\
\theta_k &\overset{\text{i.i.d.}}{\sim} G_0, \forall k \in \{1, 2, \cdots\} \\
z_i &\overset{\text{i.i.d.}}{\sim} \text{Mult}\left(\pi\right), \forall i \in \{1, \cdots, n\} \\
\pi &\sim SB(\alpha),
\end{aligned}
$$

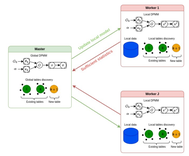


Figure 4. Workflow du DisCGS

- Chaque *worker* est initialisé avec le même modèle global et met à jour son modèle local en utilisant ses données privées.
- Les statistiques suffisantes et les tailles associées à chaque *cluster* local sont calculées et transmises au serveur (*master*).
- Ce dernier procède à la mise à jour du modèle global et à l'estimation de la partition globale sans avoir accès aux données. Le modèle à jour est ensuite partagé avec chaque composant.

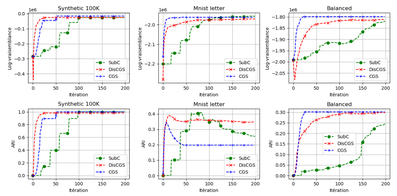


Figure 5. La log-vraisemblance et le score ARI à chaque itération

2*Nombre d'observations	ARI		NMI		ACC		Temps d'exécution	
	Dis.	Cen.	Dis.	Cen.	Dis.	Cen.	Dis.	Cen.
20K	**0.99**	0.89	**0.99**	0.96	**0.99**	0.89	**66.53**	1704.21
40K	0.96	**0.99**	0.97	**0.99**	**0.99**	0.97	**99.45**	10898.28
60K	**0.91**	0.89	0.92	**0.96**	**0.92**	0.89	**135.76**	25738.46
80K	**0.94**	0.89	**0.96**	**0.96**	**0.91**	0.89	**201.59**	27492.08
100K	**0.91**	0.89	**0.94**	0.89	**0.91**	0.89	**207.58**	44688.31

Table 1. Métriques de *clustering* et temps d'exécution obtenus par l'inférence distribuée (Dis.) et centralisée (Cen.)

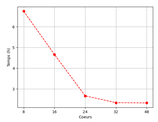


Figure 6. Temps d'exécution (h) en fonction du nombre de cœurs de DisCGS

Inférence MCMC distribuée pour les modèles à blocks latents Bayésiens non paramétriques

[Khoufache et al. PAKDD'2024]

Dans cet article, nous présentons une nouvelle méthode d'inférence par chaîne de MCMC pour les modèles à blocs latents Bayésiens non paramétriques (DisNPLBM), en utilisant l'architecture master/worker.

$$
\begin{aligned}
x_{i,j} \mid \{z_i, w_j, \theta_{z_i,w_j}\} &\sim F\left(\theta_{z_i,w_j}\right), \\
\theta_{z_i,w_j} \sim G_0,\ z_i \mid \pi \sim \text{Mult}(\pi),\ w_j \mid \rho &\sim \text{Mult}(\rho), \\
\pi \sim \text{SB}(\alpha),\ \rho &\sim \text{SB}(\beta).
\end{aligned}
$$

Méthode proposée

- Chaque workers infère une partition locale des lignes sachant la partition globale des colonnes.
- Les statistiques suffisantes associées à chaque groupe de lignes local sont envoyées au master.
- Au niveau du master, la partition globale des lignes est estimée. Ensuite, la partition des colonnes est estimée sachant la partition globale des lignes.

2^*n	ARI		NMI		$\hat{K} \times \hat{L}$		Running time (s)	
	Dis.	Cen.	Dis.	Cen.	Dis.	Cen.	Dis.	Cen.
20K	1.0	1.0	1.0	1.0	30	30	400.21	2265.69
40K	1.0	1.0	1.0	1.0	30	30	693.02	6452.78
60K	1.0	1.0	1.0	1.0	30	30	1122.80	10511.01
80K	1.0	1.0	1.0	1.0	30	30	1373.04	19965.01
100K	1.0	1.0	1.0	1.0	30	30	1572.90	41897.12

Table 2. ARI, NMI et temps d'exécution obtenus par la méthode distribuée (Dis.) et centralisée (Cen).

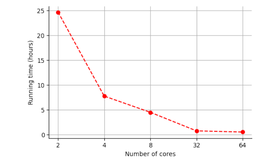


Figure 7. Temps d'exécution (h) en fonction du nombre de cœurs de DisCGS

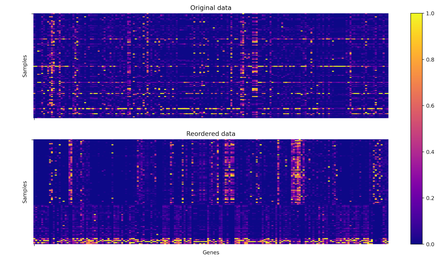


Figure 8. Heatmap sur des données d'expression de gênes.

Références

Reda, K., Anisse, B., Hanene, A., and Mustapha, L. (2024a). Distributed mcmc inference for bayesian non-parametric latent block model. In *The Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*.

Reda, K., Mustapha, L., Hanene, A., Etienne, G., and Djamel, B. (2024b). Distributed collapsed gibbs sampler for dirichlet process mixture models in federated learning. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*.