Apprentissage Joint de la Représentation et du Clustering avec un Réseau Convolutif sur Graphe

Chakib Fettal[*,**], Lazhar Labiod[*], Mohamed Nadif[*]

[*]Université de Paris, CNRS, Centre Borelli UMR9010
[**]Informatique Caisse des Dépôts et Consignations
<prénom.nom>@u-paris.fr

Résumé. Nous proposons un modèle pour l'apprentissage simultané de la représentation et le partitionnement (ou clustering) de graphes attribués. Basé sur un simple réseau convolutif sur graphe, notre modèle effectue le clustering en minimisant la différence entre la représentation réduite des données convoluées et la reconstruction des centroïdes calculés dans l'espace de dimension réduite. Nous montrons l'efficacité du modèle par rapport à l'état de l'art sur différents jeux de données de graphes attribués sur la tâche du clustering. Cet article est un résumé du papier (Fettal et al., 2022).

1 Introduction

L'apprentissage de nouvelles représentations, à la fois fidèles aux données initiales tout en faisant en sorte qu'elles aient une structure favorable au *clustering*, conduit souvent à un meilleur partionnemment profitable à l'utilisateur (Yamamoto et Hwang, 2014; Allab et al., 2016, 2018; Labiod et Nadif, 2021). Dans le contexte des graphes attribués, qui sont des graphes dont les nœuds ont des attributs, cette démarche n'est pas exploitée. Plus encore, les méthodes de partitionnement de graphes attribués souffrent d'une grande complexité spatiale et/ou computationnelle. Contrairement à la plupart des approches existantes, cet article vise à surmonter cette faiblesse en considérant un apprentissage simultané d'un plongement (embedding) et d'un *clustering* de graphe. En effet, en raison de la structure non euclidienne des graphes ainsi que la grande dimension des diverses caractéristiques décrivant les nœuds, la recherche de représentations euclidiennes appropriées qui incorporent l'information sur la structure et les caractéristiques de ces graphes est un défi intéressant pour l'apprentissage automatique (Hamilton et al., 2017). La littérature récente propose d'apprendre ces représentations automatiquement. De telles méthodes d'apprentissage de représentations visent à représenter les nœuds dans un espace de faible dimension où la proximité des représentations des nœuds doit être suffisamment similaire à celle de la représentation originale du graphe. Récemment, les *Réseaux convolutifs sur graphe* ou *Graph Convolutional Networks* (GCNs) (Kipf et Welling, 2017) ont suscité beaucoup d'attention en raison de leur capacité à apprendre des représentations de graphes de haute qualité, et par extension, sont efficaces pour différentes tâches liées aux graphes telles que la classification supervisée de nœuds, la prédiction de liens, et le partitionnement de nœuds qui est la tâche sur laquelle notre article se concentre. Cependant, ce

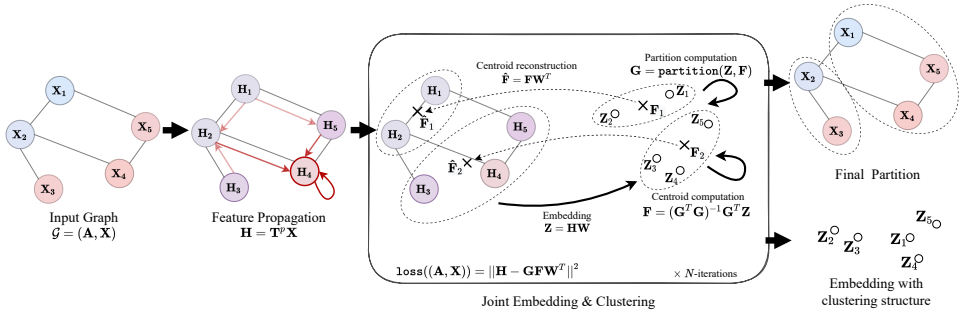


FIG. 1: Schéma du modèle : GCC crée une représentation initiale du graphe avant d'apprendre itérativement une représentation réduite et un clustering des données. Le signal du graphe est représenté par les couleurs des nœuds. La propagation permet d'obtenir un signal plus lisse.

partitionnement de nœuds est généralement effectué en tant que tâche en aval, mais certaines approches basées sur le GCN pour l'embedding et le partitionnement simultanément ont récemment vu le jour (Zhang et al., 2019). Dans notre article, nous proposons de nous appuyer sur le GCN pour développer un nouveau modèle de *Graph Convolutional Clustering* (GCC) qui est capable de prendre en compte les deux tâches simultanément.

2 Embedding et Partitionnement Simultané de Noeuds

Dans cette section, nous décrivons comment nous formulons le problème de l'embedding et du partitionnement simultanés des nœuds d'un graphe attribué. Nous proposons ensuite un nouveau modèle pour le résoudre (illustré dans la figure 1).

Soit $\mathcal{G} = (V, \mathbf{A}, \mathbf{X})$ un graphe non orienté attribué où V représente l'ensemble des sommets constitué des nœuds $\{v_1, \ldots, v_n\}$, $\mathbf{A} \in \mathbb{R}^{n \times n}$ est une matrice d'adjacence où a_{ij} représente le poids de l'arête entre les nœuds v_i et v_j, et $\mathbf{X} \in \mathbb{R}^{n \times d}$ est une matrice de caractéristiques au niveau des nœuds. Dans ce qui suit, k représente le nombre de clusters, f est la dimension de l'embedding, $\mathbf{1}_m$ représente un vecteur colonne de m uns et $\mathbf{I}_m$ une matrice d'identité de dimension m. $\mathbf{M}$ étant une matrice, on note alors $\mathbf{m}_i$ est son i-ème vecteur ligne et m_{ij} est le j-ème élément de la i-ième ligne.

Nous formulons le problème simultané de l'embedding des nœuds et du partitionnement comme suit

$$\min_{\substack{\theta_1,\theta_2,\\ \mathbf{G},\mathbf{F}}} \underbrace{\left|\left|\texttt{dec}_{\theta_2}\Big(\texttt{enc}_{\theta_1}\big(\texttt{agg}(\mathbf{A},\mathbf{X})\big)\Big) - \texttt{agg}(\mathbf{A},\mathbf{X})\right|\right|^2}_{\text{terme de reconstruction}} + \alpha \underbrace{\left|\left|\texttt{enc}_{\theta_1}\big(\texttt{agg}(\mathbf{A},\mathbf{X})\big) - \mathbf{GF}\right|\right|^2}_{\text{terme de régularisation par le clustering}}$$
$$\text{s.t.} \quad \mathbf{G} \in \{0,1\}^{n \times k},\ \mathbf{G}\mathbf{1}_k = \mathbf{1}_n \tag{1}$$

où $\texttt{enc}_{\theta_1}$ est la fonction d'encodage, $\texttt{dec}_{\theta_2}$ est la fonction de décodage, $\texttt{agg}(\mathbf{A}, \mathbf{X})$ est un agrégat de $\mathbf{A}$ et $\mathbf{X}$ qui représente l'information contenue dans le graphe (structure et caractéris-

tiques des noeuds), $\mathbf{G} \in \{0,1\}^{n \times k}$ la matrice de classification binaire, $\mathbf{F} \in \mathbb{R}^{k \times d}$ joue le rôle de centroïdes dans l'espace d'embedding et α est un coefficient qui régule le compromis entre la recherche de reconstruction de la matrice $\texttt{agg}(\mathbf{A}, \mathbf{X})$ et le partitionnement des noeuds.

Le terme de régularisation par le clustering est la fonction coût de k-means Lloyd (1982) sur les observations encodées. Il pénalise les transformations qui n'aboutissent pas à des représentations favorables au clustering.

Dans (Salha et al., 2020), les auteurs proposent un autoencodeur linéaire sur graphe (LGAE) et montrent qu'un encodeur linéaire avec un décodeur à produit interne peut être suffisamment puissant pour atteindre des résultats compétitifs par rapport aux modèles plus complexes basés sur le GCN pour les tâches de prédiction de liens et de partitionnement de nœuds. Par conséquent, nous définissons également notre encodeur comme étant une simple transformation linéaire, c'est-à-dire : $\texttt{enc}(\texttt{agg}(\mathbf{A}, \mathbf{X}); \mathbf{W}_1) = \texttt{agg}(\mathbf{A}, \mathbf{X})\mathbf{W}_1$. Dans LGAE, le décodeur tente de reconstruire la matrice d'adjacence $\mathbf{A}$ plutôt qu'une agrégation de $\mathbf{A}$ et $\mathbf{X}$. Cela signifie que ce type de décodeur n'est pas adapté à notre problème. Par conséquent, nous définissons également le décodeur comme une transformation linéaire $\texttt{dec}(\mathbf{Z}; \mathbf{W}_2) = \mathbf{Z}\mathbf{W}_2$ où $Z = \texttt{agg}(\mathbf{A}, \mathbf{X})\mathbf{W}_1$.

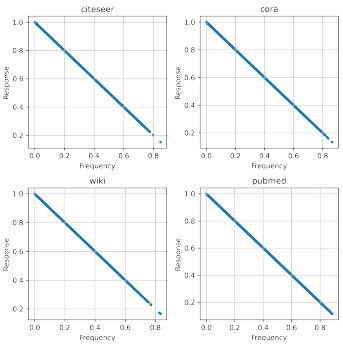


FIG. 2: Réponse en fréquence du filtre GCN proposé en fonction de la fréquence sur quatre datasets.

Notre choix pour la fonction d'agrégation s'inspire de la convolution de graphe simple proposée dans SGC (Wu et al., 2019). Nous définissons $\texttt{agg}(\mathbf{A}, \mathbf{X}) = \mathbf{T}^p\mathbf{X}$ mais plutôt que d'avoir $\mathbf{T}$ comme étant la matrice d'adjacence symétriquement normalisée après ajout de *self loops*, nous la définissons comme suit $\mathbf{T} = {\mathbf{D_T}}^{-1}(\mathbf{I} + \tilde{\mathbf{S}})$ où $\tilde{\mathbf{S}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ avec $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ et $\tilde{\mathbf{D}}$ (resp. $\mathbf{D_T}$) étant la matrice diagonale des degrés de $\tilde{\mathbf{A}}$ (resp. $\mathbf{I} + \tilde{\mathbf{S}}$).

Le GCN, et par extension le SGC, effectue un filtrage du signal graphique avec la matrice $\mathbf{I} - \tilde{\mathbf{S}} = \mathbf{I} - \tilde{\mathbf{D}}^{-1/2}(\mathbf{I} - \tilde{\mathbf{L}})\tilde{\mathbf{D}}^{-1/2}$ où $\tilde{\mathbf{L}}$ est le Laplacien $\tilde{\mathbf{A}}$. La fonction de réponse en fréquence de ce filtre est $h(\tilde{\lambda_l}) = 1 - \tilde{\lambda_l}$ où λ_l est une fréquence du graphe. Dans le GCN, l'empilement de K-couches, ou de manière équivalente l'augmentation de $\tilde{\mathbf{S}}$ à la puissance K dans le SGC, implique de faire le filtrage avec la fonction de réponse en fréquence $h_K(\tilde{\lambda_l}) = (1 - \tilde{\lambda_l})^K$. Ce filtre est passe-bas sur $[0, 1]$ mais pas sur $[0, 1.5]$. Nous proposons ensuite d'ajouter des *self loops* et de normaliser la matrice $\tilde{\mathbf{S}}$. Ceci a les effets suivants : (1) Du point de vue spectral, la normalisation proposée réduit encore le spectre de la matrice pour qu'il se situe dans $[0, 1]$, comme on peut le voir sur la figure 2, ce qui rend le filtre véritablement passe-bas. (2) Du point de vue spatial, chaque sommet transformé devient une moyenne pondérée de ses voisins, ce qui est plus intuitif, mais prend également en compte l'information sur le degré de la colonne, contrairement à la normalisation d'adjacence par marche aléatoire.

Nous motivons davantage ce choix dans la section sur les expériences. Ainsi, avec cette fonction d'agrégation, notre problème se transforme en

$$\begin{aligned} \min_{\mathbf{G},\mathbf{F},\mathbf{W_1},\mathbf{W_2}} & \left|\left|\mathbf{T}^p\mathbf{X} - \mathbf{T}^p\mathbf{X}\mathbf{W_1}\mathbf{W_2}\right|\right|^2 + \alpha\left|\left|\mathbf{T}^p\mathbf{X}\mathbf{W_1} - \mathbf{G}\mathbf{F}\right|\right|^2 \\ \text{s.t.} \quad & \mathbf{G} \in \{0,1\}^{n \times k},\ \mathbf{G}\mathbf{1}_k = \mathbf{1}_n. \end{aligned} \tag{2}$$

Les deux termes de (2) permettent d'exprimer une connexion entre les deux tâches, le premier terme joue le rôle d'auto-encodeur linéaire et le second le rôle de clustering dans l'espace d'embedding. Nous décidons dans la suite de donner le même poids aux deux termes ($\alpha = 1$).

Pour obtenir un renforcement mutuel entre l'embedding et le partitionnement, nous supposons $\mathbf{W} = \mathbf{W_1} = \mathbf{W_2}^\top$ et ajoutons une contrainte d'orthogonalité sur $\mathbf{W}$ dans (2). Cela donne lieu au problème suivant

$$\begin{aligned} \min_{\mathbf{G},\mathbf{F},\mathbf{W}} \quad & \left|\left|\mathbf{T}^p\mathbf{X} - \mathbf{T}^p\mathbf{X}\mathbf{W}\mathbf{W}^\top\right|\right|^2 + \left|\left|\mathbf{T}^p\mathbf{X}\mathbf{W} - \mathbf{G}\mathbf{F}\right|\right|^2 \\ \text{s.t.} \quad & \mathbf{G} \in \{0,1\}^{n\times k},\ \mathbf{G}\mathbf{1}_k = \mathbf{1}_n,\ \mathbf{W}^\top\mathbf{W} = \mathbf{I}_k. \end{aligned} \tag{3}$$

Notons que la résolution de ce problème peut être prouvée comme étant équivalente à

$$\min_{\mathbf{G},\mathbf{F},\mathbf{W}} \quad \left|\left|\mathbf{T}^p\mathbf{X} - \mathbf{G}\mathbf{F}\mathbf{W}^\top\right|\right|^2 \text{ s.t. } \quad \mathbf{G} \in \{0,1\}^{n\times k},\ \mathbf{G}\mathbf{1}_k = \mathbf{1}_n,\ \mathbf{W}^\top\mathbf{W} = \mathbf{I}_k. \tag{4}$$

2.1 Procédure d'Optimisation

Pour chaque matrice, en fixant les deux autres matrices, on obtient une formule qui peut être résolue directement. Les solutions à ces problèmes modifiés diminuent la fonction de coût globale de façon monotone. Les règles d'initialisation et de mise à jour sont décrites dans ce qui suit.

Initialisation. Nous initialisons $\mathbf{W}$ avec les f premières composantes obtenues en appliquant une *Analyse en Composantes Principales* (ACP) randomisée sur $\mathbf{T}^p\mathbf{X}$. Les matrices $\mathbf{F}$ et $\mathbf{G}$ sont ensuite obtenues par k-means sur $\mathbf{T}^p\mathbf{X}\mathbf{W}$.

Règle de mise à jour pour F. En fixant $\mathbf{G}$ et $\mathbf{W}$, on cherche à calculer $\mathbf{F}$. On aboutit à un problème linéaire des moindres carrés ; on obtient la solution optimale au problème donné. La règle de mise à jour est alors

$$\mathbf{F} = (\mathbf{G}^\top\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{T}^p\mathbf{X}\mathbf{W}. \tag{5}$$

Intuitivement, chaque vecteur ligne $\mathbf{f}_i$ est défini comme la moyenne des embeddings $\mathbf{T}^p\mathbf{X}\mathbf{W}$ qui sont assignés au cluster i.

Règle de mise à jour pour W. En fixant $\mathbf{G}$ et $\mathbf{F}$ on cherche à calculer $\mathbf{W}$. La règle de mise à jour est la suivante

$$\mathbf{W} = \mathbf{U}\mathbf{V}^\top \quad \text{s.t.} \quad [\mathbf{U},\mathbf{\Sigma},\mathbf{V}] \quad = \texttt{SVD}\big((\mathbf{T}^p\mathbf{X})^\top\mathbf{G}\mathbf{F}\big) \tag{6}$$

où $\mathbf{\Sigma} = (\sigma_{ii})$, $\mathbf{U}$, et $\mathbf{V}$ sont respectivement les valeurs singulières, les vecteurs singuliers (gauche et droit) de la matrice $(\mathbf{T}^p\mathbf{X})^\top\mathbf{G}\mathbf{F}$.

Règle de mise à jour pour G. En fixant $\mathbf{F}$ et $\mathbf{W}$ et en cherchant à calculer $\mathbf{F}$, nous obtenons un problème qui peut être optimisé avec l'étape d'affectation de l'algorithme des k-means. La règle de mise à jour est alors donnée par

$$g_{ij^*} \leftarrow \begin{cases} 1 & \text{if } j^* = \arg\min_j \left|\left|(\mathbf{T}^p\mathbf{X}\mathbf{W})_i - \mathbf{f}_j\right|\right|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Les étapes de GCC sont décrites dans l'Algorithme 1 et illustré dans la figure 1. La convergence de GCC est garantie, mais il n'atteindra qu'un optimum local en fonction des conditions initiales. Une stratégie possible pour surmonter ce problème consiste à exécuter plusieurs fois l'algorithme GCC et à sélectionner le meilleur résultat par rapport à la fonction objectif. La complexité est brièvement décrite dans Table 1.

Algorithme 1 : GCC

Entrées : Matrice d'adjacence $\mathbf{A}$, Matrice des caractéristiques $\mathbf{X}$, Ordre de propagation p, Nombre de clusters k, Dimension d'embedding f, Tolérance ϵ, Nombre maximal d'itérations max_iter
Sorties : Indicateur d'appartenance $\mathbf{G} \in \{0,1\}^{n\times k}$, centres intégrés $\mathbf{F} \in \mathbb{R}^{k\times f}$, matrice d'embedding $\mathbf{W} \in \mathbb{R}^{d\times f}$
Initialiser $\mathbf{W}$ avec une ACP randomisée sur $\mathbf{T}^p\mathbf{X}$;
Initialiser $\mathbf{G}$ avec un k-means sur $\mathbf{T}^p\mathbf{X}\mathbf{W}$;
while $||\mathbf{T}^p\mathbf{X} - \mathbf{G}\mathbf{F}\mathbf{W}^\top|| > \epsilon$ **or** $\max\_\text{iter}$ *non atteint* **do**
    Mise à jour de $\mathbf{F}$ en utilisant la formule (5);
    Mise à jour de $\mathbf{W}$ en utilisant la formule (6);
    Mise à jour de $\mathbf{G}$ en utilisant la formule (7);
end

TAB. 1: Etude de la complexité de l'algorithme. Pour le calcul de la complexité global nous supposons que $t' = t$ et $k, f \leq \min(n, d)$.

Etape	Complexité		Etape	Complexité
Calcul $\mathtt{agg}(\mathbf{A}, \mathbf{X})$	$O(p\|E\|d)$		**MAJ de** $\mathbf{G}$	$O(nkf)$
Init. de $\mathbf{W}$ **et** $\mathbf{G}$	$O\left(n(d\log(k) + df + tkf)\right)$		**Calcul de coût**	$O(dkf + nd)$ ou $O(ndf)$
MAJ de $\mathbf{F}$	$O(ndf)$		**Complexité totale**	$O(p\|E\|d + tndk)$
MAJ de $\mathbf{W}$	$O(ndk)$			

2.2 Sélection de l'ordre de propagation *p*

La sélection de l'ordre de propagation p fait partie intégrante de la performance de l'algorithme. Un ordre de propagation p plus petit peut signifier que l'information de voisinage propagée est insuffisante, tandis qu'un ordre de propagation p plus grand peut entraîner un lissage excessif du signal du graphe. Avec l'AGC (Zhang et al., 2019), les auteurs ont proposé de sélectionner d'abord un intervalle de valeurs possibles pour p, puis de retenir le premier p qui est un minimum local d'une métrique intra-groupe. Puisque notre fonction de perte contient des informations sur la performance du clustering, elle peut servir de métrique pour la sélection de p. Ainsi, de manière similaire à l'AGC, nous sélectionnons p via notre fonction de perte comme suit : Nous nous arrêtons et sélectionons $p = p^*$ si la variation de la racine carrée de la fonction de perte $||\mathbf{T}^p\mathbf{X} - \mathbf{G}\mathbf{F}\mathbf{W}^\top||$ entre p^* et $p^* - 1$ est inférieure à $\frac{d}{n}$ pour $p \in \{0, \ldots, 150\}$. La figure 4 montre comment notre matrice de propagation se compare aux normalisations classiques.

3 Expériences

Pour évaluer le modèle proposé, nous menons des expériences sur quatre datasets et le comparons à un certain nombre d'approches de partitionnement de nœuds.

TAB. 2: Performances de partitionnement sur quatre ensembles de données, calculées comme la moyenne sur 20 cycles. La moyenne d'AGE a été calculée sur 3 exécutions. AGE, LAE et LVAE n'ont pas réussi à s'adapter à Pubmed ; OOM signifie "out of memory".

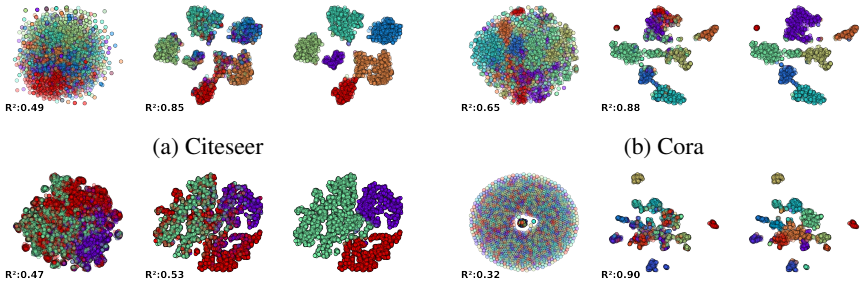
Method	Input	Citeseer			Cora			Pubmed			Wiki		
		Acc	F1	NMI	Acc	F1	NMI	Acc	F1	NMI	Acc	F1	NMI
Sph. k-means	$\mathbf{X}$	42.64	40.16	19.91	33.97	30.93	15.33	59.51	58.16	31.26	33.65	23.30	29.90
DCN	$\mathbf{X}$	19.16	11.44	2.91	20.01	11.81	2.32	15.87	7.06	4.07	44.28	17.14	12.45
Spectral	$\mathbf{A}$	21.60	9.46	1.54	30.00	8.78	2.36	58.96	43.53	18.30	23.20	13.74	18.05
LAE (2020)	$(\mathbf{A}, \mathbf{X})$	43.49	41.33	22.66	65.43	66.21	48.89		OOM		45.26	40.90	45.99
LVAE (2020)	$(\mathbf{A}, \mathbf{X})$	39.46	38.26	20.53	64.11	65.31	48.47		OOM		47.38	42.92	47.79
AGE (2020)	$(\mathbf{A}, \mathbf{X})$	57.85	55.01	35.74	69.17	67.30	56.91		OOM		53.79	41.39	52.63
GIC (2021)	$(\mathbf{A}, \mathbf{X})$	68.78	64.02	43.82	70.45	68.95	52.55	64.30	64.86	26.02	46.46	40.29	48.24
S²GC (2021)	$(\mathbf{A}, \mathbf{X})$	68.13	63.79	42.26	69.68	66.41	54.83	70.81	**69.96**	**32.32**	52.71	44.40	48.96
GCC (ours)	$(\mathbf{A}, \mathbf{X})$	**69.45**	**64.54**	**45.13**	**74.29**	**70.35**	**59.17**	**70.82**	69.89	32.30	**54.56**	**46.10**	**54.61**

Nous comparons notre algorithme avec des méthodes utilisant uniquement des caractéristiques des nœuds, k-means sphérique (Hornik et al., 2012) servira de référence avec le DCN (Yang et al., 2017), des méthodes utilisant uniquement la structure du graphe. Spectral dénote l'algorithme de clustering spectral classique. Enfin des Méthodes utilisant les deux, LVAE (Salha et al., 2020) est l'autoencodeur variationnel et linéaire sur graphe et LAE est sa version non probabiliste. GIC (Mavromatis et Karypis, 2021), AGE (Cui et al., 2020) propose un filtre de lissage laplacien qui agit comme un filtre passe-bas appliqué dans un schéma d'apprentissage adaptatif. S²GC (Zhu et Koniusz, 2021) propose, par contre, une nouvelle méthode pour l'agrégation des voisinages à K sauts. Les performances de partionnement des différentes méthodes sont indiquées dans le tableau 2. Les meilleures performances sont indiquées en gras. Il est clair que les méthodes qui utilisent à la fois $\mathbf{A}$ et $\mathbf{X}$ ont des performances significativement meilleures que celles qui utilisent l'un ou l'autre individuellement. Nous observons également que GCC surpasse les autres méthodes de partitionnement de graphes attribués sauf Pubmed où S²GC et GCC sont comparables. Nous rapportons également le temps d'exécution moyen de chaque algorithme dans le tableau 3.

TAB. 3: Temps d'horloge en secondes pour les différents algorithmes, moyenné sur 20 exécutions (3 pour AGE).

Method	CiteSeer	Cora	Pubmed	Wiki
Sph. km	18.1	3.2	8.3	20.2
LAE	12.3	8.9	OOM	27.3
LVAE	11.9	6.3	OOM	29.3
AGE	2461	936.3	OOM	3058.7
GIC	8.4	5.7	13.9	8.3
S²GC	7.7	1.0	24.8	6.1
GCC	2.5	1.0	11.8	2.9

La figure 3 présente les représentations, en basse dimension produites par notre modèle, projetées dans un espace à 2 dimensions par t-SNE. Nous pouvons voir une nette différence dans les structures des projections des données brutes et celles des embeddings. Pour évaluer la qualité de l'embedding, nous utilisons la mesure R-carré, i.e., $R^2 = \frac{\mathrm{Tr}(S_b)}{\mathrm{Tr}(S_t)}$, où $\mathrm{Tr}(S_b)$ est l'inertie inter classe et $\mathrm{Tr}(S_t)$ est l'inertie totale. Nous reportons cette mesure dans la figure 3 sur les graphiques des vraies étiquettes pour quantifier la séparabilité. Le R-carré est plus grand pour les projections bidimensionnelles des embeddings GCC sur les quatre ensembles de données. Sur Pubmed, la structure est moins prononcée (0,47 vs 0,53) mais nous pouvons toujours voir la formation de trois clusters.



(a) Citeseer (b) Cora

(c) Pubmed (d) Wiki

FIG. 3: *Colonne gauche :* Projection des données originales colorées en fonction des étiquettes réelles. *Colonne centrale :* Projection de l'embedding de GCC coloré en fonction des étiquettes réelles. *Colonne droite :* Projection de l'embedding de GCC coloré en fonction des étiquettes prédites. Le R-carré est utilisé pour mesurer la séparabilité des classes réelles.

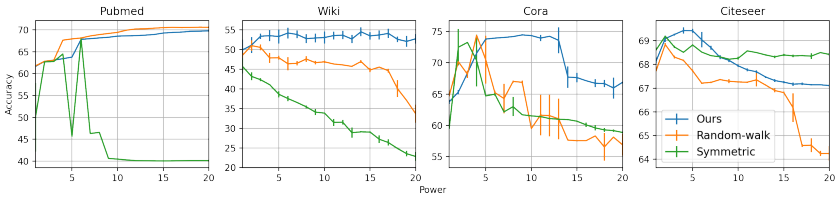


FIG. 4: Accuracy de GCC avec différentes matrices de propagation moyenné sur 20 exécutions

4 Conclusion

Dans cet article, nous avons exploité la formulation simple du réseau convolutif sur graphe pour obtenir un modèle efficace qui traite de façon simultanée l'apprentissage des deux tâches dans un cadre unifié ; l'embedding des nœuds et le clustering. Premièrement, nous avons fourni une normalisation qui fait que l'encodeur GCN agit comme un filtre passe-bas au sens strict. Deuxièmement, nous avons proposé une nouvelle approche dans laquelle la fonction objective à optimiser exploite les informations provenant à la fois du coût de reconstruction de l'embedding GCN et de la structure de clustering de ces embeddings. Troisièmement, nous avons dérivé GCC dont la complexité a été rigoureusement étudiée. Ce faisant, nous avons montré comment GCC atteint de meilleures performances par rapport à d'autres algorithmes de partitionnement de graphes.

Remerciements. Ce travail a été soutenu par une subvention de projet émergence 2021 IDEX-Université de Paris (Spectrans).

Références

Allab, K., L. Labiod, et M. Nadif (2016). A semi-NMF-PCA unified framework for data clustering. *IEEE Transactions on Knowledge and Data Engineering 29*(1), 2–16.

Allab, K., L. Labiod, et M. Nadif (2018). Simultaneous spectral data embedding and clustering. *IEEE transactions on neural networks and learning systems 29*(12), 6396–6401.

Cui, G., J. Zhou, C. Yang, et Z. Liu (2020). Adaptive graph encoder for attributed graph embedding. In *SIGKDD*, pp. 976–985.

Fettal, C., L. Labiod, et M. Nadif (2022). Efficient graph convolution for joint node representation learning and clustering. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*.

Hamilton, W. L., R. Ying, et J. Leskovec (2017). Representation learning on graphs : Methods and applications. *IEEE Data Eng. Bull. 40*(3), 52–74.

Hornik, K., I. Feinerer, M. Kober, et C. Buchta (2012). Spherical k-means clustering. *Journal of Statistical Software, Articles 50*(10), 1–22.

Kipf, T. N. et M. Welling (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Labiod, L. et M. Nadif (2021). Efficient regularized spectral data embedding. *Advances in Data Analysis and Classification 15*(1), 99–119.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Trans. Inf. Theory 28*, 129–136.

Mavromatis, C. et G. Karypis (2021). Graph infoclust : Maximizing coarse-grain mutual information in graphs. In *PAKDD (1)*, pp. 541–553.

Salha, G., R. Hennequin, et M. Vazirgiannis (2020). Simple and effective graph autoencoders with one-hop linear models. In *ECML-PKDD*, pp. 319–334.

Wu, F., A. Souza, T. Zhang, C. Fifty, T. Yu, et K. Weinberger (2019). Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871.

Yamamoto, M. et H. Hwang (2014). A general formulation of cluster analysis with dimension reduction and subspace separation. *Behaviormetrika 41*(1), 115–129.

Yang, B., X. Fu, N. D. Sidiropoulos, et M. Hong (2017). Towards k-means-friendly spaces : Simultaneous deep learning and clustering. In *ICML*, Volume 70, pp. 3861–3870.

Zhang, X., H. Liu, Q. Li, et X.-M. Wu (2019). Attributed graph clustering via adaptive graph convolution. In *IJCAI*, pp. 4327–4333.

Zhu, H. et P. Koniusz (2021). Simple spectral graph convolution. In *9th International Conference on Learning Representations, ICLR, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Summary

We propose a model for joint representation learning and clustering of attributed graphs. Based on the simple graph convolutional network, our model performs clustering by minimizing the difference between the low representaion space of the convolved data and the reconstruction of the centroids in the embedding space. The experiments show the effectiveness of the derived model against state-of-the-art methods on different attributed graph datasets for both clustering and visualization purposes (Fettal et al., 2022).