Contributions to scalable clustering of networks and graphs

Chakib Fettal

Université Paris Cité

Ecole Doctorale Informatique, Télécommunications et Electronique de Paris (ED 130)

Centre Borelli UMR 9010

Contributions to Scalable Clustering of Networks and Graphs

Par Chakib Fettal

Thèse de Doctorat en Informatique
Spécialisation en Science des Données

Dirigée par Mohamed Nadif
Présentée et soutenue publiquement le 02 Février 2024

Devant un jury composé de :

Pr. Sébastien Adam	Université de Rouen	Rapporteur
Pr. Philippe Lenca	IMT Atlantique	Rapporteur
Pr. Christophe Marsala	Sorbonne Université	Examinateur
Pr. Ndeye Niang-Keita	Cnam	Examinatrice
Pr. Mohamed Nadif	Université Paris Cité	Directeur de Thèse
Dr. Lazhar Labiod	Université Paris Cité	Encadrant
M. Marc Gnanou	Informatique CDC	Encadrant Industriel

Remerciements

Tout d'abord, je tiens à remercier mon directeur de thèse, Mohamed Nadif, pour son soutien tout au long de cette thèse ainsi que pour sa patience et pour avoir partagé son savoir avec moi. Je souhaite aussi remercier mon responsable industriel, Marc Gnanou, de m'avoir donné l'opportunité de faire cette thèse et pour avoir créé un environnement de travail positif et encourageant. Je tiens également à exprimer ma gratitude envers mon encadrant, Lazhar Labiod, pour avoir guidé mes efforts avec sagesse, foi et patience. Mes sincères remerciements vont également aux rapporteurs, Pr. Sébastien Adam et Pr. Philippe Lenca ; et aux examinateurs, Pr. Christophe Marsala et Pr. Ndeye Niang-Keita, qui ont généreusement consacré leur temps et leur expertise à évaluer ce travail.

Je souhaite aussi remercier toutes les personnes que j'ai côtoyées durant ces quelques années à Paris Cité et à ICDC. Je pense notamment à Serkan, Salimata, Hugo, Axel, Oualid, Amy, Willie, Mira, Amine, Stéphane, Abdou, Karine, Joel, Abou-Oumar, Laurent, Hakim, Sami, Sylvie, Clément, Matar, Gauthier, Amira et j'en passe. Je pense aussi à Nazim et Mathieu avec qui j'avais effectué mes premiers pas dans la recherche pendant notre stage à Paris Cité. Cette thèse étant financée par ICDC et l'ANRT, je tiens donc à les remercier pour avoir rendu ce travail possible. Enfin, un grand merci à toutes les personnes, qu'elles aient participé de près ou de loin à la réalisation de ce manuscrit.

I would like to dedicate this thesis to my family. To my dear mother and father to whom I owe everything, as well as, to my brother and sister. My gratefulness also goes to the rest of my family for their help during this journey. A special thanks goes to my aunts and uncles (and Fadila) without whom I would not be in a position to fulfill this work. I wish to also dedicate it to my friends; thank you for making these past three years easier to handle: Abdelkader, Amine, Mohamed, Oussama B., Anis, Hichem, Hamza, Abdessamad, Abdrahmane, Oussama M., Moutia, Rym, Rabah and the others.

Résumé

Les graphes sont une structure de données importante utilisée dans de nombreux domaines car ils constituent un outil puissant pour la modélisation et l'analyse de systèmes complexes. Ils sont utilisés pour représenter les relations entre les entités, comme les individus dans un réseau social ou les nœuds dans un réseau informatique. Les graphes ont été utilisés dans diverses applications dans différents domaines, tels que l'analyse des réseaux sociaux, la bioinformatique, l'épidémiologie et bien d'autres encore. Dans l'analyse des réseaux sociaux, par exemple, les graphes peuvent être utilisés pour étudier les modèles d'interactions entre les individus dans un réseau social et identifier les groupes d'individus ayant des intérêts ou des comportements similaires. Cela peut être utile pour le marketing ou les recommandations ciblées.

Le partitionnement de graphes, également connu sous le nom de détection de communautés, est une technique importante dans l'analyse des données de graphes. Il permet d'identifier des groupes de nœuds similaires dans le graphe. Cela peut révéler des motifs et des structures sous-jacents dans le graphe qui ne sont pas immédiatement apparents. Par exemple, dans un réseau social, le clustering peut révéler des groupes d'individus ayant des intérêts ou des comportements similaires, et en bioinformatique, il peut révéler des modules fonctionnels dans les réseaux d'interaction protéine-protéine.

Cette thèse tente de résoudre les problèmes de scalabilité des modèles de clustering de graphes de l'état de l'art et présente des approches pour le clustering et l'apprentissage de représentation de différents types de graphes, y compris les graphes classiques, les graphes bipartis, les graphes attribués, les graphes attribués bipartis et les graphes attribués multi-vues. À cette fin, nous exploitons des techniques simples telles que: les projections linéaires, le lissage laplacien, le transport optimal, etc. Les approches proposées partagent toutes trois caractéristiques clés: simplicité, efficacité et peu d'hyperparamètres. Grâce à leur nature simple mais efficace, les méthodes proposées sont compétitives par rapport à l'état de l'art tout en étant généralement plus efficaces en termes de calcul. Nous démontrons l'efficacité et l'efficience de nos modèles par rapport à l'état de l'art par le biais d'une expérimentation approfondie et de tests de Significativité statistique.

Mots clés : graphes, partitionnement, apprentissage de représentations

Abstract

Graphs are an important data structure used in many fields because they provide a powerful tool for modeling and analyzing complex systems. They are used to represent relationships between entities, such as individuals in a social network or nodes in a computer network. Graphs have been used in various applications across different fields, such as social network analysis, bioinformatics, computer science, transportation, epidemiology and many more. In social network analysis, for example, graphs can be used to study patterns of interactions between individuals in a social network and identify groups of individuals with similar interests or behaviors. This can be useful for targeted marketing or recommendations. In bioinformatics, graphs can be used to identify functional modules in protein-protein interaction networks.

Graph clustering, also known as community detection, is an important technique in the analysis of graph data. Clustering allows for the identification of groups of similar nodes within the graph. This can reveal underlying patterns and structures in the graph that may not be immediately apparent. For example, in a social network, clustering can reveal groups of individuals with similar interests or behaviors, and in bioinformatics, clustering can reveal functional modules in protein-protein interaction networks.

The thesis tries to address scalability issues of the state-of-the-art graph clustering models and presents approaches for clustering and representation learning different types of graphs, including classical graphs, bipartite graphs, attributed graphs, bipartite attributed graphs, and multi-view attributed graphs. To this end we leverage techniques such as: linear projections, Laplacian smoothing, optimal transport, etc. The proposed approaches all share three key characteristics: simplicity, cost-effectiveness, and having few hyper-parameters. Thanks to their simple yet effective nature, the proposed methods are competitive with the state of the art while also generally being more computationally efficient. We showcase the efficacy and efficiency of our models against state-of-the-art methods through extensive experimentation and significance testing.

Keywords : graphs, clustering, representation learning

Résumé Substantiel

Le *Compte Personnel de Formation* (également appelé *Mon Compte Formation* [1]) est un système de financement public français pour les programmes de formation. La plateforme a été développée à l'origine par le Groupe Caisse des Dépôts. En raison de son importance, elle est toujours l'un des projets phares pilotés par la CDC et, par extension, par Informatique CDC (ICDC). La plateforme contient des informations sur les utilisateurs ainsi que sur les programmes de formation. Ces données peuvent être naturellement représentées sous forme de graphes basés sur les similarités qui existent entre les différents utilisateurs et les différents programmes de formation. Ainsi, l'investigation de techniques d'exploration de données et d'apprentissage automatique liées aux graphes peut s'avérer pertinente dans le contexte de ICDC.

L'exploration et l'apprentissage sur les graphes est un domaine du machine learning qui est en plein essor. Il traite de l'analyse et de la compréhension des données représentées sous forme de graphes. Ce domaine a fait l'objet d'une attention particulière ces dernières années en raison de la prolifération des données structurées sous forme de graphes dans divers domaines, notamment les réseaux sociaux, les réseaux biologiques, le World Wide Web, etc. La capacité à extraire des informations et des connaissances utiles de ce type de données est devenue essentielle dans de nombreux domaines de la recherche et de l'industrie. L'objectif de l'exploration et de l'apprentissage sur les graphes est de développer des algorithmes efficaces et efficients pour découvrir des modèles, des relations et des structures latentes dans les données structurées par des graphes. Le but ultime est de fournir des informations qui peuvent être utilisées pour améliorer la prise de décision et soutenir diverses applications telles que les systèmes de recommandation, la détection des fraudes et la bio-informatique, entre autres. Cependant, comme le volume de données structurées en graphes continue de croître, le besoin de techniques efficientes d'exploration de graphes devient encore plus pressant, car plusieurs problèmes peuvent survenir en raison de la grande échelle des données dans les approches d'apprentissage automatique des graphes comme :

- L'absence de données étiquetées: Les graphes contiennent générale-ment moins de données étiquetées que les autres types de données, ce qui complique l'apprentissage des algorithmes d'apprentissage automatique basés sur les graphes.

[1] https://www.moncompteformation.gouv.fr/

- Sparsité des données: Les graphes peuvent être très éparses, ce qui signifie que la plupart des nœuds n'ont que quelques arêtes. Cette sparsité peut rendre difficile l'apprentissage de représentations de qualité à partir du graphe ou la réalisation de prédictions précises, et plus le graphe est grand, plus ce problème est prononcé.

- Complexité spatiale: Bien que les graphes soient généralement peu denses, certaines techniques d'exploration de graphes surmontent cette sparsité en appliquant des transformations à la matrice d'adjacence du graphe, ce qui peut entraîner des problèmes de perte de mémoire.

- Complexité de calcul: Certains algorithmes d'apprentissage automatique basés sur les graphes, tels que ceux qui impliquent la factorisation des matrices, peuvent avoir une complexité temporelle élevée, ce qui rend leurs exécutions sur des graphes à grande échelle peu pratiques.

Il est possible d'atténuer ces problèmes de différentes manières. En cas de manque de données étiquetées, il est possible, par exemple, d'envisager le contexte de l'apprentissage non supervisé, qui est un type d'apprentissage automatique dans lequel le modèle apprend à partir des données sans utiliser d'exemples étiquetés. Quant aux problèmes de données éparses, de complexité spatiale et de calcul, ils peuvent être atténués en les prenant en considération lors de l'élaboration de solutions aux problèmes liés aux graphes.

Dans cette thèse, nous abordons ces questions de complexité et proposons de nouvelles techniques pour les traiter efficacement dans un contexte non supervisé, en particulier celui du clustering (et de l'embedding dans une certaine mesure) en ce qui concerne différents types de graphes. Le partitionnement non supervisé des données et des graphes en particulier est depuis longtemps un sujet d'intérêt dans la communauté de l'exploration des données. Le clustering de nœuds, également connu sous le nom de détection de communautés, est un objectif récurrent dans l'analyse des graphes car il permet d'identifier des groupes de nœuds similaires dans les réseaux. Cette technique peut révéler des modèles et des structures sous-jacentes qui ne sont pas toujours évidentes. L'analyse des réseaux sociaux est un cas d'utilisation clé du clustering de graphe, qui permet d'identifier des groupes d'individus ayant des intérêts ou des comportements similaires [Handcock, 2007; Mishra, 2007]. Cela peut être utile pour le marketing ciblé ou les recommandations [He, 2010]. Un autre cas d'utilisation est la bio-informatique, où le clustering peut être utilisé pour identifier des modules fonctionnels dans les réseaux d'interaction protéine-protéine [Brohee, 2006; Dittrich, 2008; Pizzuti, 2014]. Depuis les premiers jours de la classification automatique (non supervisée), un très grand nombre d'approches ont été proposées dont certaines vont être privilégiées dans cette thèse.

Avec l'avènement de l'apprentissage profond et son succès dans le cadre supervisé, les chercheurs ont essayé de reproduire ce succès dans un contexte non supervisé, tel que le contexte du clustering de graphes. Cela a conduit, cependant, à une crise de reproductibilité en raison de l'utilisation d'un grand nombre d'hyper-paramètres spécifiques aux données en utilisant souvent à tort les labels disponibles pour évaluer les algorithmes. Ce qui est particulièrement le cas dans le deep subspace clustering [Haeffele, 2021] par exemple. Notons

toutefois que cela peut être surmonté par des approches de type *ensemble* comme celle réalisée dans un cadre totalement non supervisé par [Affeldt, 2020; Affeldt, 2022].

Par sa simplicité et son efficience l'algorithme k-means reste l'algorithme de clustering le plus populaire même s'il nécessite la connaissance du nombre de clusters. A noter que la simplicité de k-means entraîne cependant certaines limitations, notamment dans la manière dont il traite les clusters non sphériques de tailles différentes et pas assez bien séparées. Cela s'explique par le fait que le critère optimisé est associé à un modèle de mélange gaussien sphérique contraint. Le spectral clustering [Von Luxburg, 2007] et les algorithmes de type Expectation-Maximization (EM) [Dempster, 1977], par exemple, remédient à ces faiblesses. Ainsi, plusieurs approches seront proposées dans cette thèse, qui sont simples par nature mais qui visent également à surmonter les difficultés liées au clustering de différents types de données, tels que les réseaux attribués, les réseaux à vues multiples, etc.

Dans cette thèse, nous proposons des approches pour les deux tâches clustering et embedding de différents types de graphes, à savoir les graphes classiques, les graphes bipartis, les graphes attribués, les graphes attribués bipartis et les graphes attribués multi-vues. Les approches que nous avons développées partagent toutes trois caractéristiques clés qui, selon nous, contribuent à la grande popularité de l'algorithme de clustering k-means. Ces trois caractéristiques sont les suivantes :

- Simplicité: Dans le sens où il n'y a pas de grands ensembles de paramètres à apprendre comme c'est traditionnellement le cas dans l'apprentissage profond. La plupart des modèles présentés dans cette thèse utilisent des transformations dont le coût est linéaire par rapport à la taille de l'entrée pour apprendre les représentations, et des règles de clustering simples pour générer des partitions.

- La nature rentable de l'algorithme: Les modèles proposés ici sont également rentables par nature c'est à dire qu'il donnent de bons résultats par rapport à leurs temps d'exécution, cela peut être compris dans le sens de l'optimalité de Pareto, ce qui signifie qu'ils surpassent les modèles de l'état de l'art utilisés dans nos benchmarks en termes de complexité de calcul, de temps d'exécution empirique ou de performance de clustering empirique.

- Peu d'hyper-paramètres: Pour tous les modèles que nous avons introduits, peu d'hyperparamètres doivent être définis. Pour ce faire, nous proposons soit des valeurs par défaut testées, soit des règles de sélection des hyperparamètres.

Ce document est organisé comme suit : Dans la Partie 1, nous présentons l'état de l'art, en particulier, dans le chapitre 1, nous introduisons les réseaux et les notions liées aux graphes. Ensuite, dans le chapitre 2, nous fournissons une introduction au clustering avec une attention particulière portée sur le clustering des graphes. Dans Partie 2, nous présentons nos contributions en détail :

Dans le Chapitre 3, nous avons présenté un nouvel algorithme pour le clustering de graphes avec des contraintes de taille arbitraires grâce à l'utilisation d'un transport optimal. Cette

approche généralise le concept de coupes normalisées et de coupes ratio à toute notion de taille et de distributions de taille. L'algorithme proposé s'est avéré efficace lorsqu'il est utilisé comme étape de post-traitement en conjonction avec les algorithmes classiques de coupure de graphe, comme l'ont démontré des expériences sur des ensembles de données équilibrés et déséquilibrés. Les résultats ont mis en évidence l'efficacité de notre approche en termes de performances de clustering et sa capacité à récupérer des partitions qui correspondent étroitement à la distribution souhaitée.

Dans le Chapitre 4, nous avons présenté une nouvelle approche pour le clsutering de graphes bipartis à l'aide de la théorie du transport qui répond aux défis liés au clsutering de données éparses telles que les matrices documents-termes. Le problème est formulé sous la forme d'un programme bilinéaire qui est résolu par un algorithme efficace de descente de coordonnées par blocs. Les résultats des expériences menées sur divers ensembles de données documents-termes indiquent que la méthode proposée identifie efficacement et simultanément les classes de documents et les classes de termes pertinentes sémantiquement. En outre, la méthode proposée est plus performante que les techniques récentes de co-clustering basées sur le transport optimal et est plus efficace sur le plan computationnel. Cela a conduit à une publication [Fettal, 2022b] ainsi qu'à une version française publiée [Fettal, 2023a].

Dans le travail discuté dans le Chapitre 5, nous avons présenté un algorithme efficace pour le clustering de graphes attribués par le biais du processus de clustering de sous-espace. Nous avons proposé un algorithme efficace pour le clustering de graphes attribués par le biais du processus de subspace clustering. L'algorithme commence par apprendre une représentation initiale du graphe à l'aide d'une étape de propagation du voisinage simple mais efficace. Ensuite, il apprend une matrice de coefficients factorisée à l'aide de contraintes d'orthogonalité, qui est ensuite intégrée dans un nouvel espace de caractéristiques pour créer une matrice d'affinité symétrique et non négative. Cette matrice d'affinité est ensuite utilisée dans un algorithme de clustering spectral implicite. L'expérimentation menée a montré que notre proposition est efficace et efficiente par rapport aux algorithmes de clustering de graphes attribués actuellement en vigueur. Ce travail a été publié dans [Fettal, 2023b].

Dans le chapitre 6, nous avons montré l'efficacité de l'utilisation de la formulation simple du GCN pour l'embedding et le clustering efficaces des nœuds. Nous avons proposé une normalisation qui fait de l'encodeur codeur GCN un filtre passe-bas, une nouvelle approche qui exploite les informations provenant à la fois de la perte de reconstruction de l'embedding GCN et de la structure en clusters des embeddings, ainsi qu'un algorithme dont nous avons rigoureusement étudié la complexité et montré qu'il était plus performant que les autres algorithmes de clustering de graphes. Nous avons également étudié rigoureusement la complexité d'un nouvel algorithme et montré qu'il était plus efficace que d'autres algorithmes de clustering de graphes. Les résultats expérimentaux ont fourni des preuves solides de la performance et de l'efficacité de notre approche. Ce projet a donné lieu à une publication [Fettal, 2022c] ainsi qu'à une version française publiée dans [Fettal, 2022a].

Dans le chapitre 7, une nouvelle approche pour le clustering de graphes bipartis a été proposée par l'utilisation du co-clustering et de la convolution de graphes bilatérale. Cette

approche aborde les questions de complexité computationnelle et spatiale en utilisant des matrices factorielles et des transformations explicites de noyaux non négatifs. Nous avons démontré que le modèle proposé avait un effet de groupement et que la convolution bilatérale améliorait les performances même en l'absence du graphe vérité terrain. Des expériences sur des ensembles de données synthétiques et réelles ont démontré que ce modèle est compétitif par rapport aux méthodes actuelles de pointe pour le clustering de graphes attribués à du texte, tout en étant efficace et robuste face à des structures de graphes non informatives. Ce travail a été publié comme un article court dans une conférence [Fettal, 2022d] puis comme une version étendue dans un journal [Fettal, 2024a]; une version française est disponible dans [Fettal, 2023d].

Dans le chapitre 8, nous abordons le problème du clustering de graphes attribués multi-vues et tentons de le simplifier au maximum. Pour ce faire, nous projetons des représentations lissées des nœuds de chaque vue sur un sous-espace linéaire de même dimension et nous en faisons la moyenne avec des poids proportionnels à la qualité du clustering obtenu sur chaque vue. Ce travail a été publié dans [Fettal, 2023c].

Dans le chapitre 9, nous étendons l'approche proposée dans le chapitre 5 pour le clustering multi-vues en utilisant la propriété de sommation des noyaux. Cela nous permet de créer un algorithme de sous-espace multi-vues efficace qui peut s'executer sur des données avec des millions de nœuds sur un ordinateur portable standard.

Dans le chapitre 10, nous avons introduit une approche de l'apprentissage non supervisé de la représentation des textes, en nous concentrant sur la cohérence sémantique. En utilisant le lissage sur les noeuds, nous améliorons les embeddings de phrases à partir de modèles pré-entraînés, ce qui se traduit par une amélioration des performances pour les tâches de classification et de clustering de textes. L'efficacité de la méthode a été démontrée sur huit ensembles de données de référence. Ce travail est basé sur [Fettal, 2024b].

Enfin, dans le chapitre 11, nous avons sélectionné un cas d'utilisation industriel au sein de la Caisse des Dépôts et de ICDC, où nous explorons la détection des demandes de paiement suspectes faites par des organismes de formation. Nous examinons le processus de sélection d'un modèle initial à partir d'une variété de modèles d'apprentissage automatique couramment utilisés pour les données tabulaires, et nous montrons comment l'augmentation des caractéristiques disponibles, grâce à l'utilisation de graphes dans un cadre transductif, peut conduire à de meilleurs résultats statistiquement significatifs.

Une conclusion synthétisant toutes les contributions à la fois d'ordre académique et industrielle, ainsi qu'un ensemble de nouvelles perspectives de recherche pouvant être menées ultérieurement, clôturent cette thèse.

Contents

List of Figures

List of Tables

Introduction

Context and Motivations

The *Mon Compte Formation* (also called *Compte Personnel de Formation* [2]) is a the french public funding system for training programs. The platform was originally developed by the Groupe Caisse des Dépôts (CDC) and due to its importance is still one of the flagship projects steered by the CDC and by extension Informatique CDC (ICDC). The platform contains information about users as well as the training programs. This data can be naturally represented as graphs based on the similarities that exist between the different users and the different training programs. As such, exploring graph-related data mining and machine learning techniques can be relevant in the context of CDC.

Graph mining and learning is a rapidly growing field in the area of machine learning that deals with the analysis and understanding of data represented in the form of graphs. This field has gained significant attention in recent years due to the proliferation of graph-structured data in various domains, including social networks, biological networks, the World Wide Web, etc. The ability to extract useful information and knowledge from this type of data has become essential in many areas of research and industry. Some tasks present in graph mining and learning include graph-based representation learning, community detection, link prediction, node classification, and many more. The goal of graph mining and learning is to develop efficient and effective algorithms to uncover hidden patterns, relationships, and structures in graph-structured data, with the ultimate aim of providing insights that can be used to improve decision-making and support various applications such as recommender systems, fraud detection, and bioinformatics. However, as the volume of graph-structured data continues to grow, the need for scalable graph mining techniques is becoming even more pressing as there are several issues that can arise due to scale in graph machine learning approaches:

- **Lack of labeled data:** Graphs usually have less labeled data than other types of data which makes the training of graph-based machine learning algorithms difficult.
- **Data sparsity:** Graphs can be very sparse, which means that most nodes have only a few edges. This sparsity can make it difficult to learn meaningful representations

[2] https://www.moncompteformation.gouv.fr/

of the graph or to perform accurate predictions and the larger the graph, the more pronounced this problem is.

- **Spatial Complexity:** Despite graphs being generally sparse, some graph mining techniques break this sparsity by applying transformations on the adjacency matrix of the graph which can cause out of memory errors.
- **Computational complexity:** Some graph-based machine learning algorithms, such as those that involve matrix factorization, can have high time complexity, making them impractical for large-scale graphs.

Mitigating these issues is possible. In the case of a lack of labeled data, it is possible, for example, to consider the unsupervised learning context which is a type of machine learning where the model learns from data without the use of labeled examples. As for the data sparsity, Spatial and computational complexity problems, they can be attenuated by taking them into consideration when developing solutions to graph related problems.

In this thesis, we tackle these scalability issues and propose new techniques to effectively address them in an unsupervised context, particularly that of clustering (and embedding to some extent) with respect to different types of graphs. Unsupervised partitioning of data and graph data in particular has long been a topic of interest in the data mining community. Node clustering, also known as community detection, is an important technique in the analysis of graph data because it allows for the identification of groups of similar nodes within the graph. This can reveal underlying patterns and structures in the graph that may not be immediately apparent. One key use case of graph clustering is in social network analysis, where it can be used to identify groups of individuals with similar interests or behaviors [Handcock, 2007; Mishra, 2007]. This can be useful for targeted marketing or recommendations [He, 2010]. Another use case is in bioinformatics, where clustering can be used to identify functional modules in protein-protein interaction networks [Brohee, 2006; Dittrich, 2008; Pizzuti, 2014]. Since the first days of automatic classification, a very large number of approaches have been proposed.

With the advent of deep learning and the success it had in the supervised setting, researchers have tried to replicate this success to an unsupervised context, such as the context of (graph) clustering. However, this has led to a replicability crisis due to the use of very specific and large numbers of hyper-parameters in deep clustering models, this is especially the case in deep subspace clustering [Haeffele, 2021]. When reporting results, authors tune these hyper-parameters on a number of benchmark datasets, possibly using supervised performance metrics in this tuning process which leads to data leakage. Since many papers propose clustering models but no heuristics to tune their hyper-parameters on unseen datasets, it nullifies any empirical utility they possibly could have. John Von Neumann famously said:

> “ With four parameters I can fit an elephant, with five I can make him wiggle his trunk ”

This problem resulted in the fact that, to this day, the k-means clustering algorithm is still widely popular in spite of the deep learning revolution. We argue that this is largely due to its simple and cost-effective nature even if it requires the knowledge of the number of clusters. The simplicity of k-means does, however, result in certain limitations, particularly in the way it handles non-spherical clusters with different sizes or when the clusters are not well separated. This is explained by the fact that the optimized criterion is associated with a constrained Spherical Gaussian mixture model. Spectral clustering and EM-type algorithms, for instance, have emerged to remedy these weaknesses. Thus, several approaches will be proposed in this thesis that are simple in nature, have few hyperparameters, but also aim to overcome the difficulties of clustering different types of data, such as attributed networks, multi-view networks, etc.

Outline and Contributions

In this thesis we propose approaches for clustering (and embedding) different types of graphs, namely, classical graphs, bipartite graphs, attributed graphs, bipartite attributed graphs, and multi-view attributed graphs. The approaches we have developed all share three key characteristics that we believe contribute to the widespread popularity of the k-means clustering algorithm. These three characteristics are:

- **Simplicity:** In the sense that there are no large sets of parameters to learn as is traditionally the case in deep learning. Most models introduced in this thesis use transformations whose cost is linear in the size of the input to learn representations, and simple clustering rules to generate partitions.
- **Cost-effectiveness:** The models proposed here are also cost-effective in nature, it can be understood in the sense of Pareto optimality, meaning that they either outperform state of the art models used in our benchmarks in terms of computational complexity, empirical execution times or empirical clustering performance.
- **Few hyper-parameters:** For all models we introduced, few hyper-parameters need to be set. And for them we either propose sensible default values, or hyper-parameter selection rules.

This document is organized as follows: In Part I, we present the state of the art, specifically, in **Chapter 1**, we introduce networks and graph related notions. Then in **Chapter 2**, we provide an introduction to clustering with a particular attention paid to graph clustering. In Part II, we introduce our contributions in detail. In **Chapter 4**, we leverage Optimal Transport to propose a min cut algorithm for graph clustering with size constraints which is of particular interest for balanced clustering. In **Chapter 3**, the previous concept was extended and adjusted to deal with bipartite graphs, and led to a publication as well as a published french translation [Fettal, 2022b; Fettal, 2023a]

In **Chapter 5**, we introduce another approach for attributed graph clustering but this time we use the recent paradigms that consists in simultaneously learning graph node representations when clustering. This joint clustering and embedding can lead to better results for both tasks. It also led to publications as well as a translated french publication [Fettal, 2022c; Fettal, 2022a]. The work discussed in **Chapter 6** deals with attributed graph subspace clustering. In it, we introduced a scalable subspace clustering procedure using a factorized subspace coefficient matrix and nonnegative feature maps. It was published as [Fettal, 2023b].

Another contribution is described in **Chapter 7** where we extend the approach proposed in Chapter 6 to bipartite attributed graphs. It was originally published as a short paper in a conference then as an extended version in a journal and also has a published french translation [Fettal, 2022d; Fettal, 2024a; Fettal, 2023d].

In **Chapter 8**, we tackle the problem of multi-view attributed graph clustering and try to simplify it to its utmost. We do this by projecting smoothed representations of the nodes in each on a linear subspace of the same dimension and averaging them with weights proportional to their clusterability. This work was published as [Fettal, 2023c]. In **Chapter 9**, we extend the approach proposed in chapter 6 to multi-view the setting via using the summation property of kernels. This allows us to create an efficient multi-view subspace algorithm that scales to millions of nodes on a laptop.

In **Chapter 10**, we explore the effect of using graph smoothing on text embeddings with semantic graphs built from the data to learn more semantically discriminative representations of textual data which led to improvements in text categorization tasks. This chapter is based on [Fettal, 2024b]. Finally, in **Chapter 11**, we will dive into an industrial use case within Caisse des Dépôts and Informatique Caisse des Dépôts, where we will explore the detection of suspect payment requests made by training organisms. We will examine the process of selecting an initial model from a variety of commonly used machine learning models for tabular data, and demonstrate how augmenting the available features through the use of graphs in a transductive setting can lead to statistically significant better results. Get ready to gain insights into the practical application of machine learning in the financial sector.

List of Publications

Articles in International Conferences

- C. Fettal, L. Labiod, and M. Nadif, *More Discriminative Sentence Embeddings via Semantic Graph Smoothing,"* in **EACL**, 2024.
- C. Fettal, L. Labiod, and M. Nadif, *"Simultaneous Linear Multi-view Attributed Graph Representation Learning and Clustering,"* in **WSDM**, 2023, selected for oral presentation.
- C. Fettal, L. Labiod, and M. Nadif, *"Scalable Attributed Graph Subspace Clustering,"* in **AAAI**, 2023, selected for oral presentation.
- C. Fettal, L. Labiod, and M. Nadif, *"Efficient and Effective Optimal Transport-Based Biclustering,"* in **NeurIPS**, 2022.
- C. Fettal, L. Labiod, and M. Nadif, *"Subspace Co-clustering with Two-Way Graph Convolution,"* in **CIKM**, 2022.
- C. Fettal, L. Labiod, and M. Nadif, *"Efficient graph convolution for joint node representation learning and clustering,"* in **WSDM**, 2022.

Articles in International Journals

- C. Fettal, L. Labiod, and M. Nadif, *"Boosting Subspace Co-Clustering via Bilateral Graph Convolution,"* in **IEEE TKDE**, 2024.

Articles in National Conferences

- C. Fettal, L. Labiod, and M. Nadif, *"Biclustering Basée sur le Transport Optimal,"* in **EGC**, 2023.
- C. Fettal, L. Labiod, and M. Nadif, *"Subspace Co-clustering avec Convolution Bilatérale sur Graphe,"* in **EGC**, 2023.
- C. Fettal, L. Labiod, and M. Nadif, *"Apprentissage Joint de la Représentation et du Clustering avec un Réseau Convolutif sur Graphe,"* in **EGC**, 2022.

Codes

The majority of the code implementations of the contributions made in this thesis were made publicly available at [3].

[3] https://github.com/chakib401/

Part I

State of the Art

Chapter 1

Networks and Graphs

Objective

The significance of studying networks and their mathematical representation as graphs will be explored in this chapter.

Contents

1. Networks

A network is a collection of interconnected nodes, or points, that are connected by links or edges. The nodes can represent individuals, organizations, devices, or other entities, while the links or edges represent the relationships or connections between them. Networks can be used to model a wide range of systems and phenomena, including transportation networks, social networks, and communication networks.

1.1. History

The history of networks can be traced back to the development of communication technologies in ancient times. One of the earliest forms of communication networks were smoke signals. Indigenous people in many parts of the world used smoke signals to communicate over long distances. The messages were conveyed by the use of smoke plumes of different shapes, sizes and colors. Another early form of communication network was the use of semaphore towers. These towers were used in ancient Greece and Rome to transmit messages over long distances. The towers were equipped with a system of flags or lights that could be used to convey different messages. These towers were strategically placed along major trade routes and were used to warn of incoming invasions, report on the movement of troops and ships, and to transmit news and other important information.

The modern history of networks began with the invention of the telegraph in the 1830s and 1840s, which allowed for the rapid transmission of messages over long distances using electrical signals. In the late 19th and early 20th century, the telephone was invented and networks of telephone lines began to be built, allowing for the transmission of both voice and data. The creation of the Internet in the 1960s marked a significant milestone in the history of networks. The Internet was initially developed as a way for government researchers and academics to share information and resources, but it quickly evolved into a global network that connects millions of people and organizations. With the advent of personal computers and mobile devices in the 1980s and 1990s, the number of people and devices connected to the Internet continued to grow rapidly, leading to the development of new technologies and applications such as email, the World Wide Web, and social media. Nowadays, network technology continues expanding with the implementation of new protocols, such as 5G and IoT technology, allowing networks to connect an even larger and diverse range of devices and to handle high speed and low latency applications, such as virtual reality and autonomous vehicles.

1.2. Applications

Networks play a critical role in modern society, enabling communication, collaboration, and the sharing of information and resources. They have become essential for business, government, and society as a whole, allowing for increased efficiency, productivity, and inno-

vation. They are used in a wide range of applications and industries, some examples include:

Communications Network technologies such as the telephone and the Internet are used to transmit voice and data over long distances. These technologies have revolutionized the way we communicate, allowing us to connect with people across the globe in real-time. The telephone network uses a complex system of wires and switches to connect calls and transmit voice data, while the Internet uses a network of computers and routers to transmit data using various protocols such as TCP/IP.

Transportation Networks of roads, highways, railroads, and airports are used to move people and goods from one location to another. These transportation networks are essential for the functioning of modern economies, allowing for the efficient movement of goods and people. They are also important for connecting communities and promoting economic development.

Power and utilities Networks of pipelines, power lines, and cable are used to transport energy and other resources to customers. These networks are critical for the functioning of modern societies, providing us with the energy and resources we need to power our homes and businesses. The power grid is a complex network of power plants, transmission lines, and substations that are used to generate and distribute electricity.

Social networks Online platforms such as Facebook, Twitter, and LinkedIn are used to connect people and facilitate communication and collaboration. These social networks allow individuals to connect with friends and family, share information, and stay connected with people all over the world. They also have an impact on the way we interact with each other, and have become an important tool for businesses and organizations to promote products and services, and reach out to new customers.

Supply chain management Companies use networks of suppliers, manufacturers, and distributors to manage the flow of goods and materials throughout the economy. A supply chain is the network of organizations, people, activities, information, and resources involved in moving a product or service from supplier to customer. This includes the movement and storage of raw materials, work-in-progress inventory, and finished goods from point of origin to point of consumption.

Biology Network biology is a field of study that aims to understand the structure and function of biological networks, including networks of proteins, genes, and metabolic pathways in cells. This field of study is becoming increasingly important as we learn more about the complex interactions that occur within living organisms and how these interactions influence the functioning of living systems.

Cybersecurity Computer networks and the internet are vulnerable to cyber-attacks, organizations use various measures to secure their networks, such as firewalls and intrusion detection systems. Cybersecurity is becoming an increasingly important concern as more and more of our personal and professional lives are conducted online.

Smart cities IoT technology is increasingly being used to create "smart" networks of sensors, cameras, and other devices that can be used to monitor and control various aspects of city infrastructure and services such as traffic and public transportation. These smart networks allow cities to gather and analyze data in real-time and make better decisions about how to manage resources and improve services.

Healthcare Telemedicine and connected medical devices are increasingly being used to create networks that connect patients, doctors, and hospitals, making it possible to provide remote monitoring and care. These healthcare networks allow doctors to remotely monitor patients' health and provide care remotely, which is particularly useful in rural areas and for patients with mobility issues.

1.3. Properties

Real-world networks can exhibit a variety of properties that are important to understand when analyzing and modeling these systems. Some notable properties of real-world networks include:

Scale-free Many real-world networks exhibit a scale-free property, meaning that they have a few highly connected nodes (called "hubs") and many poorly connected nodes. This property is often observed in social networks, where a small number of individuals have a large number of connections, while most individuals have only a few connections.

Small-world Many real-world networks are also small-world networks, meaning that they have a high degree of clustering (i.e., nodes tend to be connected to those that they are connected to) and a relatively small average path length (i.e., the average number of steps needed to get from one node to another). This property is often observed in social networks, where individuals tend to be connected to those that they know, and the number of intermediaries between any two individuals is relatively small.

Hierarchical Some real-world networks have a hierarchical structure, in which nodes are grouped into different levels or layers and the links between layers are sparser than those within layers. This is often observed in transportation networks, where cities and towns form hubs that are connected by less populated areas, or in hierarchical organizations

Community structure Many real-world networks exhibit a community structure, meaning that the nodes can be grouped into densely connected clusters or communities. These

clusters may represent groups of individuals with similar interests, geographic proximity, or other characteristics.

Evolving Real-world networks are often dynamic, evolving over time as nodes and links are added, removed, or rewired. Understanding the dynamics of network evolution is critical for understanding how networks change and adapt over time.

Correlation Real-world networks are often correlated, meaning that the properties of nodes or edges are dependent upon one another. For example, many real-world networks have degree-degree correlations, where nodes with high degree tend to connect to other nodes with high degree.

Weighted and directed Real-world networks can also be weighted, meaning that the connections between nodes have values, such as the amount of flow or influence. They can also be directed, meaning that connections have a direction, such as a sender and a receiver.

These are some of the properties that are commonly observed in real-world networks. However, the specific properties of a network can vary depending on the system it represents and each network has its own unique characteristics.

2. Graphs

A graph and a network are similar concepts that are used to represent and analyze systems of interconnected elements. A graph can be used to represent a network, where the vertices of the graph represent the nodes of the network, and the edges of the graph represent the links or connections between the nodes. In this way, a graph provides a mathematical representation of the structure of a network, and graph theory can be used to analyze the properties of the network, such as connectivity, centrality, and community structure.

2.1. History

The study of networks, and their mathematical representation as graphs, has a long and rich history. The earliest known use of graph theory dates back to the 18th century. The formal definition of a graph and the study of graphs as mathematical objects was first introduced by Leonhard Euler. He used the concept of a graph to solve the famous Seven Bridges of Königsberg problem in 1735. This problem involves finding a walk through the city that would cross each of the seven bridges once and only once. Euler's solution to this problem is considered to be the first theorem in graph theory, and it is also considered as the starting point of the branch of mathematics. In the 19th century, graph theory focused primarily on the study of regular graphs, chromatic polynomials, topological graph theory and combinatorial design theory. Key developments included the discovery of chromatic polynomials by Cauchy and Bolyai, the formulation of the Four Color Problem by Francis

Guthrie, and the proposal of the "Fifteen Schoolgirl problem" by Thomas Kirkman, which led to the development of combinatorial design theory. These works laid the foundation for graph theory as a branch of mathematics. In the 20th century, the study of networks began to branch out into other fields such as physics, sociology, and computer science. Mathematicians such as Paul Erdös and Alfred Rényi developed the concept of random graphs and graph entropy, which laid the foundation for the study of complex networks. Sociologists such as Harrison White, Mark Granovetter, and Ronald Burt began to study social networks and their implications for society. In the 21st century, the study of networks and graph theory has continued to evolve and expand, with applications in many fields such as biology, computer science, economics, and many more. In computer science and machine learning, the development of Graph Convolutional Networks (GCN) has expanded the ability to analyze graph-structured data, which has made it possible to extract features from the data that may have been difficult to capture with other types of methods.

2.2. Definitions

Graphs A weighted undirected graph can be represented by a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that $\mathcal{V}$ is a set of graph nodes and $\mathcal{E}$ is a set of edges represented with triplets of the form $(v, u, w) \in \mathcal{V}^2 \times \mathbb{R}$ where v and u are the extremities of the edge and w is the edge weight. The neighborhood of a node v denoted by $N(v)$ is the set of all nodes that are connected to v. A p-th order neighborhood is the set of nodes that are at most p edges away from v.

Adjacency Matrix The adjacency matrix (generally denoted by $\mathbf{A}$) is a squared matrix which is the most commonly used representation for graphs. An entry a_{ij} is nonzero if and only if there exists an edge between vertex v_i and v_j. It has a different structure and properties following the type of the graphs it represents. For example, in the case of undirected weighted graphs with n nodes, The adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a real symmetric matrix.

Laplacian Matrix The unnormalized Laplacian matrix of a graph is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D}$ is the diagonal matrix of degrees of the graph i.e. $d_{ii} = \sum a_{ij}$. The Laplacian matrix uncovers many properties of graphs. Its spectral decomposition allows for spectral embedding in dimensionality reduction methods. It is used to extend the concept of signal processing to graphs. It is also used in finding approximate solutions to min cut problems in graph partitioning. Furthermore, because the graph Laplacian is a real symmetric matrix, it has a complete set of orthonormal eigenvectors, which we denote by $\{\mathbf{u}_l\}_{l=1}^n$ and a set of associated real and nonnegative eigenvalues $\{\lambda_l\}_{l=1}^n$. Its spectral decomposition is written as $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\top$ such that Λ is a diagonal matrix with the eigenvectors of $\mathbf{L}$ along its diagonal and $\mathbf{U}$ is a matrix with the eigenvectors of $\mathbf{L}$ as its columns.

2.3. Types of Graphs

Bipartite Graphs A bipartite graph is a graph whose nodes can be divided into two sets U and V which are independent and disjoint. It can be represented using triplet $\mathcal{G} = (U, V, \mathcal{E})$. Bipartite Graphs have an adjacency matrix of the form:

$$\mathbf{A} = \begin{pmatrix} \mathbf{0}_{|U|\times|U|} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{0}_{|V|\times|V|} \end{pmatrix}. \tag{1.1}$$

It can then be fully characterized using the condensed representation $\mathbf{B}$ which is generally called the biadjacency matrix of size $|U| \times |V|$.

Attributed Graphs An attributed graph is a graph that in addition to having the graph topology has features associated with its nodes and/or edges. Here we are interested in the case of node-attributed graphs which can be characterized by a triplet $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ where $\mathcal{V}$ is the vertex set, $\mathcal{E}$ is the edge set, and $\mathbf{X}$ is a set of node level features.

Multi-view Graphs A multi-view graph of cardinally m is a set $\{\mathcal{G}_i\}_{i=0}^{i=m}$ such that each graph $\mathcal{G}_i$ characterizes the same nodes but not necessarily in the same way. For example, this could mean that node v_i and v_j could be connected in the l-th graph $\mathcal{G}_l$ but not in the g-th graph $\mathcal{G}_g$.

2.4. Graph Signal Processing

Graph Signal Processing, or GSP for short, provides a framework to analyze and process signals defined on graphs, by extending traditional signal processing concepts and tools to the graph domain. This allows for the representation and manipulation of signals in a way that is tailored to the specific structure of the graph.

Graph Signals Graph signals are mappings from the set of vertices to the real numbers. A graph signal for a given graph $\mathcal{G}$ can be represented using vector $\mathbf{f} = [f(v_1), \ldots, f(v_n)]$ such that $f : \mathcal{V} \to \mathbb{R}$ is a real-valued function on the vertex set. The smoothness of a signal $\mathbf{f}$ over graph $\mathcal{G}$ can be characterized using the Laplacian quadratic form associated with Laplacian $\mathbf{L}$:

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j} a_{ij}(\mathbf{f}_i - \mathbf{f}_j)^2. \tag{1.2}$$

Graph Fourier Transform Analogously to the classical Fourier transform, the graph Fourier transform is defined as the expansion of the signal $\mathbf{f}$ in terms of the eigenvectors of the graph Laplacian:

$$\hat{\mathbf{f}} = \mathbf{U}\mathbf{f}. \tag{1.3}$$

This The inverse graph Fourier transform is then given as

$$\mathbf{f} = \mathbf{U}^\top \hat{\mathbf{f}}. \tag{1.4}$$

The graph eigenvalues carry a notion of frequency similar to those of the eigenfunctions in classical FT, namely, the eigenvectors associated with low frequencies λ_l, vary slowly across the graph, and those associated with larger values oscillate rapidly.

Graph Filters Classical graph signal filtering is the process of attenuating or amplifying the contribution of some component frequencies. Using matrix theory, it is possible to write the filtering process as

$$\hat{\mathbf{f}}_{out} = \hat{h}(\mathbf{L})\hat{\mathbf{f}_{in}} \tag{1.5}$$

where h is the transfer function of the filter and

$$\hat{h}(\mathbf{L}) = \mathbf{U} \begin{pmatrix} \hat{h}(\lambda_1) & & 0 \\ & \ddots & \\ 0 & & \hat{h}(\lambda_n) \end{pmatrix} \mathbf{U}^\top \tag{1.6}$$

A specific class of filters that has an intuitive interpretation in the vertex domain is that of the polynomial filters. When the filter is a p-th order polynomial of the form $\hat{h}(\mathbf{L}) = \sum_{m=0}^{p} \theta_m \mathbf{L}^m$, the frequency filtered signal at vertex i, is a linear combination of the components of the input signal at vertices within a p-hop local neighborhood of vertex i:

$$\mathbf{f}_{out}[i] = \alpha_{ii}\mathbf{f}_{in}[i] + \sum_{j \in N(i,p)} \alpha_{ij}\mathbf{f}_{in}[j] \tag{1.7}$$

where $N(i,p)$ is the p-th order neighborhood of vertex i. It is possible to then make the correspondence with a polynomial filter as follows:

$$\alpha_{ij} = \sum_{m=d_{\mathcal{G}}(i,j)}^{p} \theta_m (\mathbf{L}^m)_{ij} \tag{1.8}$$

where $d_{\mathcal{G}}$ is the shortest distance between node i and j.

Graph Convolution Since it is impossible to directly define a convolution product for graphs since we cannot express a meaningful translation operator on graphs. Another way of generalizing convolutions is by defining them as

$$\mathbf{f} * \mathbf{h} := \mathbf{U}((\mathbf{U}^\top \mathbf{f}) \odot (\mathbf{U}^\top \mathbf{h})) \tag{1.9}$$

where $\odot$ is the Hadamard (element-wise) product. This definition makes the convolution in the vertex domain equivalent to multiplication in the spectral domain. For more details, see [Shuman, 2013].

2.5. Neural Networks on Graphs

Convolutional Neural Networks (CNNs) are a special type of neural network that have been particularly successful in image and video recognition tasks. Researchers tried to replicate this success in graph-related tasks by finding ways to extend the different concepts used by CNNs in the regular euclidean image domain to irregular and structured graph data.

Convolutional Neural Networks on Graphs [Defferrard, 2016] initially proposed a way to extend the convolutional neural network from the euclidean domain to graphs using a localized polynomial spectral filter of the form

$$\hat{h}(\Lambda) = \sum_{m=0}^{p} \theta_m \Lambda^m \tag{1.10}$$

and then doing the filtering obtaining an output signal via formula 1.9. They noted, however, that the computational complexity of the multiplication of the signal with the eigenvector matrix $\mathbf{U}$ is costly and subsequently proposed to instead use the Chebyshev expansion on Λ:

$$\hat{h}(\Lambda) = \sum_{m=0}^{p} \theta_m T_m(\hat{\Lambda}). \tag{1.11}$$

where $\hat{\Lambda} = \Lambda/\lambda_{\max} - \mathbf{I},$. The filtering operation can then be written as

$$\hat{\mathbf{f}}_{out} = \hat{h}(\mathbf{L})\hat{\mathbf{f}_{in}} = \sum_{m=0}^{p} \theta_m T(\hat{\mathbf{L}}), \tag{1.12}$$

where $\hat{\mathbf{L}} = \mathbf{L}/\mathbf{L}_{\max} - \mathbf{I}$. Each Chebyshev term can then be computed with the recurrence:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_m(x) &= 2x\,T_{m-1}(x) - T_{m-2}(x). \end{aligned} \tag{1.13}$$

Their network then learns the Chebyshev coefficients using backpropagation.

Graph Convolutional Networks [Kipf, 2017] proposed a layer-wise linear model where each layer is followed by a non-linearity. They do this by setting $p = 1$ in each layer and making the approximation $\lambda_{\max} \approx 2$. Using a normalized Laplacian. The formula of the filtering operation in each layer becomes

$$\mathbf{f}_{out} = \theta_0 \mathbf{f}_{in} + \theta_1(\mathbf{L}_{sym} - \mathbf{I})\mathbf{f}_{in} = \theta_0 \mathbf{f}_{in} - \theta_1 \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{f}_{in} \tag{1.14}$$

The number of parameters is further decreased by setting $\theta = \theta_0 = -\theta_1$. For further stability, they also introduce a renormalization trick

$$\mathbf{I} + \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}} \rightarrow \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}, \tag{1.15}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self loops. Multiple layers are then stacked together, and the definition is extended to multi-dimensional signals represented via matrix $\mathbf{X}$, giving rise to this recursive formulation of the Graph Convolutional Network:

$$\begin{aligned} &\mathbf{H}^{(l+1)} \leftarrow \sigma\left(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\Theta^{(l)}\right) \\ &\text{with} \quad \mathbf{H}^{(0)} = \mathbf{X} \end{aligned} \tag{1.16}$$

where $\Theta^{(l)}$ are the learnable weights of the l-th layer, they are optimized for some downstream task.

Simplified Graph Convolutional Networks [Wu, 2019] proposed the simplified GCNs (SGCs) by arguing that the non-linearities are superfluous. SGC uses a single layer of graph convolution which makes it computationally less expensive than traditional GCNs, yet still allows to retain the ability to extract useful features from graph-structured data. After dropping the non-linearities from the GCN, we obtain

$$\mathbf{H} \leftarrow (\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2})\dots(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2})\mathbf{X}\Theta^{(0)}\Theta^{(1)}\dots\Theta^{(p)}. \tag{1.17}$$

Since all the p weight matrices are free, they can be reparameterized into a single weight matrix and the adjacency matrix products collapse into a single matrix power operation. The rule for the SGC then becomes:

$$\mathbf{H} \leftarrow \left(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\right)^{p}\mathbf{X}\Theta. \tag{1.18}$$

The weight matrix Θ is then optimized for a downstream task in a similar manner as the original GCN. SGCs have the advantage of having less parameters to learn, which make them faster to train. Additionally, they tend to require less data to generalize well. Similarily to GCNs, SGC architectures have been studied and applied in various domains such as graph classification, node classification, and even link prediction.

Chapter 2

A Primer on Graph Clustering

Objective

The goal of this chapter is to discuss the task of clustering, particularly as it pertains to specific types of graphs, namely, classical, attributed, bipartite, multi-view, and their hybrids.

Contents

1. Clustering

Clustering (or clustering analysis) is the task of grouping a set of observations in such a way that observations in the same group (called a cluster) are more similar, according to some metric, to each other than to those in other clusters. Here, we will be discussing the more popular approaches to clustering.

1.1. Types of Clustering

Hard versus Fuzzy Clustering Before starting, it is important to note that when people refer to clustering approaches, they generally implicitly refer to hard clustering where the goal is to generate a hard partition of the data, meaning that each point exclusively belongs to a single cluster. Fuzzy (or soft clustering), on the other hand, refers to a clustering in which each data point can belong to more than one cluster, and where we quantify the "degree" of its membership to each cluster.

Partitional clustering Partitional clustering splits a data set into a set of mutually exclusive and jointly exhaustive clusters. Given a data set of n points, a partitioning method constructs a k partition of the data, with each partition representing a cluster. To this end, clusters are often characterized using a cluster representative, and an optimization problem based on these representatives is introduced. An example of this type of approach is the widely used k-Means algorithm where the representative is called a centroid and is defined as the mean of all data points that belong to the cluster it represents. Data points are grouped according to their closest centroid. Given a data matrix $\mathbf{X}$, the k-means problem is formulated as:

$$\underset{\{\mathbf{g}_i\}_{i=1}^{n},\{\mathbf{f}_j\}_{j=1}^{k}}{\arg\min} \quad \sum_{i=1}^{n}\sum_{j=1}^{k} g_{ij}\|\mathbf{x}_i - \mathbf{f}_j\|^2 \tag{2.1}$$

where $\mathbf{f}_j$ is the j-th centroid and g_{ij} is the assignment of i-th point i.e. $g_{ij} \in \{0,1\}$ and $g_{i\cdot} = 1$. Equivalently, in matrix form, the formulation of k-means is:

$$\underset{\mathbf{G},\mathbf{F}}{\arg\min} \quad \|\mathbf{X} - \mathbf{GF}\|^2 \text{ such that } \mathbf{G1} = \mathbf{1} \text{ and } \mathbf{G} \in \{0,1\}^{n\times k} \tag{2.2}$$

This optimization problem is NP-hard, however, effective heuristics exist, one of which is the Lloyd algorithm which is commonly referred to as the k-means algorithm due to its widespread use. Following an initial selection of centroids, this algorithm consists of two steps repeated alternatingly until convergence, an assignment step where each data point is assigned to the cluster whose centroid is closest. The second step is called the update step where each centroid is recalculated as the mean of the data points that belong to its cluster.

Hierarchical Clustering Hierarchical Clustering is one of the more popular clustering algorithms and consists in creating a hierarchy of clusters. This hierarchy is generally repre-

sented visually through a dendrogram. Hierarchical approaches fall into two categories:

- **Agglomerative Clustering** Agglomerative approaches are the more popular type of hierarchical clustering. They use a bottom-up approach where algorithms start with the data each being in a unique cluster and then start to merge clusters according to a similarity recursively until reaching a stopping criterion. Depending on the similarity measure, multiple agglomerative algorithms have been proposed, for example, we can cite the single-linkage clustering, complete-linkage clustering, etc. These approaches generally have a time complexity of $\mathcal{O}(n^3)$
- **Divisive Clustering** In contrast to agglomerative approaches, divisive approaches proceed in a top-down manner, where they start with the data all grouped in a single clustering and start to recursively split clusters until reaching a stopping criterion. Examples of divisive algorithms include DIANA and MONA. These approaches generally have a time complexity of $\mathcal{O}(2^n)$ but faster heuristics do exist.

Graph-Based Clustering Graphs are structures characterized by a set of nodes (or vertices) connected via a set of edges. Mathematically, a graph $\mathcal{G}$ are represented by a pair $\mathcal{G} = (\mathcal{E}, \mathcal{V})$ where $\mathcal{V}$ is the set of vertices, and $\mathcal{E}$ is the set of edges. Graph clustering generally refers to the task of node clustering inside graphs. It is often posed as a *mincut* problem where a cut is a partition of its vertices $\mathcal{V}$ into two disjoint subsets $\mathcal{A}$ and $\bar{\mathcal{A}}$. The value of a cut is given by

$$\text{cut}(\mathcal{A}) = \sum_{v_i \in \mathcal{A}, v_j \in \bar{\mathcal{A}}} w_{ij}. \tag{2.3}$$

The multi-way *mincut* problem is then to find a partition $(\mathcal{A}_1, ..., \mathcal{A}_k)$ of the set of vertices $\mathcal{V}$ into k different groups that is minimal in some metric. Intuitively, we wish for the edges between different subsets to have small weights, and for the edges within a subset have large weights. Formally, it is defined as

$$\text{min-cut}(\mathbf{A}, k) = \min_{\mathcal{A}_1, ..., \mathcal{A}_k} \sum_{i=1}^{k} \text{cut}(\mathcal{A}_i). \tag{2.4}$$

This problem can also be stated as a trace minimization problem by representing the resulting partition $\mathcal{A}_1, \ldots, \mathcal{A}_k$ using an assignment matrix $\mathbf{Q}$ such that for each row i, we have that

$$x_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ is in } A_j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

This condition is equivalent to introducing two constraints which are $\mathbf{Q} \in \{0,1\}^{n \times k}$ and $\mathbf{Q}1 = 1$. The minimum k-cut problem can then be formulated as

$$\text{min-cut}(\mathbf{A}, k) = \min_{\substack{\mathbf{Q} \in \{0,1\}^{n \times k} \\ \mathbf{Q}1 = 1}} Tr(\mathbf{Q}^\top \mathbf{L} \mathbf{Q}), \tag{2.6}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{A}$ refers to the graph Laplacian of the graph $\mathcal{G}$ and $\mathbf{D}$ is the diagonal matrix of degree of $\mathbf{A}$, i.e., $d_{ii} = \sum_j w_{ij}$.

In practice, however, solutions to the *mincut* problem do not yield satisfactory partitions due to the formation of small groups of vertices. Consequently, versions of the problem that take into account some notion of "size" for these groups have been proposed. The most commonly used ones are normalized cut (*ncut*) and ratio cut (*rcut*). A solution $\mathbf{Q}$ for the ncut problem is formed by stacking the first k-eigenvectors of the symmetrically normalized Laplacian $\mathbf{L}_s = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ as its columns, and then applying a clustering algorithm such as k-means on its rows and assign the original data points accordingly.

Model-Based Clustering Model-based clustering is a statistical approach to clustering and assumes that the data was generated from a finite mixture of component models. The idea is that there are as many clusters as there are individual models in the mixture and that the component responsible for the generation of a particular observation determines the cluster to which said observation belongs. Given a data matrix $\mathbf{X}$, the density function of a mixture model comprised of k components is given by:

$$f(\mathbf{X}; \Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \pi_j \varphi(\mathbf{x}_i; \theta_j) \tag{2.7}$$

where $\Theta = (\pi_1, \ldots, \pi_k, \theta_1, \ldots, \theta_k)$ are the parameters of the mixture, π_j is the mixing proportion of the j-th component (with $\sum_i \pi_i = 1$) while $\varphi(.|\theta_j)$ refers to the density function of the j-th component.

The most common type of mixture model is that of the Gaussian mixture model where the individual components correspond to Gaussian distributions i.e. $\forall_i\ \theta_i = (\mu_i, \Sigma_i)$. To find the parameters, a common strategy is often to maximize the log-likelihood as a function of the parameters given the data matrix $L(\Theta; \mathbf{X}) = f(\mathbf{X}; \Theta)$. This maximization can be done using the *Expectation-Maximization* (EM) algorithm. Other variants based on the original EM algorithm have been proposed such as the Classification EM (CEM) algorithm and the Stochastic EM algorithm (SEM).

Deep Clustering Deep clustering frameworks combine representation learning and clustering into an end to end model. The idea is to make a neural network component learn suitable representations to adapt to the down-stream clustering task which is performed via a clustering module that is also included in the model. The interaction between these two modules can be sequential, iterative, or simultaneous.

1.2. Clustering Performance Metrics

To evaluate or compare clustering approaches, we can use a combination of the following external indices:

Clustering Accuracy In a supervised learning setting, given a set of ground-truth class assignments $\mathbf{y} \in \mathcal{C}^n$ and a set of class predictions $\hat{\mathbf{y}} \in \mathcal{C}^n$ generated by some model, both of size n where $\mathcal{C}$ is the set of possible classes. A popular measure of performance is accuracy:

$$\mathrm{ACC}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} [y_i = \hat{y}_i], \tag{2.8}$$

where [.] is the Iverson bracket. In the unsupervised setting of clustering, since clusters do not naturally match classes, a transformation of the cluster labels must be applied in order to define an extension of accuracy to clustering:

$$\mathrm{CA}(\mathbf{y}, \hat{\mathbf{y}}) = \max_{\pi \in P} \frac{1}{n} \sum_{i=1}^{n} [y_i = \pi(\hat{y}_i)], \tag{2.9}$$

where P is the set of functions possible bijections from $\mathcal{C}'$ to $\mathcal{C}$, where $\mathcal{C}$ is the set of classes and $\mathcal{C}'$ is the set of clusters. This problem is an instance of the linear assignment problem, it can be solved via the Hungarian method.

Clustering F1-score Analogously to accuracy, a similar extension of the F1-score to clustering can be defined by solving a similar optimization problem by finding the best bijection from the cluster labels to the ground-truth class labels.

Adjusted Rand Score Another clustering quality measure is the Adjusted Rand Score [Hubert, 1985] which lies in $[-0.5, 1]$. A value close to zero means that the assignment is random while a value of one means a perfect match between two partitions. Given two partitions $A = A_1, \ldots, A_k$ and $b = B_1, \ldots, B_k$, we can quantify the overlap between two partitions using table

$X \diagdown Y$	Y_1	Y_2	$\cdots$	Y_s	sums
X_1	n_{11}	n_{12}	$\cdots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\cdots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\cdots$	n_{rs}	a_r
sums	b_1	b_2	$\cdots$	b_s	

.

The adjusted rand index is then computed as follows:

$$\mathrm{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}} \tag{2.10}$$

Normalized Mutual Information The normalized mutual information is a normalized version of the mutual information score. It takes values between one and zero with one corresponding to a perfect correlation and zero corresponding to no mutual information.

Given the same table as for ARI, we define normalized mutual information as:

$$\text{NMI} = \frac{I(Y;X)}{H(X)+H(Y)}, \tag{2.11}$$

where $H(.)$ is the entropy and $I(Y;X)$ is the mutual information of the two partitions. Other normalizations are possible, for example, it is possible for the denominator to be equal to $\sqrt{H(X)H(Y)}$ or $\max\{H(X), H(Y)\}$.

2. Bipartite Graph Clustering: Biclustering

Bipartite graph clustering is a technique for identifying groups or clusters of nodes in a bipartite graph. The goal is to find groups of "left" nodes that are densely connected to groups of "right" nodes. The clusters are formed by the grouping of the vertices of one set and not both. It is used in many real-world applications, like community detection, market segmentation and collaborative filtering.

Biclustering, also known as co-clustering, is a type of data mining technique used to identify and extract patterns from a two-dimensional data set, such as a matrix. The goal of biclustering is to find groups of rows and columns of a matrix that have similar values, in a way that the patterns identified by biclustering are not present in any individual rows or columns. Since bipartite graphs can be represented using a biadjacency matrix then biclustering is suitable for bipartite graph clustering since it will yield a partition of both rows and columns and since these rows and columns represent the left set and right node sets respectively a simultaneous clustering of both of them using biclustering results in a clustering of the bipartite graph.

2.1. Types of Biclustering

Partitional Biclustering We define partitional co-clustering approaches as ones that learn a smaller summarized version of the original data matrix, it is generally referred to as the *summary matrix* which can be seen as the extension of the concept of cluster representatives in traditional clustering to co-clustering. To this end, partitional algorithms learn the summary matrix based on two assignment matrices, one for the rows and one for the columns of the data matrix. This is usually done in an alternating manner where row assignments are computed given the current column assignments and vice versa. For example, a popular criterion of the co-clustering of continuous data is the least-squares criterion:

$$C(\mathbf{X}, \mathbf{Z}, \mathbf{W}, \mathbf{C}) = \sum_{i,j,h,l} z_{ih} w_{jl} (x_{ij} - c_{hl})^2 \tag{2.12}$$

where $\mathbf{Z}$ and $\mathbf{W}$ represent the assignment matrices of the rows and columns respectively and $\mathbf{C}$ is the summary matrix. The fact that the compressed matrix $\mathbf{C}$ is computed based on the

two matrices $\mathbf{Z}$ and $\mathbf{W}$ is made clearer with the matrix formulation:

$$C(\mathbf{X}, \mathbf{Z}, \mathbf{W}, \mathbf{C}) = \langle L(\mathbf{X}, \mathbf{C}) \otimes \mathbf{Z}, \mathbf{W} \rangle . \tag{2.13}$$

where $\otimes$ denotes the tensor-matrix multiplication and $L(\mathbf{X}, \mathbf{C})$ is the fourth order tensor of all squared pairwise differences between entries of $\mathbf{X}$ and $\mathbf{C}$. A double k-means procedure was introduced in order to heuristically solve problem 2.12. The steps are similar to those of the regular k-means except that there are two assignment steps, one for the rows and one for the columns before updating the summary matrix.

Graph-based Biclustering Graph co-clustering algorithms generally propose to perform graph clustering on the bipartite adjacency matrix directly, given a data matrix $\mathbf{X}$ which represents the biadjacency matrix. In the traditional max association cut problem, the goal is to maximize the weight of the edges inside connected components (or clusters) after performing a cut. Then, if we were to formulate a the max association problem using a bipartite adjacency matrix, we would obtain that

$$Tr(\mathbf{Q}^\top \mathbf{A} \mathbf{Q}) \quad = \quad Tr\left(\begin{bmatrix} \mathbf{Z} \\ \mathbf{W} \end{bmatrix}^\top \begin{bmatrix} \mathbf{0}_{n\times n} & \mathbf{X} \\ \mathbf{X}^\top & \mathbf{0}_{d\times d} \end{bmatrix} \begin{bmatrix} \mathbf{Z} \\ \mathbf{W} \end{bmatrix} \right) \quad = \quad 2Tr\left(\mathbf{Z}\mathbf{X}\mathbf{W}^\top \right) \tag{2.14}$$

where $\mathbf{Z}$ and $\mathbf{W}$ can be seen as the assignment matrices of the rows and columns respectively. From this bipartite max-assoc problem, we can derive several co-clustering algorithms introduced in the literature by performing transformations on matrix $\mathbf{A}$ or by applying different graph clustering algorithms. For example, [Dhillon, 2001] proposed an efficient variant of the spectral clustering algorithm to obtain a partition over both rows and columns using singular value decomposition. [Barber, 2007] proposed to replace the biadjacency matrix $\mathbf{X}$ with a modularity matrix $\mathbf{M} = (m_{ij}) = (x_{ij} - \frac{x_{i.}x_{.j}}{x_{..}})$ before applying a recursive graph clustering algorithm. Other approaches for solving the modularity maximization for bipartite graphs are proposed in [Labiod, 2011; Ailem, 2016].

Model-based biclustering In model-based approaches, blocks inside the data matrix are assumed to be drawn from a probability distribution, and each distribution corresponding to a single block has its own parameters. An interesting formulation for this problem is Latent Block Model (LBM) [Govaert, 2003], where given a data matrix $\mathbf{X}$, a latent block is a set of matrix entries e_{ij} generated by the same probability density function. LBM assumes that e_{ij} are i.i.d. given their co-cluster membership which is identified using the pair (z_i, w_j), and that z and w are independent latent variables that follow a multinomial distribution. The likelihood function of the LBM model is

$$f(\mathbf{X}|\Theta) = \sum_{z\in\mathcal{Z}, w\in\mathcal{W}} \left(\prod_{i,h} \pi_h^{z_{ih}} \right) \left(\prod_{j,g} \rho_g^{w_{jg}} \right) \left(\prod_{i,j,h,g} \varphi(x_{ij}; \theta_{hg})^{z_{ih}, w_{jg}} \right) \tag{2.15}$$

where $\Theta = (\pi_1, \ldots, \pi_k, \rho_1, \ldots, \rho_l, \theta_{11}, \ldots, \theta_{kl})$ are the parameters of the LBM, π_h and ρ_g are the mixing proportions of the (h, g) block, and $\mathcal{Z}$ and $\mathcal{W}$ represent the set of all possible row and cluster assignments respectively. Similar to the standard mixture model, the maximum likelihood estimate of the parameters Θ can be obtained using the expectation-maximization algorithm. The well studied Stochastic Block Model (SBM) is an LBM with the same units in rows and columns and is used to model graphs [Matias, 2014]. In the case of SBMs, there is only one latent variable [Boutalbi, 2021; Riverain, 2022].

3. Attributed Graph Clustering

In this section, we will explore various perspectives on the task of attributed graph clustering such as architectures, learning paradigms and so on.

3.1. Architecture Types

We can broadly divide attributed graph clustering approaches as having one of two types of architectures based on how their representation learning component and clustering component interact.

Sequential Representation Learning & Clustering Here, models initially learn representations for the nodes before applying some clustering algorithm on said representations. For example, [Zhang, 2019] proposes to exploit high-order connections in the clustering process through an adaptive rule for neighborhood order selection when learning representations. [Zhu, 2021] applied the same principle to obtain node partitions but using another neighborhood propagation matrix.

Simultaneous Representation Learning & Clustering The other type of clustering is the type that considers the clustering objective when learning representation. This paradigm is usually referred to as clustering-friendly representation learning. For example, [Rozemberczki, 2019] places nodes in a latent space where the vertex features minimize the negative log likelihood of preserving sampled vertex neighborhoods while they are simultaneously clustered in this space. [Park, 2019] proposed a graph convolutional autoencoder architecture with a subspace clustering regularization term suitable for image clustering.

3.2. Learning Paradigms

Autoencoder-Based Approaches An autoencoder [Rumelhart, 1985] is a type of neural network used to learn lower-dimensional representations of data in an unsupervised manner. These learned representations are generally referred to as embeddings. [Kipf, 2016] proposed an extension of the traditional autoencoder that is able to deal with attributed graphs. The autoencoder is widely used in the context of graph clustering, often in a way that leads to

learning clustering-friendly embeddings. The generic formulation is:

$$\min_{\Theta} \quad L\left(f(\mathbf{A}, \mathbf{X}),\ (\phi \circ \psi)\ g(\mathbf{A}, \mathbf{X})\right) \ +\ \Omega(\Theta) \tag{2.16}$$

where L is some reconstruction loss function, f and g are some characterizing functions of the graph, and ϕ and ψ are encoding and decoding functions, respectively. All these functions can be parameterized by Θ. Finally, γ is a parameter regularization function.

Notable examples include [Wang, 2017] which introduced a marginalization process for their graph autoencoder by introducing randomness into the node features and then performing spectral clustering using the resulting embeddings. [Pan, 2018] proposed an adversarially regularized graph autoencoder and variational version to learn graph embeddings by minimizing reconstruction error while enforcing the embedding to match a prior distribution; clustering is performed by applying k-means on the embeddings. [Park, 2019] plugged a subspace clustering cost function into their autoencoder architecture to learn a subspace affinity matrix. [Mrabah, 2022] proposes an autoencoder sampling operator to deal with noisy clustering assignments, and a correction operator to deal with feature drift by gradually transforming the reconstructed graph into a clustering-oriented one.

Contrastive Approaches The goal of unsupervised contrastive representation learning is to learn a latent space where similar (positive) pairs of observations $(\mathbf{x}, \mathbf{x}^+)$ stay close to each other while dissimilar (negative) ones $(\mathbf{x}, \mathbf{x}^-)$ are far apart. The positive sample $\mathbf{x}^+$ is usually generated via an augmentation process from the original observation $\mathbf{x}$, while the negative sample $\mathbf{x}^-$ is generally taken to be an observation from the original dataset that is different from $\mathbf{x}$. For example, [Zhao, 2021] proposed to simultaneously do clustering and representation learning, in order to avoid false-negative samples in the random negative sampling by considering the cluster information. Consequently, they randomly select negative samples from the clusters which are different from the positive sample's cluster. [Hassani, 2020] learns node and graph embeddings by using a diffusion process to generate a second structural view of the graph which along with the regular view are sub-sampled and fed to two dedicated GNNs followed by a shared MLP to learn node representations. These representations are then passed to a graph pooling layer and then another shared MLP to learn the graph representations. A discriminator then contrasts the node embeddings from one graph with those of another view and vice versa.

Laplacian Smoothing Approaches These kinds of approaches learn representations via conducting simple Laplacian smoothing operations. The generic formula for obtaining graph embeddings $\mathbf{H}$ through Laplacian smoothing is

$$\mathbf{H} \leftarrow f(\mathbf{A}; p)\mathbf{X}, \tag{2.17}$$

where p is a hyper-parameter that controls the order of propagation. The difference between different models stems from the choice of the transformation f. For example, in [Zhang,

2019], we have that

$$f_{\mathrm{AGC}}(\mathbf{A};p) = \left[(\mathbf{D}+\mathbf{I})^{-1/2}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-1/2}\right]^{p}. \tag{2.18}$$

On the other hand, in [Zhu, 2021], in order to deal with over-smoothing, it is defined based on a Markov diffusion kernel as

$$f_{\mathrm{S^2GC}}(\mathbf{A};p) = \frac{1}{p}\sum_{i=0}^{p}\left[(\mathbf{D}+\mathbf{I})^{-1/2}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-1/2}\right]^{i}. \tag{2.19}$$

Different definitions of f correspond to different filters being applied on the graph signal.

3.3. Multi-view Attributed Graph Clustering

Multi-view clustering is a technique used in machine learning and data mining for grouping a set of objects, or data points, into clusters based on multiple feature sets, or "views," of the data. The goal of multi-view clustering is to take advantage of the complementary information provided by different views in order to improve the performance of the clustering algorithm. For example, if an image dataset has multiple features, such as color histograms, shape, texture features etc, clustering these features separately will be sub-optimal in comparison to multi-view clustering.

Multi-view graph clustering refers to the problem of grouping nodes in a graph into clusters based on multiple feature sets or "views" of the graph data. Each view can be represented as a different graph and can have different characteristics, such as different connectivity patterns or edge weights. There are various ways that multi-view graph clustering can be performed, such as by combining the multiple feature sets into a single graph, by aligning the multiple views, or by fusing the multiple views in some way. Various algorithmic techniques have been developed in the literature to solve this problem. For example, [Kumar, 2011] introduced co-regularized spectral clustering where the idea is to construct a Laplacian matrix for each view of the graph data and optimize the sum of spectral clustering objectives. The solutions which are the eigenvectors are regularized with an additional term that encourages agreement between the eigenvectors computed across the different views.

Depending on whether the data is represented as multiple graphs with one feature set, one graph with multiple feature sets, or a combination of both:

Multiple graphs with one feature set In this scenario, the data is represented as multiple graphs, each with the same set of vertex or edge attributes. Clustering methods in this scenario would focus on finding similar patterns in the graph structures across the different views.

One graph with multiple feature sets In this scenario, the data is represented as a single graph, but with multiple sets of vertex or edge attributes. Clustering methods in this scenario would focus on finding similar patterns in the attributes across the different views.

Multiple graphs with multiple feature sets In this scenario, the data is represented as multiple graphs, each with its own set of vertex or edge attributes. Clustering methods in this scenario would focus on finding similar patterns in both the graph structures and the attributes across the different views.

Part II

Contributions

Chapter 3

Bipartite Graph Clustering via Optimal Transport

Objective

This chapter was published as [Fettal, 2022b]. Our goal was to create a more efficient and effective optimal transport-based approach for the clustering of bipartite graphs than the ones that existed in the literature.

Contents

1. Introduction

Let $G = (U, V, E)$ be a *bipartite graph*, which is a graph whose vertices can be divided into two disjoint sets $U = \{1, 2, \ldots, |U|\}$ with $|U| = n$, $V = \{1, 2, \ldots, |V|\}$ with $|V| = d$ and the set of edges E where each edge connects a vertex of U to a vertex of V. The adjacency matrix for this type of graph has the following structure

$$\mathbf{A} = \begin{pmatrix} \mathbf{0}_{n\times n} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{0}_{d\times d} \end{pmatrix} \tag{3.1}$$

where $\mathbf{B}$ of size $n \times d$ is called the *biadjacency matrix* of G, its rows and columns corresponding to the two sets of vertices; each entry represents an edge between a row and a column. *Biclustering* (or *Co-clustering*) is the extension of clustering to this type of graph. Following [Hartigan, 1972], several biclustering models have attempted to solve the problem by viewing $\mathbf{B}$ as a two-mode matrix and searching for a simultaneous partition of its rows and columns [Dhillon, 2001]. In this way, biclustering seeks to reveal subsets of U which exhibit a similar behaviour across a subset of V in matrix $\mathbf{B}$.

Biclustering has been used in a number of different contexts. [Eisen, 1998] used microarray data to find relations between genes and conditions, finding that genes with similar functions often cluster together. [Harpaz, 2011] applied this paradigm to data from the US Food and Drug Administration reporting system in order to identify groups of drugs with adverse effects. [Dolnicar, 2012] used it to find market segments among tourists so as to enable more effective targeted marketing. There have been various other applications [Dhillon, 2001; Templin, 2010; Gu, 2008].

Several solutions to the biclustering problem have been proposed in the literature (see [Govaert, 2013]). [Dhillon, 2003] used an information-theoretic approach to solve the problem by minimizing the difference in mutual information between $\mathbf{B}$ and a summary matrix; they implicitly assume that the data points are generated from a Poisson latent block model [Govaert, 2018]. [Barber, 2007] adapted classical modularity to bipartite networks and then used it to identify modules within them. [Wang, 2011] proposed a biclustering paradigm based on nonnegative matrix tri-factorization of the biadjacency matrix.

Recently, *Optimal Transport* (OT) has taken the machine learning community by storm. OT has helped to solve a variety of data mining problems, and biclustering is no exception. [Laclau, 2017b] proposed two models for biclustering: a first model, CCOT, which does co-clustering based on the scaling vectors obtained by applying the Sinkhorn-Knopp algorithm on a square subsampled version of matrix $\mathbf{B}$, and a second model, CCOT-GW, which uses scaling vectors obtained by computing entropic Gromov-Wasserstein barycenters, and which does not require subsampling. Then came [Titouan, 2020], where the authors did biclustering by minimizing a new metric, COOT, which generalizes the Gromov-Wasserstein distance between $\mathbf{B}$ and a summary matrix, similarly to what was done in [Dhillon, 2003]. More specifically, they proposed two new metrics: COOT, together with an entropically regularized metric COOT$_\lambda$. However, both [Laclau, 2017b] and [Titouan, 2020] have certain drawbacks.

First, both algorithms do not tackle the biclustering from the beginning; the co-clusters are deduced at the convergence. Thereby biclustering is a consequence and not a main goal. Secondly, they suffer from high computational complexity; CCOT and CCOT-GW also consume large amounts of memory. Finally, we will see that these algorithms are not suited to dyadic sparse data.

In this work, while integrating the biclustering objective from the beginning, we propose a generic framework for biclustering through optimal transport, which generalizes some previous biclustering approaches. We propose two efficient methods for solving this problem: one that gives an almost hard biclustering, and another that gives a *fuzzy* or *soft* biclustering through entropic regularization. These methods outperform other optimal transport biclustering models, in terms of both document and term clustering, on several regular and large scale datasets, while being more computationally and memory efficient. We emphasize once again that the approach we propose is specifically tailored to datasets consisting of dyadic data.

2. Methodology

Notations. In what follows, $\Delta^n = \{\mathbf{p} \in \mathbb{R}_+^n | \sum_{i=1}^n p_i = 1\}$ denotes the n-dimensional standard simplex. $\Pi(\mathbf{w}, \mathbf{v}) = \{\mathbf{Z} \in \mathbb{R}_+^{n \times k} | \mathbf{Z}\mathbf{1} = \mathbf{w}, \mathbf{Z}^\top \mathbf{1} = \mathbf{v}\}$ denotes the transportation polytope, where $\mathbf{w} \in \Delta^n$ and $\mathbf{v} \in \Delta^k$ are the marginals of the joint distribution $\mathbf{Z}$ and $\mathbf{1}_n$ is a vector of ones. Matrices are denoted with uppercase boldface letters, and vectors with lowercase boldface letters. For a matrix $\mathbf{M}$, its i-th row is $\mathbf{m}_i$ and its j-th column is $\mathbf{m}'_j$ We have that $\|.\|_0$ is the 0-norm which returns the number of nonzero elements of its argument.

2.1. Preliminaries

We first need to introduce exact discrete OT and its entropically regularized counterpart, and show how biclustering can be posed as an integer program.

Discrete OT as a linear program. The goal of discrete optimal transport is to find a minimal cost transport plan between a source probability distribution $\mathbf{w}$ and a target distribution $\mathbf{v}$. Here we are interested in the discrete case of the Kantorovich formulation of OT, that is

$$\mathrm{OT}(\mathbf{M}, \mathbf{w}, \mathbf{v}) \triangleq \min_{\mathbf{Z} \in \Pi(\mathbf{w}, \mathbf{v})} \langle \mathbf{M}, \mathbf{Z} \rangle \tag{3.2}$$

where $\mathbf{M} \in \mathbb{R}^{n \times k}$ is the cost matrix, and m_{ij} quantifies the effort needed to transport a probability mass from $\mathbf{w}_i$ to $\mathbf{v}_j$.

Discrete entropy regularized OT. It has been suggested in the literature [Cuturi, 2013; Chizat, 2020] that the use of a regularization such as entropic regularization can lead to better computational and statistical efficiency.

$$\mathrm{OT}_\lambda(\mathbf{M}, \mathbf{w}, \mathbf{v}) \triangleq \min_{\mathbf{Z} \in \Pi(\mathbf{w}, \mathbf{v})} \langle \mathbf{M}, \mathbf{Z} \rangle - \lambda H(\mathbf{Z}) \tag{3.3}$$

where H is the entropy defined as $H(\mathbf{Z}) \triangleq -\sum_{i,j} z_{ij} \log z_{ij}$ and λ controls the strength of regularization. The computational efficiency comes from the fact that the unique solution of this problem is of the structure $\mathbf{Z} := \mathtt{diag}(\mathbf{a}) \exp(-\mathbf{M}/\lambda) \mathtt{diag}(\mathbf{b})$, a rescaled elementwise negative exponential of the cost $\mathbf{M}$, where $\mathbf{a}$ and $\mathbf{b}$ are scaling vectors. These vectors can be found efficiently using the Sinkhorn-Knopp algorithm.

Biclustering as an integer program. The *Block seriation* problem [Marcotorchino, 1987] consists in finding two permutation matrices, one for the rows and one for the columns s.t. dense blocks appear along the diagonal of the permuted matrix. A possible definition of the block seriation problem is as follows: given a matrix $\mathbf{B} \in \mathbb{R}^{n \times d}$ s.t b_{ij} gives the strength of the association between row i and column j (such as in the case of a biadjacency matrix, for example), we have

$$\max_{\mathbf{C}} \quad \sum_{i,j} b_{ij} c_{ij} \tag{3.4}$$

$$\begin{aligned}
\text{subject to} \quad & \forall\, i, j \quad c_{ij} \in \{0, 1\} \\
& \forall\, j \quad \textstyle\sum_i c_{ij} \geq 1 \\
& \forall\, i \quad \textstyle\sum_j c_{ij} \geq 1
\end{aligned}
\qquad
\begin{aligned}
& \forall\, i, j, i', j' \quad c_{ij} + c_{ij'} + c_{i'j'} - c_{i'j} \leq 2 \\
& \qquad\qquad\quad\; c_{i'j'} + c_{i'j} + c_{ij} - c_{ij'} \leq 2 \\
& \qquad\qquad\quad\; c_{i'j} + c_{ij} + c_{ij'} - c_{i'j'} \leq 2 \\
& \qquad\qquad\quad\; c_{ij'} + c_{i'j'} + c_{i'j} - c_{ij} \leq 2
\end{aligned}$$

A solution $\mathbf{C}$ is a block diagonal matrix up to a permutation of its rows and columns. The block seriation problem is an integer programming problem that is NP-hard. One approach for solving this problem uses a simplified version where a rank constraint $\mathtt{rank}(\mathbf{C}) \leq k$ is added for k the number of desired biclusters. Integrating this constraint into (3.4), we can define a new problem by low-rank factorization of $\mathbf{C}$, i.e. $\mathbf{C} = \mathbf{Z}\mathbf{W}^\top$, which we formulate as

$$\max_{\substack{\mathbf{Z} \in \Gamma(n,k) \\ \mathbf{W} \in \Gamma(d,k)}} \sum_{i,j,h} b_{ij} z_{ih} w_{jh} \tag{3.5}$$

where $\Gamma(n, k) = \{\mathbf{Z} \in \{0, 1\}^{n \times k} \mid \mathbf{Z}\mathbf{1} = \mathbf{1}\}$ is the set of hard partitions of dimension $n \times k$. A simple heuristic for solving this problem involves alternatingly solving for $\mathbf{Z}$ given $\mathbf{W}$, and vice-versa, using classical clustering algorithms, before identifying biclusters through the rearranged matrix $\mathbf{C}$, which displays a block diagonal structure, as shown in figure 3.1a. The biclusters are identified by grouping together the rows and columns that form a block along the diagonal.

2.2. Biclustering using Optimal Transport

Here we propose a new biclustering problem based on block seriation and optimal transport. For this purpose we first define what we term an *anti-adjacency matrix*. Note that a similar concept has been discussed in [Wang, 2018a].

Definition 1 (Anti-adjacency matrix)**.** *Given a graph characterized by an adjacency matrix* $\mathbf{A}$*, we have a corresponding anti-adjacency matrix* $\overline{\mathbf{A}}$ *s.t.* $\overline{a}_{ij}$ *quantifies the discrepancy between nodes* i *and* j*.*

We consider a bipartite graph characterized by its biadjacency matrix $\mathbf{B} = (b_{ij}) \in \mathbb{R}^{n \times d}$. The rows of $\mathbf{B}$ are endowed with weights $\mathbf{w} \in \Delta^n$ and its columns with weights $\mathbf{v} \in \Delta^d$. We also consider a row exemplar distribution $\mathbf{r} \in \Delta^r$ and a column exemplar distribution $\mathbf{c} \in \Delta^c$. Depending on the availability of *a priori* information about the data, these weight vectors can be set to uniform distributions.

Now let its anti-biadjacency matrix be $\overline{\mathbf{B}} = L(\mathbf{B})$, where $L : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ means that b_{ij}, the association between node i and node j, is transformed into a discrepancy measure $L(\mathbf{B})_{ij}$. Thus, we define the optimal transport block seriation problem as the following bilinear program

$$\mathrm{BCOT}(\mathbf{w}, \mathbf{v}, \mathbf{r}, \mathbf{c}) \triangleq \min_{\substack{\mathbf{Z} \in \Pi(\mathbf{w}, \mathbf{r}) \\ \mathbf{W} \in \Pi(\mathbf{v}, \mathbf{c})}} \sum_{i,j,k} L(\mathbf{B})_{ij} z_{ik} w_{jk} \equiv \min_{\substack{\mathbf{Z} \in \Pi(\mathbf{w}, \mathbf{r}) \\ \mathbf{W} \in \Pi(\mathbf{v}, \mathbf{c})}} \left\langle L(\mathbf{B}), \mathbf{Z}\mathbf{W}^\top \right\rangle \tag{3.6}$$

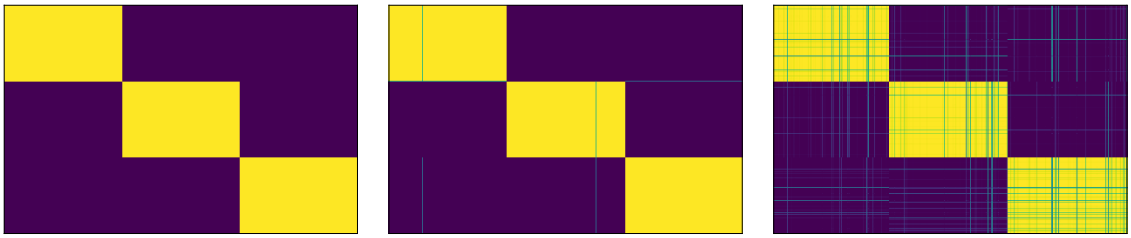
where $\mathbf{Z}$ is a transport plan (or coupling) between between the row distribution $\mathbf{w}$ and the row exemplar distribution $\mathbf{r}$, and similarly for $\mathbf{W}$ w.r.t. the column distribution $\mathbf{v}$ and the column exemplar distribution $\mathbf{c}$.

Inducing a biclustering via BCOT. We will now show how to obtain a partition of the rows and the columns given a solution pair $(\mathbf{Z}, \mathbf{W})$. In what follows our aim is to identify an *almost-hard clustering* couple for rows and columns from the couplings $\mathbf{Z}$ and $\mathbf{W}$.

Definition 2 (h-almost hard clustering)**.** *We define an* h*-almost hard clustering as a clustering whose assignment matrix is* $\mathbf{C} \in \mathbb{R}^{n \times k}$ *s.t.* $\|\mathbf{C}\|_0 = n + h$ *and for each row* $\boldsymbol{c}$ *of* $\mathbf{C}$ *we have that* $\|\boldsymbol{c}\|_0 > 0$*. When* $h = 0$*, we obtain a standard hard clustering with one non-zero element per row.*

Proposition 1. *For* $\mathbf{w}$*,* $\mathbf{v}$*,* $\mathbf{r}$ *and* $\mathbf{c}$ *containing no zeros, there exists an optimal pair of coupling matrices* $\mathbf{Z}$ *and* $\mathbf{W}$ *that are* h*-almost hard clusterings with* $h \in \{0, \ldots, k-1\}$*. Furthermore, when* $n = k$ *(resp.* $d = k$*) and* $\mathbf{w} = \mathbf{r}$ *(resp.* $\mathbf{v} = \mathbf{c}$*), this* $\mathbf{Z}$ *(resp.* $\mathbf{W}$*) becomes a hard clustering, i.e.,* $\mathbf{Z} \in \Gamma(n, n)$ *(resp.* $\mathbf{W} \in \Gamma(d, d)$*).*

Proof. Problem (3.2) is a bounded linear program since $\Pi(\mathbf{w}, \mathbf{v})$ is a polytope i.e. a bounded polyhedron. The fundamental theorem of linear programming states that if the feasible set is non-empty then the solution lies in the extremity of the feasible region. This means that a solution $\mathbf{Z}$ to problem (3.2) is an extreme point of $\Pi(\mathbf{w}, \mathbf{v})$. We have that the extreme points of $\Pi(\mathbf{w}, \mathbf{v})$ can have at most $n + d - 1$ nonzero elements. To prove this we have to show that the bipartite graph induced by biadjacency matrix $\mathbf{Z}$, the solution to the optimal transport problem has no cycles. The maximum number of edges in an acyclic graph is $|V| - 1$ where $|V|$ is the number of nodes in the graph. Since the number of edges in the bipartite graph induced by biadjacency matrix $\mathbf{Z}$ is $n + d - 1$, the matrix $\mathbf{Z}$ can not have more than $n + d - 1$ nonzero entries. For a detailed proof see proposition 3.3 in [Peyré, 2017].



(a) Block Seriation. (b) BCOT. (c) BCOT$_\lambda$.

Figure 3.1: Biclusters formed using three different methods on the Pubmed dataset. Classical block seriation results in a biclustering that is hard. BCOT results in a biclustering that is almost hard with few nonzero entries outside the main block diagonal. BCOT$_\lambda$ results in a soft biclustering with many nonzero elements outside the block diagonal.

We also have to show that for probability measures $\mathbf{w}$ and $\mathbf{v}$ that have no zero probability events, there is at minimum $\max(n, d)$ number of nonzero elements in $\mathbf{Z}$. This is straightforward since $\mathbf{w}$ and $\mathbf{v}$ contain no zeros, there will always be at least one nonzero element in every row and column of $\mathbf{Z}$ that represents some transfer of mass between elements of $\mathbf{w}$ and $\mathbf{v}$.

BCOT is a bilinear program that has a finite global solution which means that there exists at least one optimal solution pair $(\mathbf{Z}, \mathbf{W})$ such that $\mathbf{Z}$ is an extreme point of $\Pi(\mathbf{w}, \mathbf{r})$ and $\mathbf{W}$ is an extreme point of $\Pi(\mathbf{v}, \mathbf{c})$ (theorem 1 in [Konno, 1976]).

We then have that, For BCOT, $\mathbf{Z}$ has at most $n + k - 1$ and at least $\max(n, k) = n$ nonzero entries and that $\mathbf{W}$ has at most $d + k - 1$ and at least $\max(d, k) = d$ elements which are both h-almost hard clusterings with $h \in \{0, \ldots, k - 1\}$.

When $n = k$ and $\mathbf{w} = \mathbf{r}$, the solution $\mathbf{Z}$ is a permutation matrix (up to a constant factor) and the number of nonzero elements in it is exactly n which means that it represents a hard partition $\mathbf{Z} \in \Gamma(n, n)$. The proof is the same for $\mathbf{W}$. □

This means that the solutions are already almost a hard partition of the data, since $k << n, d$. To obtain a final hard clustering in the strict sense, we assign each row (resp. column) to the one corresponding to the row of $\mathbf{Z}$ (resp. $\mathbf{W}$) with the largest value. This should not significantly change the structure of the solution. Figure 3.1b provides an illustration: here we see the block diagonal structure generated by the product of the two coupling matrices $\mathbf{C} = \mathbf{Z}\mathbf{W}^\top$, with its similarity in appearance to the biclustering produced by the hard block seriation 3.1a, apart from a few nonzero entries off the block diagonal that are hard to see immediately.

Intuition for BCOT. To explain the intuition behind the proposed approach we need to look at how the problem is solved. The optimization procedure as described in algorithm 1 consists in alternating between the computation of an optimal transport plan $\mathbf{Z}$ given $\mathbf{W}$ and vice versa. As regards solving for $\mathbf{Z}$ given $\mathbf{W}$, the problem can be rewritten as

$$\mathrm{BCOT}(\mathbf{w}, \mathbf{v}, \mathbf{r}, \mathbf{c}) \equiv \min_{\mathbf{Z} \in \Pi(\mathbf{w}, \mathbf{r})} \langle L(\mathbf{B})\mathbf{W}, \mathbf{Z} \rangle \,. \tag{3.7}$$

This is an optimal transport problem with $L(\mathbf{B})\mathbf{W}$ as the cost matrix. The resulting transport plan $\mathbf{Z}$ can be seen as a kind of row cluster assignment matrix: if $z_{ih} > 0$, then row i is assigned to cluster h. The same holds for $\mathbf{W}$, which can be seen as a column cluster assignment matrix. This also means that since $L(\mathbf{B})$ is the dissimilarity between the rows and the columns, then the cost matrix $L(\mathbf{B})\mathbf{W}$ represents the dissimilarity between rows and row exemplars (or representatives or centroids). In particular, $L(\mathbf{B})_i\mathbf{w}_h$ is the dissimilarity or cost of probability mass transportation between row i and row cluster exemplar h. The reasoning is the same for the columns and the optimal coupling $\mathbf{W}$.

Low-rank optimal transport. Biclustering is the main purpose of the approach we proposed, but there is another interesting use case.

Proposition 2. *For equal target row and column representative distributions, i.e.,* $\mathbf{r} = \mathbf{c}$*, and containing no zero entries, then given a solution pair* $\mathbf{Z}$ *and* $\mathbf{W}$ *to BCOT, the matrix* $\mathbf{Q} = \mathbf{Z}\,\mathtt{diag}(1/\mathbf{r})\mathbf{W}^\top$ *is an approximation of the optimal transport plan that is a solution to problem (3.2) and whose rank is at most* $\min(\mathtt{rank}(\mathbf{Z}), \mathtt{rank}(\mathbf{W}))$.

Proof. From linear algebra, we have that $rank(\mathbf{Q}) \leq \min(rank(\mathbf{Z}), rank(\mathtt{diag}(1/\mathbf{r})), rank(\mathbf{W}))$. Since $\mathbf{Z}$ and $\mathbf{W}$ cannot have a rank greater than k due to their dimension, and since $\mathtt{diag}(1/\mathbf{r})$ is a full rank matrix due to the assumption that $\mathbf{r}$ has no zero entries, we then have that $rank(\mathbf{Q}) \leq \min(rank(\mathbf{Z}), rank(\mathbf{W}))$.

For a proof that $\mathbf{Q}$ is indeed a valid transport plan i.e. $\mathbf{Q} \in \Pi(\mathbf{w}, \mathbf{v})$, we refer the reader to proposition 2.2 in [Peyré, 2017]. □

Some recent works [Forrow, 2019; Scetbon, 2021] have suggested that this kind of low-rank regularization is preferable to entropic regularization as regards certain aspects. For example, the rank parameter is easier to select, since it has simple bounds (an integer between 1 and n). This may be contrasted with the regularization strength λ in the Sinkhorn algorithm, which is continuous.

2.3. Fuzzy Biclustering via Regularized Optimal Transport

As previously mentioned, using entropic regularization may be interesting because of its various useful features including statistical and computational efficiency. However, another feature of entropic regularization is that the optimal couplings $\mathbf{Z}$ and $\mathbf{W}$ are dense matrices as a consequence of the structure of the optimal solution of entropically regularized OT problems. We formulate the problem as follows

$$\mathrm{BCOT}_\lambda(\mathbf{w}, \mathbf{v}, \mathbf{r}, \mathbf{c}) \triangleq \min_{\substack{\mathbf{Z}\in\Pi(\mathbf{w},\mathbf{r}) \\ \mathbf{W}\in\Pi(\mathbf{v},\mathbf{c})}} \left\langle L(\mathbf{B}), \mathbf{Z}\mathbf{W}^\top \right\rangle - \lambda_{\mathbf{Z}} H(\mathbf{Z}) - \lambda_{\mathbf{W}} H(\mathbf{W}) \tag{3.8}$$

where $\lambda_{\mathbf{Z}}$ and $\lambda_{\mathbf{W}}$ are the regularization parameters.

Fuzzy block seriation. We propose a fuzzy variant of the block seriation problem that allows us by extension to define a fuzzy variant for BCOT using entropic regularization. Let the fuzzy block seriation problem be defined as

$$\max_{\substack{\mathbf{Z}\in\Gamma_s(n,k)\\ \mathbf{W}\in\Gamma_s(d,k)}} \sum_{i,j,h} b_{ij} z_{ih} w_{jh} + \Omega(\mathbf{Z},\mathbf{W}) \tag{3.9}$$

where $\Omega(\mathbf{Z},\mathbf{W})$ is some regularization term introduced to make the partition matrices $\mathbf{Z}$ and $\mathbf{W}$ dense (for example, entropic regularization or low-rank constraints), and $\Gamma_s(n,k) = \{\mathbf{Z} \in \mathbb{R}_+^{n\times k} | \mathbf{Z1} = \mathbf{1}\}$ is the set of fuzzy partitions. Intuitively, for a solution pair $(\mathbf{Z},\mathbf{W})$, up to a constant factor, each entry in the block seriation matrix $\mathbf{C} = \mathbf{Z}\mathbf{W}^\top$ can be seen as the probability of its corresponding row and column belonging to the same bicluster i.e. $c_{ij} = \mathbf{z}_i\mathbf{w}_j = \sum_{h=1}^r z_{ih}w_{jh} = p(\mathbf{b}_i, \mathbf{b}'_j) = \sum_{h=1}^r p(\mathbf{b}_i, \mathbf{b}'_j \in h)$.

It is easy to see how problem (3.9) is related to problem (3.8) and that the couplings corresponding to solutions of the problem give the probability that the different rows and columns belong to the same biclusters. Figure 3.1c shows biclusters produced by the solutions of BCOT_λ. Similarly to BCOT, a block diagonal structure is formed. However, there are also several off-block diagonal nonzero entries that represent the probabilities of the row-column pairs belonging to the same biclusters.

3. Links to Existing Work

3.1. Modularity Maximization in Bipartite Graphs.

The model presented in [Barber, 2007] is able to co-cluster binary and contingency matrices by directly maximizing an adapted version of the modularity measure traditionally used for networks. The criterion that it optimizes is

$$\max_{\substack{\mathbf{Z}\in\Gamma(n,k)\\ \mathbf{W}\in\Gamma(d,k)}} \sum_{i,j,h} z_{ih} w_{jh} \left(b_{ij} - \frac{b_{\cdot j} b_{i\cdot}}{b_{\cdot\cdot}} \right). \tag{3.10}$$

By setting $L(\mathbf{B}) = -(\mathbf{B} - \frac{1}{b_{\cdot\cdot}}\mathbf{B}\mathbf{1}\mathbf{1}^\top\mathbf{B})$, this problem becomes equivalent to ours; the difference is in the constraints on $\mathbf{Z}$ and $\mathbf{W}$.

3.2. Modularity-Based Sparse Soft Graph Clustering.

In [Hollocou, 2019], the authors proposed a fuzzy variant of the above problem (although in the context of traditional clustering rather than biclustering). Solving the problem gives, for each element of the dataset, a probability of that element belonging to a given cluster. Our proposed entropic regularization variant represents a kind of extension of this problem to bipartite graphs.

3.3. Directional Co-clustering with a Conscience.

The model in [Salah, 2017a; Affeldt, 2021] makes use of the block von Mises-Fisher mixture model for co-clustering directional data on the unit-sphere. It optimizes the following criterion:

$$\max_{\substack{\mathbf{Z}\in\Gamma(n,k)\\ \mathbf{W}\in\Gamma(d,k)}} \sum_{i,j,h} \frac{1}{\sqrt{z_{.h}w_{.h}}} z_{ih}w_{jh}b_{ij}. \tag{3.11}$$

In our formulation, if we define $L(\mathbf{B}) = -\mathbf{B}$ and apply cluster size normalization on the optimal transport plans $\tilde{\mathbf{Z}} = \mathbf{Z}\texttt{diag}(\mathbf{Z}^\top\mathbf{1})^{-1/2}$ and $\tilde{\mathbf{W}} = \mathbf{W}\texttt{diag}(\mathbf{W}^\top\mathbf{1})^{-1/2}$ after computing $\mathbf{Z}$ and $\mathbf{W}$ respectively in algorithm 1, we obtain a more general version of the algorithm proposed by the authors for solving problem (3.11).

3.4. Bipartite Correlation Clustering.

In the case where the cost function results in a complete bipartite graph with '+' and '-' edges with a function

$$L(\mathbf{B})_{ij} = \begin{cases} -1 & \text{if } b_{ij} > 0 \\ +1 & \text{otherwise} \end{cases} \tag{3.12}$$

we get what is known as Bipartite Correlation Clustering [Ailon, 2012]. The solution to this problem maximizes the number of agreements, i.e. the number of all '+' edges within clusters plus all '-' edges distributed across clusters.

4. Optimization and Complexity

Optimization. Since the block seriation problem is NP-hard, computing an exact solution is prohibitive. An efficient and widely used heuristic for solving these kinds of problems involves the use of block coordinate descent, where row assignments are computed for fixed column assignments, and then vice versa, in alternation. We express the proposed algorithm in pseudo-code as algorithm 1. At each iteration we solve two intermediate optimal transport problems with cost matrices of dimensions $n \times k$ and $d \times k$, since $\mathbf{B}$ is generally sparse, and L can be defined such that $L(\mathbf{B})$ retains a similarly sparse structure. The computation of the intermediate cost matrices $L(\mathbf{B})\mathbf{W}$ and $L(\mathbf{B})^\top\mathbf{Z}$ is reasonably efficient. We also observed that the algorithm does not need many iterations to converge, as shown in figure 3.2, be it for BCOT or BCOT$_\lambda$.

Proposition 3. *The computational complexity of the* `BCOT` *algorithm 1 when using an exact OT solver is* $\mathcal{O}\left(tk\|\mathbf{B}\|_0 + tnk(n+k)\log(n+k) + tdk(d+k)\log(d+k)\right)$*, and when using entropic regularization the complexity is* $\mathcal{O}(tk\|\mathbf{B}\|_0 + tkn + tkd)$*, where* t *is the number of iterations.*

Proof. We suppose that $L(\mathbf{B})$ is a sparse matrix with the same number of nonzero entries as

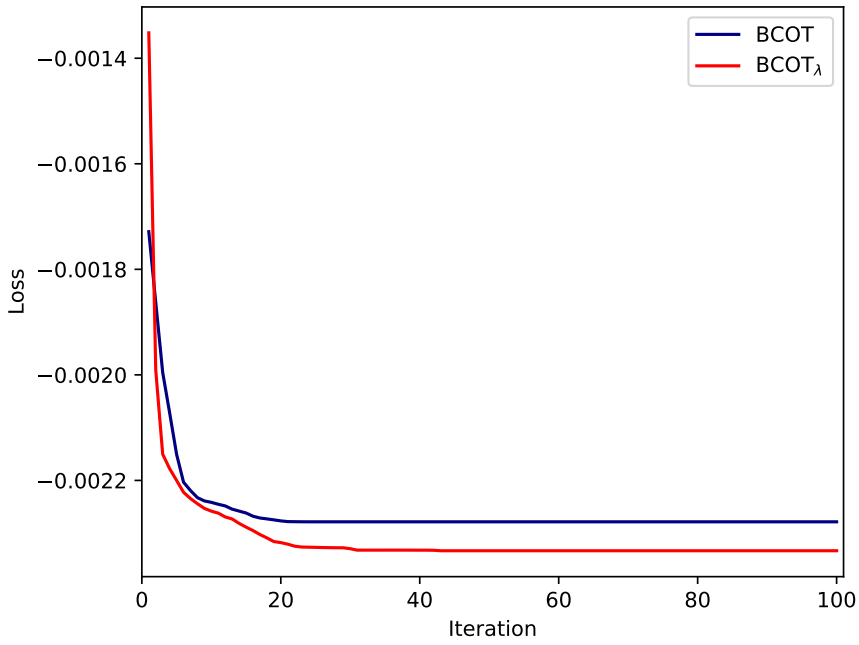


Figure 3.2: Loss for BCOT and BCOT$_\lambda$ on Pubmed.

Algorithm 1: `BCOT`

Input : $\mathbf{B}$ bi-adjacency matrix, $\mathbf{w}$ and $\mathbf{v}$ row and column weights, $\mathbf{r}$ and $\mathbf{c}$ row and column exemplar distributions

Output: π^r, π^c row and column partitions

$\mathbf{W} \leftarrow \mathbf{W}_{init}$;

while *not converged* **do**

  $\mathbf{Z} \leftarrow \arg \texttt{OT}\left(L(\mathbf{B})\mathbf{W}, \mathbf{w}, \mathbf{r}\right)$;

  $\mathbf{W} \leftarrow \arg \texttt{OT}\left(L(\mathbf{B})^\top \mathbf{Z}, \mathbf{v}, \mathbf{c}\right)$;

end

Generate π^r, π^c from $\mathbf{Z}$ and $\mathbf{W}$;

$\mathbf{B}$. The complexity of computing $L(\mathbf{B})\mathbf{W}$ and $L(\mathbf{B})\mathbf{W}$ in the BCOT algorithm is $\mathcal{O}(k\|\mathbf{B}\|_0)$.

The optimal transport problem can be formulated and solved as the Earth Mover's Distance (EMD) problem using any minimum-cost flow problem algorithm, such as one of the many variants of the network simplex algorithm. The authors in [Orlin, 1997] proposed an algorithm for the network simplex in $\mathcal{O}(|V||E|\log|V|)$, where $|V|$ is the number of nodes and $|E|$ is the number of edges in the network. In our case, when solving the EMD for $\mathbf{Z}$ and cost matrix $L(\mathbf{B})\mathbf{W}$, the number of nodes is $|V| = n + k$ and the number of edges is $|E| = nk$, which means that the complexity of the operation is $\mathcal{O}(nk(n+k)\log(n+k))$. When computing the optimal transport plan for $\mathbf{W}$, for cost matrix $L(\mathbf{B})^\top\mathbf{Z}$, the complexity is $\mathcal{O}(dk(d+k)\log(d+k))$. The overall complexity of the BCOT algorithm is then $\mathcal{O}(k\|\mathbf{B}\|_0) + tnk(n+k)\log(n+k) + tdk(d+k)\log(d+k))$

When using entropic regularization the complexity is smaller, since computing the optimal transport plan requires only a transformation of the inputs matrix, which takes $\mathcal{O}(nk)$ in the

$\mathbf{Z}$ computation step and $\mathcal{O}(dk)$ for $\mathbf{W}$. The ensuing application of the Sinkhorn-Knopp algorithm on the transformed matrices has complexities $\mathcal{O}(tnk)$ and $\mathcal{O}(tdk)$ for $\mathbf{Z}$ and $\mathbf{W}$ respectively, where t is the number of iterations necessary. The overall complexity of BCOT$_\lambda$ is then $\mathcal{O}(k\|\mathbf{B}\|_0) + tnk + tdk)$, here t includes the number of iterations of our algorithm as well as that of Sinkhorn-Knopp.

□

Table 3.1: Computational and spatial complexity of the different OT biclustering approaches. For COOT variants, we report complexities for an euclidean cost matrix. For a generic cost, the time complexity is greater. For simplicity, we suppose that $d \in O(n)$ and that we want a biclustering with the same number of row and column clusters for COOT and CCOT. t denotes the number of iterations and for CCOT, s denotes the number of necessary samplings.

Method	**Spatial complexity**	**Time complexity**
CCOT	$O(n^2)$	$O(sn^3)$
CCOT-GW	$O(n^2)$	$O(n^3)$
COOT*	$O(nk)$	$O((n+k)nk + k^2n + t(n+k)nk\log(n+k))$
COOT$^*_\lambda$	$O(nk)$	$O((n+k)nk + k^2n + tnk)$
BCOT	$O(nk)$	$O(k\|\mathbf{B}\|_0 + t(n+k)nk\log(n+k))$
BCOT$_\lambda$	$O(nk)$	$O(k\|\mathbf{B}\|_0 + tnk)$

In table 3.1, we report the computational and spatial complexities of the different biclustering approaches. Our model has the same spatial complexity as the COOT variants and a better complexity than CCOT variants. As regards the computational complexity, our model should in most cases be faster with sparse data, and our experiments support this conjecture. For reproducibility, we publicly release our code [1].

5. Experiments

We ran experiments using term-document matrices. The benefit of using biclustering on this kind of data is that the resulting biclusters contain both documents and the words that characterize them, which is helpful in interpreting the clustering of the documents. Additional experiments over synthetic and gene expression data were also conducted.

5.1. Datasets

We evaluate BCOT in relation to six benchmark document-term datasets: ACM, DBLP, PubMed, Wiki, Ohscal, and 20 Newsgroups. Their characteristics are shown in Table 3.2. ACM, DBLP, Pubmed and Wiki are attributed networks from which we use only the node-level features that correspond to term-document matrices. We also selected the Ohscal col-

[1] https://github.com/chakib401/BCOT

lection and 20 Newsgroups as large-scale document-term matrices to serve as computational efficiency benchmarks.

Table 3.2: Characteristics of the datasets.

Dataset	**#Documents**	**#Terms**	**#Document clusters**	**Sparsity** (%)
ACM [Fan, 2020]	3025	1870	3	95.52
DBLP [Fan, 2020]	4057	334	4	96.4
PubMed [Sen, 2008]	19717	500	3	89.98
Wiki [Yang, 2015]	2405	4973	17	86.99
Ohscal [Hersh, 1994]	11162	11465	10	99.47
20 Newsgroups [Lang, 1995]	18846	14390	20	99.41

5.2. Experimental Setup

In our experiments we define the loss function as $L(\mathbf{B}) = -c\mathbf{B}$, where c is selected from $\{1, k, d, n\}$. For BCOTλ, the regularization parameter lambda is selected from $\{10^{-4}$,10^{-3}, 10^{-2}, 10^{-1}, 1, 10$\}$. The best hyper-parameters are those that minimize the number of empty clusters. In the case of ties, we select according to the value of the Davies-Bouldin index of the partition [Davies, 1979]. Random restarts are not used for any of the algorithms, including k-means. We use the implementation provided by the authors for CCOT, CCOT$_\lambda$ and CCOT-GW. The code for CCOT was not available, and so we had to implement it based on the code for CCOT-GW. All the reported figures are the averages of 10 runs. All the experiments were performed on the same machine with an Intel(R) Xeon(R) CPU and 12GB RAM. For OT solvers we made use of the POT package [Flamary, 2021].

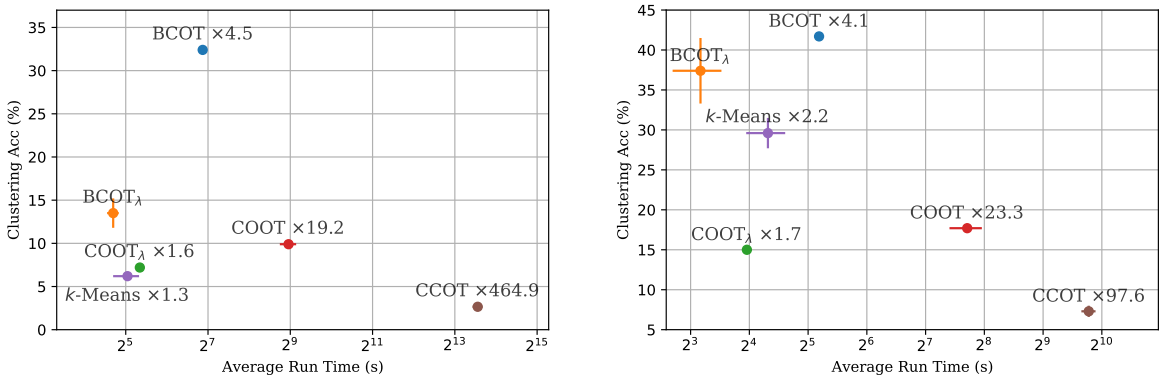
5.3. Document Clustering

Metrics. Here, the evaluation is straightforward, we adopt three popular clustering metrics: clustering accuracy (CA), normalized mutual information (NMI) [Cai, 2008], adjusted rand index (ARI) [Hubert, 1985].

Table 3.3: Document clustering performance on the four datasets. OOM denotes out of memory.

Method	**ACM**			**DBLP**			**PubMed**			**Wiki**		
	CA	NMI	ARI	CA	NMI	ARI	CA	NMI	ARI	CA	NMI	ARI
k-Means	51.1±11.3	13.7±11.2	14.0±10.6	36.9±2.4	10.4±2.0	4.3±2.0	52.3±4.7	18.2±10.5	15.3±10.1	26.0±6.1	18.6±9.3	3.3±2.9
CCOT	12.4±2.0	1.0±0.2	0.4±0.2	28.6±0.5	0.6±0.0	0.4±0.0	32.7±0.2	3.0±0.0	3.1±0.1	10.6±0.5	4.9±0.1	0.6±0.15
CCOT-GW	8.1±0.0	1.5±0.0	0.3±0.0	9.4±0.0	1.7±0.0	0.3±0.0		OOM		10.9±0.0	4.3±0.0	0.48±0.0
COOT*	39.0±0.0	1.9±0.0	2.0±0.0	30.5±1.4	1.4±0.3	1.2±0.3	43.2±1.5	1.7±0.6	1.3±1.5	25.9±1.8	28.7±2.2	12.3±1.7
COOT$_\lambda$	41.5±0.2	1.9±0.1	2.2±0.0	30.6±0.0	0.7±0.0	0.6±0.0	42.4±1.5	1.7±0.5	1.0±1.3	17.2±0.0	1.7±0.0	0.31±0.0
BCOT	**76.6±1.5**	**38.3±2.2**	**43.3±2.6**	**61.5±6.2**	**27.4±4.3**	**28.3±5.5**	53.6±4.5	15.9±1.9	12.9±2.4	49.8±1.5	47.9±1.0	30.6±1.0
BCOT$_\lambda$	76.2±0.6	37.6±0.8	42.4±1.0	59.4±9.9	26.6±7.6	27.2±9.5	**56.5±3.1**	**18.4±1.3**	**15.4±1.8**	**50.8±1.5**	**49.4±0.9**	**31.9±0.8**

Performance. Document clustering results on ACM, DBLP, PubMed and Wiki are given in table 3.3 for the three metrics. In all cases the best result is obtained either by BCOT



(a) 20 Newsgroups.

(b) Ohscal.

Figure 3.3: Accuracy against training time on NG20 and Ohscal. BCOT$_\lambda$ is the fastest and has a competitive level of accuracy. BCOT gives the best accuracy while remaining relatively efficient. The multiplication factors shown for the training times take BCOT$_\lambda$ as the reference (and so, for example, ×4.5 shown for BCOT means that it is approximately 4.5 times slower than BCOT$_\lambda$). We were not able to benchmark CCOT-GW since it failed to scale to these datasets.

or by BCOT$_\lambda$. Moreover, on Wiki, BCOT$_\lambda$ gives competitive results when compared with state-of-the-art attributed graph clustering methods presented in [Fettal, 2022c], despite not having access to the graph structure information in the Wiki citation network.

Efficiency. Figure 3.3 plots the document clustering performance (accuracy against training time) of the different methods on the two large-scale document-term matrices 20 Newsgroup and Ohscal. BCOT offers the best accuracy while BCOT$_\lambda$ is fastest method on both datasets. We see that for both BCOT and COOT, the entropic-regularized versions outspeed their exact counterparts and that CCOT suffers from very high computation times, due mainly to the fact that this method requires pairwise distance matrices to be computed on the rows and columns.

5.4. Term Clustering

Metrics. Unlike document clustering, there is no ground truth partition for terms, so we need to find another way of evaluating term clustering results. One generally acceptable technique is to analyse the semantic coherence of the clusters obtained. To this end we introduce a metric based on *point mutual information* (PMI). PMI is a frequently used information-theoretic metric for quantifying the relationship between pairs of discrete random variable outcomes. The PMI measure was chosen because prior research [Newman, 2009] has shown that it is closely associated with human judgements in determining word relatedness. The PMI between the terms w_i and w_j is calculated as

$$\mathrm{PMI}(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)} \tag{3.13}$$

In the context of term clustering, given the word co-occurrence matrix $\mathbf{K} = \mathbf{B}^\top \mathbf{B}$, the PMI is estimated as in

$$\mathrm{PMI}(w_i, w_j) = \log \frac{k_{..} k_{ij}}{k_{i.} k_{.j}} \tag{3.14}$$

To evaluate a partition of terms $\mathcal{P}$, we propose a metric based on *intra* and *inter* PMI metrics as follows:

$$\mathrm{PMI}_{intra}(P) = \sum_{i \in P} \sum_{j \in P} k_{ij} \tag{3.15}$$

$$\mathrm{PMI}_{inter}(P) = \sum_{i \in P} \sum_{j \notin P} k_{ij} \tag{3.16}$$

In this way, a good clustering should reveal a high intra-cluster semantic relatedness, corresponding to higher PMI values. Using the *intra* and *inter* PMIs, we propose the following *coherence* index

$$\mathrm{coherence}(\mathcal{P}) = \frac{1}{\sum\limits_{P \in \mathcal{P}} |P|} \sum_{P \in \mathcal{P}} |P| \left(\mathrm{PMI}_{intra}(P) - \mathrm{PMI}_{inter}(P) \right). \tag{3.17}$$

Our reasoning is this: the greater the semantic proximity between terms in the same clusters, and the greater the semantic distance between terms in different clusters, the higher the value of *coherence*.

Results. Since there is no ground truth number of term clusters, we use the cluster number estimations produced by CCOT-GW for all the other models so that it is easy to compare coherence values between them. Comparisons based on different numbers of clusters would favor the model using the larger number of clusters. Table 3.4 shows the coherences obtained across the different datasets using our approach, along with those of the baselines. It is clear that BCOT succeeds in capturing more semantics than the other approaches since, whatever the dataset, one or other of the two BCOT variants gives the highest coherence.

Table 3.4: Term clustering performance on the four datasets. OOM denotes out of memory.

Method	**ACM**	**DBLP**	**PubMed**	**Wiki**	**Ng20**	**Ohscal**
k-Means	0.19±0.01	0.05±0.03	0.31±0.18	0.28±0.02	0.28±0.04	0.01±0.02
CCOT	0.03±0.00	-0.07±0.06	0.02±0.01	0.02±0.00	0.05±0.00	0.06±0.00
CCOT-GW	0.08±0.00	0.03±0.00	OOM	0.01±0.00	OOM	OOM
COOT	0.12±0.01	0.07±0.00	0.14±0.01	0.40±0.00	0.43±0.02	0.23±0.01
COOT$_\lambda$	0.21±0.00	0.04±0.00	-0.00±0.00	-0.08±0.00	-0.02±0.00	-0.13±0.00
BCOT	**0.27±0.01**	**0.22±0.04**	0.54±0.03	**0.64±0.01**	**0.79±0.01**	**0.44±0.00**
BCOT$_\lambda$	0.24±0.00	0.16±0.02	**0.57±0.01**	0.62±0.01	0.27±0.01	0.35±0.00

5.5. Gene Clustering

This approach has also been successfully evaluated on microarray data for gene clustering. A microarray database is a repository containing microarray gene expression data.

Datasets. The gene-expression matrices used are the CuMiDa Breast Cancer and Leukemia datasets. Their characteristics are shown in Table 3.5.

Table 3.5: Characteristics of the gene expression datasets.

Dataset	Samples	Genes	k	Sparsity (%)
Breast Cancer [Feltes, 2019]	26	42945	2	0.0
Leukemia [Feltes, 2019]	64	22283	5	0.0

Metrics. The metrics are the same as for document clustering.

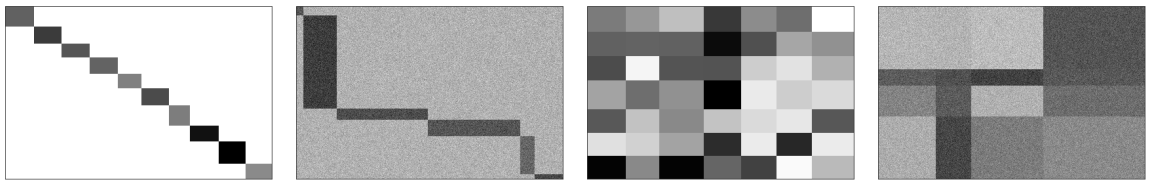
Performance In table 3.6, we report results on the two micro-array datasets, BCOT$_\lambda$ has the best performance on both of them.

Table 3.6: Gene clustering performance on the two microarray datasets.

Method	Breast Cancer			Leukemia		
	CA	NMI	ARI	CA	NMI	ARI
k-means	75.8±18.0	41.9±40.5	31.2±49.0	74.8±7.2	72.1±5.4	50.1±8.3
CCOT		OOM		40.6±0.0	0.0±0.0	0.0±0.0
CCOT-GW		OOM			OOM	
COOT	63.1±5.2	5.4±8.7	-1.2±2.9	36.2±2.7	14.0±3.6	5.4±3.2
COOT$_\lambda$	61.5±0.0	5.4±0.0	2.2±0.0	32.5±3.3	8.7±2.7	-.5±2.1
BCOT	76.9±0.0	37.2±0.0	26.7±0.0	71.2±5.4	59.6±6.9	39.9±6.3
BCOT$_\lambda$	**84.6±0.0**	**48.3±0.0**	**46.0±0.0**	**80.9±3.8**	**70.9±4.1**	**55.3±3.3**

5.6. Co-clustering

Datasets. As datasets with labels along both rows and columns are unavailable, we use synthetic data as in [Laclau, 2017b; Titouan, 2020]. Their structure is shown in figure 3.4, while their characteristics are reported in table 3.7.



(a) A (b) B (c) C (d) D

Figure 3.4: Synthetic datasets rearranged with respect to the true partition.

Metrics. From row π^r and column π^c clusters, we use the *Co-Clustering Accuracy (CCA)* proposed by [Govaert, 2008] to compare two pairs of partitions. CCA is defined from *Clus-*

Table 3.7: Characteristics of the synthetic datasets.

	Rows	**Cols**	**Biclusters**	**Sizes**	**Sparse**	**Structure**
A	500	500	10	equal	Yes	Block diagonal
B	800	1000	6	unequal	No	Block diagonal
C	800	800	7	equal	No	Checkerboard
D	2000	1200	4	unequal	No	Checkerboard

tering Accuracy (CA) associated to π^r and π^c in comparison with the true row and column clusters; it is given by

$$\mathrm{CCA}(\pi^r, \pi^c) = \mathrm{CA}(\pi^r) + \mathrm{CA}(\pi^c) - \mathrm{CA}(\pi^r) \times \mathrm{CA}(\pi^c).$$

Results. We report the biclustering performance on the synthetic datasets in table 3.8. At least one of our models finds the perfect partition in all cases. These tests additionally allow us to show the utility of the row cluster distribution $\mathbf{r}$ and column cluster distribution $\mathbf{c}$. The use of these ground truth distributions resulted in an increase of 19.5 and 4.2 points for BCOT on C and D, and an increase of 0.3 and decrease of 0.8 for BCOT$_\lambda$ on C and D.

Table 3.8: Biclustering performance on four synthetic datasets. gnd stands for ground truth.

Method	**A**	**B**	**C**	**D**
k-means	**100.0±0.0**	95.0±5.0	95.3±4.0	96.6±4.7
CCOT	54.4±3.5	70.0±0.0	29.7±0.4	55.7±1.8
CCOT-GW	99.1±0.0	83.5±0.0	83.4±0.0	75.3±0.0
COOT	99.8±0.0	78.8±2.0	99.8±0.0	93.7±1.2
COOT$_\lambda$	39.9±2.4	84.9±4.6	28.2±0.0	60.7±0.0
BCOT	99.8±0.0	80.4±2.2	99.6±0.1	91.3±0.7
BCOT$_\lambda$	**100.0±0.0**	99.1±0.4	**100.0±0.0**	**100.0±0.0**
BCOT (gnd $\mathbf{r}$, $\mathbf{c}$)	same $\mathbf{r}$, $\mathbf{c}$	99.9±0.0	same $\mathbf{r}$, $\mathbf{c}$	95.5±2.3
BCOT$_\lambda$ (gnd $\mathbf{r}$, $\mathbf{c}$)	same $\mathbf{r}$, $\mathbf{c}$	**100.0±0.0**	same $\mathbf{r}$, $\mathbf{c}$	99.2±0.9

5.7. Statistical Significance

We performed a Nemenyi post-hoc test [Nemenyi, 1963; Demšar, 2006] with a confidence level of 90% on the document and term clustering results that we obtained, to determine whether our model outperforms other OT biclustering approaches in a statistically significant way. To conduct this test we generated 20 performance rankings of the OT biclustering models based on their performance for each dataset and quality metric pair for both document and term clustering. Figure 3.5 shows the results of the test. We see that two differently performing groups were identified, one comprising BCOT and BCOT$_\lambda$ and giving better results than the other group comprising the remaining COOT and CCOT variants. This indicates that with this specific number of datasets and metrics the test was unable to tell

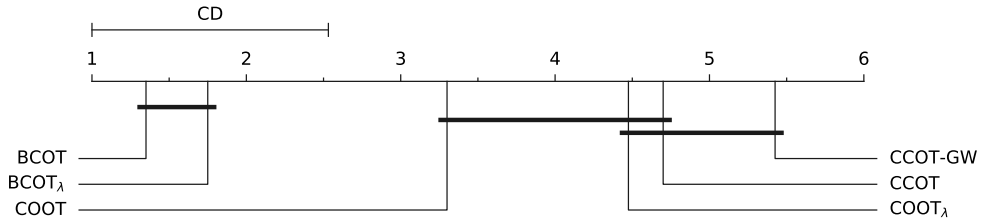


Figure 3.5: Result of the Nemenyi post hoc test.

COOT and CCOT apart in a statistically significant way.

6. Conclusion

Clustering and biclustering through optimal transport is still at a nascent stage, with many challenges remaining unsolved. This work introduces a novel problem for biclustering using optimal transport that takes into account the sparse nature of certain types of dyadic data such as document-term matrices, to enable more computationally efficient resolution. The problem is posed as a bilinear program that we solve using an efficient block coordinate descent algorithm to find a vertex solution. Experiments on a number of document-term datasets suggest that the proposed approach does a good job in finding clusters that correspond to ground truth document classes, while generating semantically coherent partitions for the terms. In this setting, our model outperforms recent OT biclustering methods by a significant margin, while being more computationally efficient.

In the next chapter, we will be proposing a generalization of this approach to generic graphs, thereby, showing its interest for other types of data.

Chapter 4

Graph Clustering via Optimal Transport

Objective

In this chapter we aimed to introduce a graph clustering method based on the min-cut problem with parameterizable cluster size distribution for any notion of size. Interesting applications include balanced and long-tailed dataset clustering.

Contents

1. Introduction

Clustering is an important task in machine learning and computer vision. Intuitively, the task of image clustering boils down to grouping images into clusters such that the images within the same clusters are similar to each other, while those in different clusters are dissimilar. Applications are diverse and wide ranging, including, for example, content-based image retrieval [Bhunia, 2020; Lee, 2022; Bhunia, 2021], image annotation [Cheng, 2018; Cai, 2013], and image indexing [Cao, 2013]. Consequently, much research has been dedicated to image clustering [Chang, 2017; Ji, 2017; Elhamifar, 2013; Ji, 2019].

A popular way of formulating the image clustering problem is through the minimum graph cut (min-cut) problem where the graph is created based on the input images. However, in practice, the min-cut problem suffers from the formation of some small groups which leads to bad performance. As a result, other versions of min-cut were proposed that take into account the size of the resulting groups, in order to make the partition more balanced. This notion of size is variable, for example, in the Normalized Cut (`ncut`) problem [Shi, 2000], size refers to the total volume of a group, while in the Ratio Cut (`rcut`) problem [Hagen, 1992], it refers to the cardinality of a group. A common method for solving the `ncut` and `rcut` problems is that of the spectral clustering algorithm [Von Luxburg, 2007; Ng, 2001] which is popular due to it often showing good empirical performance and being somewhat efficient. The spectral clustering algorithm variants are present in many image clustering frameworks, such as for subspace clustering [Agrawal, 1998] where a spectral clustering algorithm is applied to a learned subspace affinity matrix to obtain a partition of the points according to the subspaces in which they lie.

However, some restrictions that apply to the spectral clustering algorithms and to most approaches tackling the `ncut` and `rcut` problems in general do exist. A first one is that the balance constraint is not strict enough, meaning that even if we include the size regularization into the min-cut problem, the groups are still not necessarily of similar size, which is why several truly balanced clustering algorithm have been proposed in the literature [Chen, 2017; Chen, 2019a; Li, 2018b]. Another problem is that the balance constraint is too restrictive for many real world datasets, for example, a recent trend in computer vision is to propose approaches dealing with long-tailed datasets which are datasets that contain head classes that represent most of the overall dataset and then have tail classes that represent a small fraction of the overall dataset [Xu, 2022; Zhu, 2014]. Some approaches propose integrating generic size constraints to the objective like in [Genevay, 2019; Höppner, 2008; Zhu, 2010], however these approaches directly deal with the input images (or data in general) instead of graphs.

In this work, we propose a novel framework that introduces generic and at the same time stricter size constraints to the min-cut problem using Optimal Transport. To sum up, the main contributions of this work are:

- We formulate a problem for obtaining graph cuts that are balanced for an arbitrarily defined notion of size instead of specifically the volume or cardinality as is traditionally

done in spectral clustering. We also propose a more general formulation of graph cuts with cluster size constraints which can help when dealing with perfectly balanced datasets and heavily imbalanced datasets such as long-tailed datasets which follow an exponential decay in sample sizes across different classes.

- We then propose a solution for said problem through optimal transport using an approach reminiscent of the simplex algorithm and analyze its computational complexity. Links with existing works are also studied.
- Comprehensive experiments on balanced and long-tailed data sets using two variants we named `OT-ncut` and `OT-rcut` showcase the effectiveness of the proposed method compared to the most common min-cut algorithms (that use spectral clustering) both in terms of obtaining the desired cluster sizes as well as clustering performance. We release the code of our algorithm [1] for reproducibility.

The rest of this work is organized as follows: Preliminaries are presented in Section 2. Some related work is discussed in section 3. The `OT-cut` problem and algorithm along with their analysis and links to prior research are given in section 4. We present experimental results and analysis in section 5. Conclusions are then given in section 6.

2. Related Work

Our work is related with balanced clustering, as the latter is a special case of it, as well as with the more generic problem of constrained clustering.

Balanced Clustering. A common class of constrained clustering problems is balanced clustering where we wish to obtain a partition with clusters of the same size. For example, [DeSieno, 1988] introduced a conscience mechanism which penalizes clusters relative to their size, [Ahalt, 1990], then employed it to develop the Frequency Sensitive Competitive Learning (FSCL) algorithm. In [Li, 2018b], authors proposed to leverage the exclusive lasso on the k-means and min-cut problems to regulate the balance degree of the clustering results. [Lin, 2019] proposed a simplex algorithm to solve a minimum cost flow problem similar to k-means. In [Chen, 2017], authors proposed a self-balanced min-cut algorithm for image clustering implicitly using exclusive lasso as a balance regularizer in order to produce balanced partitions.

Constrained Clustering. Some clustering approaches with generic size constraints, which can be seen as an extension of balanced clustering, also exist. In [Zhu, 2010], a heuristic algorithm to transform size constrained clustering problems into integer linear programming problems was proposed. Authors in [Ganganath, 2014] introduced a modified k-means algorithm which can be used to obtain clusters of preferred sizes. Clustering paradigms based

[1] https://github.com/chakib401/ot-cut

on OT generally offer the possibility to set a target distribution for resulting partitions. [Genevay, 2019] proposed a deep clustering algorithm through optimal transport with entropic regularization. In [Fettal, 2022c], authors proposed a way to perform biclustering which is an extension of clustering to bipartite graphs through Optimal Transport while choosing the size of the resulting biclusters.

3. Preliminaries

In what follows, $\Delta^n = \{\mathbf{p} \in \mathbb{R}_+^n | \sum_{i=1}^n p_i = 1\}$ denotes the n-dimensional standard simplex. $\Pi(\mathbf{w}, \mathbf{v}) = \{\mathbf{Z} \in \mathbb{R}_+^{n \times k} | \mathbf{Z}\mathbb{1} = \mathbf{w}, \mathbf{Z}^\top \mathbb{1} = \mathbf{v}\}$ denotes the transportation polytope, where $\mathbf{w} \in \Delta^n$ and $\mathbf{v} \in \Delta^k$ are the marginals of the joint distribution $\mathbf{Z}$ and $\mathbb{1}$ is a vector of ones, its size can be inferred from the context. Matrices are denoted with uppercase boldface letters, and vectors with lowercase boldface letters. For a matrix $\mathbf{M}$, its i-th row is $\mathbf{m}_i$. Tr refers to the trace of a square matrix. $\|.\|_0$ is the zero norm that returns the number of nonzero elements in its argument. $\otimes$ denotes tensor-matrix product.

3.1. Graph Cuts

Graph Cut. Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with an weighted adjacency matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ with $n = |\mathcal{V}|$, a cut is a partition of its vertices $\mathcal{V}$ into two disjoint subsets $\mathcal{A}$ and $\bar{\mathcal{A}}$. The value of a cut is given by

$$\text{cut}(\mathcal{A}) = \sum_{v_i \in \mathcal{A}, v_j \in \bar{\mathcal{A}}} w_{ij}. \tag{4.1}$$

Minimum k-cut Problem. The goal of the minimum k-cut problem is to find a partition $(\mathcal{A}_1, ..., \mathcal{A}_k)$ of the set of vertices $\mathcal{V}$ into k different groups that is minimal in some metric. Intuitively, we wish for the edges between different subsets to have small weights, and for the edges within a subset have large weights. Formally, it is defined as

$$\text{min-cut}(\mathbf{W}, k) = \min_{\mathcal{A}_1, ..., \mathcal{A}_k} \sum_{i=1}^{k} \text{cut}(\mathcal{A}_i). \tag{4.2}$$

This problem can also be stated as a trace minimization problem by representing the resulting partition $\mathcal{A}_1, \ldots, \mathcal{A}_k$ using an assignment matrix $\mathbf{X}$ such that for each row i, we have that

$$x_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ is in } \mathcal{A}_j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

This condition is equivalent to introducing two constraints which are $\mathbf{X} \in \{0,1\}^{n\times k}$ and $\mathbf{X}\mathbb{1} = \mathbb{1}$. The minimum k-cut problem can then be formulated as

$$\text{min-cut}(\mathbf{W}, k) = \min_{\substack{\mathbf{X}\in\{0,1\}^{n\times k}\\ \mathbf{X}\mathbb{1}=\mathbb{1}}} \quad \text{Tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}), \tag{4.4}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ refers to the graph Laplacian of the graph $\mathcal{G}$ and $\mathbf{D}$ is the diagonal matrix of degree of $\mathbf{W}$, i.e., $d_{ii} = \sum_j w_{ij}$.

Normalized Cut Problem. In practice, solutions to the minimum k-cut problem do not yield satisfactory partitions due to the formation of small groups of vertices. Consequently, versions of the problem that take into account some notion of "size" for these groups have been proposed. The most commonly used one is normalized cut [Shi, 2000]:

$$\texttt{ncut}(\mathbf{W}, k) = \min_{\mathcal{A}_1,\ldots,\mathcal{A}_k} \sum_{i=1}^{k} \frac{\text{cut}(\mathcal{A}_i)}{\text{vol}(\mathcal{A}_i)}, \tag{4.5}$$

since $\text{vol}(\mathcal{A}_i) = \mathbf{x}_i^T \mathbf{D}\mathbf{x}_i$, then this problem can also be stated as a trace minimization problem:

$$\texttt{ncut}(\mathbf{W}, k) = \min_{\substack{\mathbf{X}\mathbb{1}=\mathbb{1}\\ \mathbf{x}\in\{0,1\}^{n\times k}}} \quad \text{Tr}\left(\frac{\mathbf{X}^\top \mathbf{L}\mathbf{X}}{\mathbf{X}^\top \mathbf{D}\mathbf{X}}\right), \tag{4.6}$$

where the ratio can be taken as either right or left multiplication of the numerator by the inverse of the denominator, this equivalence is due to the fact that we use the trace operator and that the denominator is a diagonal matrix. A special case of the normalized graph cut is recovered by setting $\mathbf{D} = \mathbf{I}$ in problem 4.6. This problem is referred to as the ratio cut problem due to the different groups being normalized by their cardinality instead of their volumes:

$$\texttt{rcut}(W, k) = \min_{\mathcal{A}_1,\ldots,\mathcal{A}_k} \sum_{i=1}^{k} \frac{\text{cut}(\mathcal{A}_i)}{|\mathcal{A}_i|}, \tag{4.7}$$

and similarly to the normalized cut, since $|\mathcal{A}_i| = \mathbf{x}_i^T \mathbf{x}_i$, we can formulate the ratio cut problem as a trace minimization problem:

$$\texttt{rcut}(\mathbf{W}, k) = \min_{\substack{\mathbf{X}\in\{0,1\}^{n\times k}\\ \mathbf{X}\mathbb{1}=\mathbb{1}}} \quad \text{Tr}\left(\frac{\mathbf{X}^\top \mathbf{L}\mathbf{X}}{\mathbf{X}^\top \mathbf{X}}\right). \tag{4.8}$$

Spectral Clustering for the Normalized & Ratio Cuts. A common approach to solving the normalized graph cut problems, spectral clustering, replaces the partition constraints on $\mathbf{X}$ with a form of semi-orthogonality constraints. In the case of `rcut`, we have

$$\texttt{ncut}(\mathbf{W}, k) = \min_{\substack{\mathbf{X}\in\mathbb{R}^{n\times k}\\ \mathbf{X}^\top\mathbf{X}=\mathbf{I}}} \quad \text{Tr}\left(\mathbf{X}^\top \mathbf{L}\mathbf{X}\right). \tag{4.9}$$

On the other hand for `ncut`, the partition matrix $\mathbf{X}$ is substituted with $\mathbf{H} = \mathbf{D}^{1/2}\mathbf{X}$ and a semi-orthogonality constraint is placed on this $\mathbf{H}$, i.e.,

$$\texttt{ncut}(\mathbf{W}, k) = \min_{\substack{\mathbf{H}\in\mathbb{R}^{n\times k} \\ \mathbf{H}^\top\mathbf{H}=\mathbf{I}}} \operatorname{Tr}\left(\mathbf{H}^\top\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{H}\right). \tag{4.10}$$

A solution $\mathbf{H}$ for the `ncut` problem is formed by stacking the first k-eigenvectors of the symmetrically normalized Laplacian $\mathbf{L}_s = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ as its columns, and then applying a clustering algorithm such as k-means on its rows and assign the original data points accordingly [Ng, 2001]. The principle is the same for the spectral `rcut` algorithm.

3.2. Optimal Transport

Discrete optimal transport. The goal of the optimal transport problem is to find a minimal cost transport plan $\mathbf{X}$ between a source probability distribution of $\mathbf{w}$ and a target probability distribution $\mathbf{v}$. Here we are interested in the discrete Kantorovich formulation of OT. When dealing with discrete probability distributions, said formulation is

$$\texttt{OT}(\mathbf{M}, \mathbf{w}, \mathbf{v}) \triangleq \min_{\mathbf{X}\in\Pi(\mathbf{w},\mathbf{v})} \langle \mathbf{M}, \mathbf{X}\rangle, \tag{4.11}$$

where $\mathbf{M} \in \mathbb{R}^{n\times k}$ is the cost matrix, and c_{ij} quantifies the effort needed to transport a probability mass from $\mathbf{w}_i$ to $\mathbf{v}_j$.

Discrete Gromov-Wasserstein Discrepancy. The generic discrete Gromov-Wasserstein (GW) discrepancy [Peyré, 2016] is an extension of optimal transport to the case where the source and target distributions are defined on different metric spaces:

$$\mathrm{GW}(\mathbf{M}, \mathbf{M}', \mathbf{w}, \mathbf{v}) \triangleq \min_{\mathbf{X}\in\Pi(\mathbf{w},\mathbf{v})} \langle L(\mathbf{M}, \mathbf{M}') \otimes \mathbf{X}, \mathbf{X}\rangle \tag{4.12}$$

where $\mathbf{M} \in \mathbb{R}^{n\times n}$ and $\mathbf{M}' \in \mathbb{R}^{k\times k}$ are similarity matrices defined on the source space and target space respectively, and $L : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ is a divergence measure between scalars, $L(\mathbf{M}, \mathbf{M}')$ is the $n \times n \times k \times k$ tensor of all pairwise divergences between the elements of $\mathbf{M}$ and $\mathbf{M}'$.

Gromov-Wasserstein Learning for Graphs. The Gromov-Wasserstein partitioning paradigm [Xu, 2019] supposes that the Gromov-Wasserstein discrepancy can uncover the clustering structure of the observed source graph $\mathcal{G}$ when the target graph $\mathcal{G}_{dc}$ only contains weighted self-connected isolated nodes, this means that its adjacency matrix is diagonal. The weights of this diagonal matrix as well as the source and target distribution are a special function of the node degrees. Their approach uses a regularized proximal gradient method as well as a recursive partitioning scheme and can be used in a multi-view clustering setting. The problem with this approach is that it is extremely sensitive the hyperparameter setting which

is problematic since it is an unsupervised method. Another approach is the one introduced in [Chowdhury, 2021] which generalizes spectral clustering using Gromov-Wasserstein discrepancy and heat kernels but suffers from very high computational complexity since, given a graph with n node, during its optimization procedure involves the computation a gradient which is in $O(n^3 \log(n))$ and therefore is not usable for large scale graphs.

4. Graph Cuts with Arbitrary Size Constraints via OT

In this section, we derive our optimal transport-based constrained graph cut problem and propose a simple iterative algorithm for its resolution.

4.1. Graph Cuts via Optimal Transport.

As already mentioned, the good performance of the normalized cut algorithm comes from the normalization by the volume of each group in the cut. However, the size constraint is not a hard one, meaning that obtained groups are not of exactly the same volume. This leads us to propose to replace the volume normalization by a strict balancing constraint as follows:

$$\begin{aligned} &\min_{\mathcal{A}_1,\ldots,\mathcal{A}_k} \sum_{i=1}^{k} \operatorname{cut}(\mathcal{A}_i) \\ &\text{such that} \quad \operatorname{vol}(\mathcal{A}_1) = \ldots = \operatorname{vol}(\mathcal{A}_k). \end{aligned} \tag{4.13}$$

Similarly to the `ncut` problem, this problem can be formulated as a trace minimization problem:

$$\begin{aligned} &\min_{\mathbf{X}} \quad \operatorname{Tr}(\mathbf{X}^\top \mathbf{L} \mathbf{X}) \\ &\text{such that} \quad \mathbf{X}\mathbb{1} = \mathbf{D}\mathbb{1}, \quad \mathbf{X}^\top \mathbb{1} = \frac{\sum_i d_{ii}}{k}\mathbb{1}, \quad \forall_i \|\mathbf{x}_i\|_0 = 1 \end{aligned} \tag{4.14}$$

This problem is hard and may not contain feasible solutions. However, this problem can be slightly modified to become an instance of the Gromov-Wasserstein problem, to which relatively efficient heuristics exist. Specifically, the volume constraint can be implcitly satisfied by defining $\mathbf{X}$ to be an element of the transportation polytope with a uniform target distribution instead of being a partition matrix. The degrees are also normalized by dividing them by their total sum and then representing them as proportions instead of absolute quantities, yielding the following problem:

$$\begin{aligned} &\min_{\mathbf{X}} \quad \operatorname{Tr}(\mathbf{X}^\top \mathbf{L} \mathbf{X}) \\ &\text{such that} \quad \mathbf{X} \in \Pi\left(\frac{1}{\sum_i d_{ii}}\mathbf{D}\mathbb{1}, \frac{1}{k}\mathbb{1}\right) \end{aligned} \tag{4.15}$$

This formulation is a special case of the Gromov-Wasserstein problem for a source space whose similarity matrix in the initial space is $\mathbf{M} = \mathbf{L}$ and whose similarity matrix in the destination space is $\mathbf{M}' = \mathbf{I}$. Note that a ratio cut version can easily be obtained by replacing

the volume constraint with

$$|\mathcal{A}_1| = \ldots = |\mathcal{A}_k| \tag{4.16}$$

in problem 4.14, and similarly in problem 4.15, by setting $\mathbf{D} = \mathbf{I}$, giving rise to:

$$\min_{\mathbf{X} \in \Pi(\frac{1}{n}\mathbb{1}, \frac{1}{k}\mathbb{1})} \operatorname{Tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) \tag{4.17}$$

4.2. Graph Cuts with Size Constraints.

From the previous problem, it is easy to see that target distribution does not need to be uniform, and as such, any distribution can be considered, leading to further applications like long-tailed dataset clustering. Another observation is that any notion of size can be considered and not only the volume or cardinality of the formed node groups. We formulate an initial version of the generic optimal transport graph cut problem as:

$$\min_{\mathbf{X} \in \Pi(\boldsymbol{\pi}^s, \boldsymbol{\pi}^t)} \operatorname{Tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) \equiv \min_{\mathbf{X} \in \Pi(\boldsymbol{\pi}^s, \boldsymbol{\pi}^t)} \langle \mathbf{L}\mathbf{X}, \mathbf{X} \rangle , \tag{4.18}$$

where $\boldsymbol{\pi}^s_i$ is the relative 'size' of the element i and $\boldsymbol{\pi}^t_j$ is the desired relative 'size' of the group j. Through the form that uses the Frobenius product, it is easy to see how our problem is related to the Gromov-Wasserstein problem. These size parameters can either be set using domain knowledge by the expert using our algorithm or by trying multiple guesses and selecting the best one via internal clustering quality metrics such as Davies-Bouldin index [Davies, 1979], Silhouette score [Rousseeuw, 1987], etc.

4.3. Transport Plans as Partition Matrices

The proposed approach relies on the fact that the transport plan $\mathbf{X}$ can be interpreted as a partition matrix. Fortunately, this interpretation can be made through the concept of h-almost hard clustering [Fettal, 2022c]:

Definition 3 (h-almost hard clustering)**.** *An h-almost hard clustering is a clustering whose partition matrix is $\otimes \in \mathbb{R}^{n \times k}$ such that $\| \otimes \|_0 = n + h$ and for each row $\boldsymbol{c}$ of $\otimes$ we have that $\|\boldsymbol{c}\|_0 > 0$. When $h = 0$, we obtain a standard hard clustering with one non-zero element per row.*

The extreme points of the transportation polytope are always h-almost hard clustering (see [Peyré, 2019; Fettal, 2022c] for a proof), so we add an extreme point condition to our problem in order to always obtain a transport plan $\mathbf{X}$ that can be interpreted easily as a hard partition matrix:

$$\texttt{OT-cut}(\mathbf{L}, \boldsymbol{\pi}^s, \boldsymbol{\pi}^t) \triangleq \min_{\mathbf{X} \in \texttt{ext}(\Pi(\boldsymbol{\pi}^s, \boldsymbol{\pi}^t))} \operatorname{Tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) \tag{4.19}$$

where *ext* is the set of extreme points of its argument. Consequently we have the following proposition:

Proposition 4. *A solution* $\mathbf{X}$ *to the* `OT-cut` *problem is an* h*-almost hard clustering with* $h \in \{0, \dots, k-1\}$.

We can obtain a size constrained variant of the ncut problem by setting $\boldsymbol{\pi}^s = \frac{1}{\sum_i d_{ii}} \mathbf{D}\mathbb{1}$, giving rise to the following problem:

$$\begin{aligned} \texttt{OT-ncut}(\mathbf{L}, \boldsymbol{\pi}^t) \triangleq \min_{X} \quad & \operatorname{Tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) \\ \text{such that} \quad & \mathbf{X} \in \texttt{ext}\left(\Pi\left(\frac{1}{\sum_i d_{ii}}\mathbf{D}\mathbb{1}, \boldsymbol{\pi}^t\right)\right) \end{aligned} \tag{4.20}$$

Analogously, the variant of the rcut problem is obtained by setting $\boldsymbol{\pi}^s = \frac{1}{n}\mathbb{1}$, yielding:

$$\texttt{OT-rcut}(\mathbf{L}, \boldsymbol{\pi}^t) \triangleq \min_{\mathbf{X} \in \texttt{ext}\left(\Pi\left(\frac{1}{n}\mathbb{1}, \boldsymbol{\pi}^t\right)\right)} \operatorname{Tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}), \tag{4.21}$$

4.4. Optimization and Complexity.

Problem 4.19 is nonconvex due to the extreme point constraint. We propose to use proximal gradient descent with constant step size to search for a stationary point. We add an l2-norm regularizer to simplify the update rule obtained with the proximal gradient method. The resulting update rule is:

$$\mathbf{X}^{(t+1)} \leftarrow \arg \texttt{OT}\left((\mathbf{L} - \mathbf{I})\mathbf{X}^{(t)}, \boldsymbol{\pi}^s, \boldsymbol{\pi}^t\right). \tag{4.22}$$

Note that when $\mathbf{L}$ is symmetrically normalized, we have that $\mathbf{L}_{sym} - \mathbf{I} = -\mathbf{W}_{sym}$ and the update rule becomes:

$$\mathbf{X}^{(t+1)} \leftarrow \arg \texttt{OT}\left(-\mathbf{W}\mathbf{X}^{(t)}, \boldsymbol{\pi}^s, \boldsymbol{\pi}^t\right). \tag{4.23}$$

The resolution of this problem is possible by stating it as the earth-mover's distance (EMD) linear program [Hitchcock, 1941] which can be solved via the network simplex algorithm. This algorithm has been empirically observed to converge to some stationary point in few iteration based on the initial guess $\mathbf{X}^{(0)}$. To illustrate this, in figure 4.1, we can report the evolution of loss function OT-rcut on MNIST, OT-rcut on FMNIST and OT-ncut on CIFAR-10-LT ($\rho = 10$). The pseudocode for the optimization procedure is presented in algorithm 2.

Similarly to the algorithm proposed in [Peyré, 2016] for solving the GW problem with an arbitrary loss and cost matrices, there are no convergence guarantees. Possible heuristics to improve the quality of the final solution would be doing multiple runs with different initializations, or initializing the algorithm with a partition matrix obtained from a spectral-cut algorithm projected onto the transportation polytope.

Proof. We formulate the l_2-norm regularized `OT-cut` problem as

$$\min \ \operatorname{Tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) + I_{\texttt{ext}(\Pi(\boldsymbol{\pi}^s, \boldsymbol{\pi}^t))}(\mathbf{X}) - \|\mathbf{X}\|^2$$

where $I_{\mathcal{C}}$ is the indicator function of set $\mathcal{C}$. The proximal gradient update rule with respect

to this problem is:

$$\begin{aligned}
\mathbf{X}^{(t+1)} &\leftarrow \mathtt{prox}_{\alpha(I_{\mathtt{ext}(\Pi(\boldsymbol{\pi}^s,\boldsymbol{\pi}^t))}-\|.\|^2)}\left(\mathbf{X}^{(t)} - \alpha\nabla \operatorname{Tr}\left(\mathbf{X}^{(t)}\mathbf{L}\mathbf{X}^{(t)}\right)\right) \\
&\leftarrow \mathtt{prox}_{\alpha(I_{\mathtt{ext}(\Pi(\boldsymbol{\pi}^s,\boldsymbol{\pi}^t))}-\|.\|^2)}\left((\mathbf{I}-2\alpha\mathbf{L})\mathbf{X}^{(t)}\right) \\
&\leftarrow \arg\min_{\mathbf{Z}\in\mathtt{ext}(\Pi(\boldsymbol{\pi}^s,\boldsymbol{\pi}^t))} \frac{1}{2\alpha}\left\|\mathbf{Z}-(\mathbf{I}-2\alpha\mathbf{L})\mathbf{X}^{(t)}\right\|^2 - \|\mathbf{Z}\|^2 \\
&\leftarrow \arg\min_{\mathbf{Z}\in\mathtt{ext}(\Pi(\boldsymbol{\pi}^s,\boldsymbol{\pi}^t))} \frac{1}{2\alpha}\|\mathbf{Z}\|^2 + \frac{1}{2\alpha}\left\|(\mathbf{I}-2\alpha\mathbf{L})\mathbf{X}^{(t)}\right\|^2 \\
&\qquad - \frac{1}{\alpha}\operatorname{Tr}\left(\mathbf{Z}^\top(\mathbf{I}-2\alpha\mathbf{L})\mathbf{X}^{(t)}\right) - \|\mathbf{Z}\|^2,
\end{aligned} \tag{4.24}$$

then by setting $\alpha = \frac{1}{2}$:

$$\begin{aligned}
\mathbf{X}^{(t+1)} &\leftarrow \arg\min_{\mathbf{Z}\in\mathtt{ext}(\Pi(\boldsymbol{\pi}^s,\boldsymbol{\pi}^t))} \operatorname{Tr}\left(\mathbf{Z}^\top(\mathbf{L}-\mathbf{I})\mathbf{X}^{(t)}\right), \\
&\leftarrow \arg\min_{\mathbf{Z}\in\mathtt{ext}(\Pi(\boldsymbol{\pi}^s,\boldsymbol{\pi}^t))} \left\langle \mathbf{Z}, (\mathbf{L}-\mathbf{I})\mathbf{X}^{(t)}\right\rangle,
\end{aligned} \tag{4.25}$$

here, we can drop the extreme point constraint since, even without it, the solution is guaranteed to be an extreme point of the transportation polytope. This results in the classical OT problem with cost matrix $(\mathbf{L}-\mathbf{I})\mathbf{X}^{(t)}$ and marginals π^s and π^t::

$$\mathbf{X}^{(t+1)} \leftarrow \arg\min_{\mathbf{Z}\in\Pi(\boldsymbol{\pi}^s,\boldsymbol{\pi}^t)} \left\langle \mathbf{Z}, (\mathbf{L}-\mathbf{I})\mathbf{X}^{(t)}\right\rangle. \tag{4.26}$$

□

Proposition 5. *For a graph with $|\mathcal{E}|$ edges and n nodes, the complexity of an iteration of the proposed algorithm is $\mathcal{O}\left(kn^2 \log n\right)$.*

Proof. We note that in practice $n >> k$ and that the complexity of the network simplex algorithm for some graph $\mathcal{G}_{EMD} = (\mathcal{V}_{EMD}, \mathcal{V}_{EMD})$ is in $\mathcal{O}(|\mathcal{V}_{EMD}||\mathcal{E}_{EMD}| \log |\mathcal{E}_{EMD}|)$ [Orlin, 1997]. In our case, this graph has $|\mathcal{V}_{EMD}| = n + k$ (since $n >> k$, we can drop the k) and $|\mathcal{E}_{EMD}| = nk$. The other operation that is performed during each iteration is the matrix multiplication $(\mathbf{L}-\mathbf{I})\mathbf{X}^{(t)}$ whose complexity is in $\mathcal{O}(k|\mathcal{E}|)$, in the worst case when matrix $\mathbf{L}$ is fully dense, we have that $|\mathcal{E}| = n^2$. Note that the complexity of the spectral clustering algorithm is in $\mathcal{O}(kn^2)$. □

5. Links to Prior Works

In this section we discuss how our approach generalizes and can used in conjunction with other approaches.

Algorithm 2: Proximal Gradient Descent for `OT-cut`

Input : $\mathbf{L}$ Laplacian matrix,
$\boldsymbol{\pi}^s$ node size distribution,
$\boldsymbol{\pi}^t$ cluster size distribution,
$\mathbf{G}_{init}$ initial partition matrix,
`max_iter` maximum number of iterations.

Output: $\mathbf{G}$ partition of the graph.

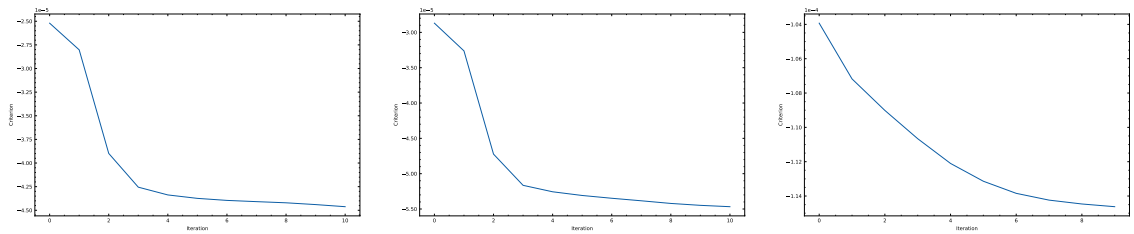
$\mathbf{X}^{(0)} \leftarrow \arg \texttt{OT}\left(\mathbf{G}_{init}, \boldsymbol{\pi}^s, \boldsymbol{\pi}^t\right)$;

while $t < max\_iter$ **do**

  $\mathbf{X}^{(t+1)} \leftarrow \arg \texttt{OT}\left((\mathbf{L} - \mathbf{I})\mathbf{X}^{(t)}, \boldsymbol{\pi}^s, \boldsymbol{\pi}^t\right)$;

end

Generate partition matrix $\mathbf{G}$ such that each node v_i is assigned it to partition $\arg\max_i x_i$;



(a) OT-rcut on MNIST

(b) OT-rcut on FMNIST

(c) OT-ncut on CIFAR-10 (ρ=10)

Figure 4.1: Evolution of loss as function of the number of iterations.

5.1. Optimal-Transport Based Biclustering

Biclustering is the extension of clustering to bipartite graphs. Here, we recover the BCOT [Fettal, 2022c] problem as a special case of `OT-cut`. Given a bipartite adjacency matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{n\times n} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{0}_{d\times d} \end{bmatrix},$$

we recover their formulation through ours by considering this anti-adjacency matrix $\bar{\mathbf{A}}$:

$$\bar{\mathbf{A}} = \begin{bmatrix} \infty_{n\times n} & L(\mathbf{B}) \\ L(\mathbf{B})^\top & \infty_{d\times d} \end{bmatrix}.$$

Then setting $\boldsymbol{\pi}^s = [\mathbf{v}, \mathbf{w}]^\top$ and $\boldsymbol{\pi}^s = [\mathbf{v}, \mathbf{w}]^\top$ and omitting the extreme point condition. All in all, we have that

$$\text{BCOT}(L(\mathbf{B}), \mathbf{w}, \mathbf{v}) \;\equiv\; \texttt{OT-cut}\left(\bar{\mathbf{A}}, [\mathbf{v}, \mathbf{w}]^\top, [\mathbf{w}, \mathbf{v}]^\top\right)$$

Dataset	#Images	#Classes	Balance
MNIST [Deng, 2012]	60,000	10	1.0
Fashion-MNIST [Xiao, 2017]	60,000	10	1.0
KMNIST [Clanuwat, 2018]	60,000	10	1.0
CIFAR-10 [Krizhevsky, 2009]	50,000	10	1.0
CIFAR-10-LT ($\rho = 5$)	25,423	10	5.0
CIFAR-10-LT ($\rho = 10$)	20,431	10	10.0
CIFAR-10-LT ($\rho = 20$)	17,023	10	20.0
CIFAR-10-LT ($\rho = 100$)	12,406	10	100.0

Table 4.1: Characteristics of the datasets from which we construct the graphs.
The balance score ρ is the ratio of the number of occurrences of the most frequent class over that of the least frequent class.

5.2. OT Kernel *k*-Means.

In [Genevay, 2019], authors proposed an algorithm for k-means with cluster size constraints and entropic regularization. By dropping the regularization and adding an extreme point constraint, one can think of the case where the adjacency matrix in our formulation is a kernel matrix and use the same principles that were used with kernel k-means [Dhillon, 2004] to optimize the OT graph cut criterion.

6. Experiments

We ran experiments on balanced and heavily imbalanced (long-tailed) datasets. We evaluated the clustering performance of three variants of each of `OT-ncut` and `OT-rcut` algorithms against the spectral `rcut` and `ncut` algorithms, as well as the ability of our approach to recover the desired partition distribution. We had initially also considered S-GWL [Xu, 2019] as a baseline but its empirical performance was very poor. Specifically, it consistently resulted in assigning all the nodes to a single cluster. We believe that this is due to the fact that their algorithm is very sensitive to the hyperparameters selected as well as the fact that they used entropic regularization which leads to coupling matrices being fully dense. Considering entropic regularization also leads to poor results for our approach. Another OT based approach which was not considered is SpecGWL [Chowdhury, 2021] due to its log-cubic complexity which makes it unusable for the datasets we considered.

6.1. Datasets

We perform experiments on balanced datasets and long-tailed datasets. The statistical summaries of these datasets are available in table 4.1. The CIFAR-10-LT ($\rho = \frac{\max_i n_i}{\min_i n_i}$) variants, are generated using a long-tailed imbalance sampling method that yields a dataset whose majority class is ρ times more frequent than the minority class following the procedure

Table 4.2: Image clustering performance on the imbalanced (long-tail) datasets. Values are the averages over five runs. Standard deviations were not reported due to being negligible (≤ 0.1). Best results are highlighted in bold font.

	CIFAR-10-LT ($\rho = 5$)			CIFAR-10-LT ($\rho = 10$)			CIFAR-10-LT ($\rho = 20$)			CIFAR-10-LT ($\rho = 100$)		
	NMI	ARI	CF1	NMI	ARI	CF1	NMI	ARI	CF1	NMI	ARI	CF1
SC-rcut	0.1	-0.0	3.3	0.1	-0.0	4.0	0.1	-0.0	4.6	0.1	-0.0	5.8
OT-rcut	11.6	7.3	20.7	**12.1**	**7.8**	**19.8**	**11.4**	7.5	**17.9**	9.8	5.7	13.7
OT-rcut$_{SC}$	11.1	6.4	20.8	10.6	6.5	18.7	11.3	7.4	17.1	9.8	5.8	13.7
OT-rcut$^*_{SC}$	11.2	6.1	19.5	10.5	5.4	16.6	10.8	5.4	14.6	11.6	5.6	14.3
SC-ncut	10.2	5.6	19.1	10.5	6.2	18.0	10.6	5.8	16.4	**12.7**	6.8	14.6
OT-ncut	**12.0**	**8.3**	**21.3**	10.1	7.3	18.9	10.6	**7.9**	17.3	8.4	6.9	13.8
OT-ncut$_{SC}$	10.8	7.5	20.7	10.8	7.5	18.6	10.5	7.8	16.2	10.4	**8.3**	**14.8**
OT-ncut$^*_{SC}$	10.4	5.9	20.4	10.4	5.6	18.0	10.6	5.7	16.4	10.9	5.6	13.1

described in [Cao, 2019].

6.2. Metrics

The evaluation is straightforward, we adopt four popular clustering metrics when dealing with the balanced datasets: clustering accuracy (CA), clustering F1 score (CF1), normalized mutual information (NMI), and adjusted rand index (ARI) [Hubert, 1985]; multiplied by 100. CA and CF1 are computed by solving a linear assignment problem [Crouse, 2016]. When dealing with the long-tailed dataset, we only use metrics that are sensitive to imbalance NMI, ARI, and CF1. When comparing the concordance of the input cluster distribution $\boldsymbol{\pi}$ and the cluster distribution obtained via one of our algorithms $\hat{\boldsymbol{\pi}}$, we use the Kullback-Leibler divergence [Kullback, 1951]. The concordance of two perfectly matching distributions will be equal to zero, otherwise it will be larger.

6.3. Experimental settings

We compare two variants of our algorithm, namely, the `OT-ncut` and `OT-rcut` implemented via the Python optimal transport package (POT) [Flamary, 2021] to the spectral clustering variants SC-ncut and SC-rcut which were based on the implementation of the spectral clustering in the Scikit-Learn package [Pedregosa, 2011]. For each image dataset represented in matrix form as $\mathbf{Y}$, we use subspace Least Squares Regression Subspace Clustering (LSR) [Lu, 2012] to create the graph, we get $\mathbf{A} = \mathbf{Y}\mathbf{Y}^\top \left(\mathbf{Y}\mathbf{Y}^\top + \mathbf{I}\right)^{-1}$. Note that all experiments are run five times. In the results tables, base `OT-cut`'s variants use random initialization. The variants that end with * use the ground truth target distribution. Finally, for variants ending in $_{SC}$, we choose the initial transport plan $\mathbf{X}^{(0)}$ by first obtaining a partition matrix through the corresponding spectral clustering algorithm, i.e., spectral ncut (SC-ncut) for OT-ncut and spectral rcut (SC-rcut) to OT-rcut. We perform 10 iterations of our algorithm to fine-tune the initial guesses of spectral cuts and perform 20 iterations when using random initialization.

Table 4.3: Clustering performance on balanced image datasets. Values are the averages over five runs. Standard deviations were not reported due to being negligible (≤ 0.1). Best results are highlighted in bold font. OT-rcut* has the same results since the ground truth sizes are uniform, similarly, OT-rcut$_{SC}$ also has the same results due to SC-rcut returning a bad guess that is equivalent to a random initialization.

	MNIST				FMNIST				KMNIST				CIFAR 10			
	ACC	NMI	ARI	CF1	ACC	NMI	ARI	CF1	ACC	NMI	ARI	CF1	ACC	NMI	ARI	CF1
SC-ncut	40.2	34.7	17.6	37.6	53.4	53.2	36.5	51.6	37.3	30.4	19.8	35.2	22.2	10.1	5.6	21.5
OT-ncut$_{SC}$	**48.2**	**36.4**	**27.1**	**48.2**	47.8	51.0	35.5	47.0	43.6	33.2	24.2	42.8	21.5	11.5	**6.4**	21.3
OT-ncut$^{*}_{SC}$	41.5	35.5	25.0	41.1	**56.3**	53.0	**40.7**	**56.0**	**44.7**	**33.6**	**24.2**	**44.5**	23.0	10.8	5.8	23.0
SC-rcut	11.2	0.0	-0.0	2.0	10.0	0.0	0.0	1.8	10.0	0.0	0.0	1.8	10.0	0.0	0.0	1.8
OT-rcut	38.3	32.3	20.7	38.2	54.3	**53.9**	39.0	54.3	41.6	33.1	22.3	41.6	**23.8**	**11.7**	**6.4**	**23.8**
OT-rcut$_{SC}$	Same results as OT-rcut															
OT-rcut$^{*}_{SC}$	Same results as OT-rcut															

All experiments were performed on a 64gb RAM machine with a 12th Gen Intel(R) Core(TM) i9-12950HX (24 CPUs) processor with a frequency of 2.3GHz.

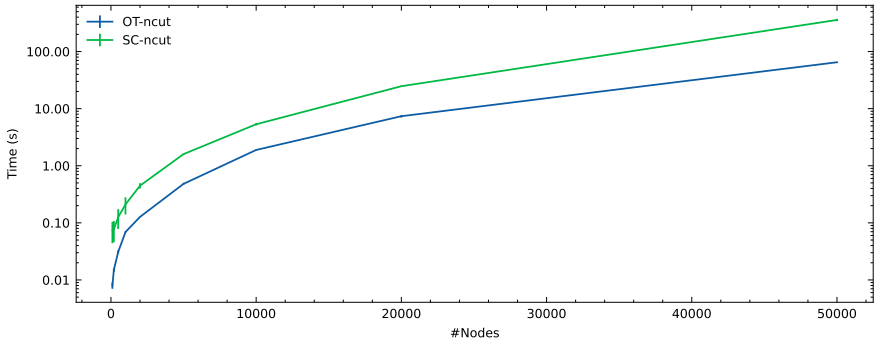


Figure 4.2: Training times of OT-ncut in seconds (log scale) over subsets of different sizes of MNIST.

6.4. Results

We emphasize the fact that the most important quality metric is the relative difference instead of the absolute value of the metrics since our objective is not to learn a better graph over the dataset but rather to get a better cut over the chosen graph.

Performance on balanced datasets. For balanced datasets, the results are reported in table 4.3. One of our two approaches yields the best results in all 16 cases. Furthermore, each one of them improves over the results of their spectral counterpart, exceeding them in 31 out of 32 cases. Notably, our `OT-rcut` variant achieves a significant improvement over the spectral ratio cut algorithm.

Performance on long-tailed datasets. Table 4.2 presents the results obtained on the long-tailed datasets. In all cases, `OT-ncut` and `OT-rcut` outperform their spectral clustering counterparts, yielding the best performance in 11 out of 12 cases. Notably, the improvement of

Table 4.4: The Kullback-Leibler divergence between the imposed target distribution and the one obtained using `OT-cut` variants.

	OT-ncut*	OT-ncut	OT-rcut*	OT-rcut
MNIST	3.00e-09	2.30e-09	0.0	0.0
FMNIST	1.70e-09	2.60e-09	0.0	0.0
KMNIST	2.50e-09	4.10e-09	0.0	0.0
CIFAR-10	3.70e-09	3.10e-09	0.0	0.0
CIFAR-10-LT ($\rho = 5$)	0.0	2.53e-07	1.62e-08	0.0
CIFAR-10-LT ($\rho = 10$)	1.48e-08	6.70e-09	1.08e-08	0.0
CIFAR-10-LT ($\rho = 20$)	7.00e-09	1.26e-08	7.07e-08	0.0
CIFAR-10-LT ($\rho = 100$)	8.62e-08	5.14e-08	7.80e-08	0.0

`OT-rcut` over SC-rcut is particularly significant, consistent with the findings in the balanced case.

Concordance of the Desired & Resulting Cluster Sizes. To evaluate our algorithm's ability to produce a partition with the desired group size distribution, we use the Kullback-Leibler (KL) divergence metric. Specifically, we compare the distribution obtained by our OT-rcut and OT-ncut variants against the target distribution specified as a hyperparameter ($\boldsymbol{\pi}^t$). Table 4.4 presents the KL divergences for both variants on various datasets. Our approaches achieve near-perfect performance on most datasets. Notably, OT-rcut is able to fully recover the desired group sizes.

Running Time. As shown in figure 4.2, OT-ncut with random initialization is more efficient than the spectral ncut algorithm, significantly outspeeding it on all subsets of MNIST despite being theoretically more complex. This is due to the fact that our algorithm needs few iterations to converge. Regularization can be introduced to further speed up our algorithms such as low-rank [Scetbon, 2022] and entropic [Cuturi, 2013] regularizations .

7. Conclusion

In this work we proposed a new graph cut algorithm for partitioning with arbitrary size constraints through optimal transport. This approach generalizes the concept of the normalized and ratio cut to arbitrary size distributions and this for any notion of size. The proposed algorithm works well when used in conjunction with a classical spectral graph cut algorithm as a post-processing step to obtain some desired distribution. Experiments on balanced and imbalanced datasets showed the effectiveness of our approach both in terms of clustering performance, computational speed, as well as its ability to recover partitions that almost perfectly match the desired ones.

In the next chapter, we will be working on attributed graphs i.e. graphs with attributes on their nodes. To deal with such types of graphs, approaches find a way to take into consid-

eration both graph structure and node information. We will be doing that via neighborhood propagation and using a new efficient low-rank subspace clustering approach to partition the resulting representations.

Chapter 5

Attributed Graph Joint Embedding and Clustering

Objective

This chapter was published as [Fettal, 2022c]. Our goal was to create an efficient yet effective approach for the simultaneous embedding and clustering of attributed-graph nodes based on the simplified graph convolutional network, as well as the reduced k-means.

Contents

1. Introduction

In data science, low-dimensional representation learning is commonly used for visualization purposes, but it can also play a significant role in the clustering task, where the aim is to divide a dataset into homogeneous clusters. Indeed, working with a low-dimensional space can be useful when partitioning data, and a number of approaches are reported in the literature. Recently, the authors in [Karim, 2020a] have performed experiments on the sequential combination of deep representation learning techniques such as the autoencoder (AE), variational AE [An, 2015] and convolutional AE [Ghasedi Dizaji, 2017; Karim, 2020b], and some popular clustering methods such as k-means; interactive Python Notebooks and further technical details can be found at [1]. They observed that this improves clustering results but that there is no 'one size fits all'. Thus, low-dimensional representation learning followed by cluster analysis can be helpful in data science. The `k-means` applied on data embeddings, derived from classical embedding methods such as the AE for instance, is a popular approach. This procedure is carried out sequentially and is referred to as *the tandem approach* [Yamamoto, 2014]. However, AE may sometimes be unsuitable for reducing dimension before clustering; it can fail to retain information which could be valuable for the clustering task. Hence, jointly optimizing for both tasks –representation learning and clustering– is a good alternative [Fard, 2020; Allab, 2016; Allab, 2018; Labiod, 2021]. Learning representations that are both faithful to the data while simultaneously adjusting them to have a clustering-friendly structure can lead to a better clustering performance [De Soete, 1994; Fard, 2020; Asano, 2020].

Clustering in the context of attributed graphs, which are graphs whose nodes and/or edges have attributes or features, despite being an important unsupervised task, has proved more impervious to such advances. Furthermore, some of these attributed graph clustering methods suffer from high spatial and/or computational complexity. Unlike most existing approaches, this work aims to overcome this weakness by considering a joint graph embedding and clustering, which alternates iteratively between both tasks, that is to say between embedding and clustering. Attributed graphs are used to model a wide variety of real-world networks such as recommender systems [Salah, 2017b; Fan, 2019; Ying, 2018], computer vision [Satorras, 2018; Qi, 2017; Yang, 2018a], Natural language processing [Marcheggiani, 2017; Song, 2018] and physical systems [Hoshen, 2017; Sanchez-Gonzalez, 2018]. Due to the irregular high-dimensional non-euclidean structure of graphs as well as the various node-level features it may contain, looking for suitable euclidean-representations that incorporate the structural and features' information of these graphs is an interesting challenge in machine learning [Hamilton, 2017]. Recent literature proposes to learn these representations automatically. Loosely speaking, these representation learning methods aim at embedding the nodes into a low-dimensional space where in the embedded nodes' proximity should be similar enough to that of those in the original graph representation. These methods can be based on approaches such as factorization [Belkin, 2003; Ahmed, 2013], random walks [Perozzi, 2014;

[1] https://github.com/rezacsedu/Deep-learning-for-clustering-in-bioinformatics

Grover, 2016], or neighborhood autoencoders [Wang, 2016; Zhu, 2020]. Recently, the *Graph Convolutional Network* (GCN) [Defferrard, 2016; Kipf, 2017] has garnered a lot of attention due to its ability to learn high-quality graph representations, and by extension, its effectiveness for different graph-related tasks such as node classification, link prediction, and node clustering which is the task our work focuses on. This node clustering, however, is generally performed as a downstream task but some efficient GCN-based approaches for the simultaneous embedding and clustering have recently emerged [Zhang, 2019; Anton Tsitsulin, 2020]. In this work, we propose to rely on the GCN to develop a novel *Graph Convolutional Clustering* model referred to as GCC that is capable of taking into account both tasks simultaneously. Our contributions in this work can be summarized as follows:

- We provide a variant of the GCN propagation matrix and demonstrate how it makes the GCN truly act as a low-pass filter.
- We propose a new formulation combining the graph convolutional representation learning and the clustering processes and show how our proposed GCC approach is related to some other methods.
- We derive an efficient algorithm referred to as `GCC`[2] and study its computational complexity in detail. We release the code[3] for easy reproducibility.
- We perform experimentations to showcase the worth of our proposal both in terms of clustering and quality of embedding.

This work is organized as follows. Section 2 presents related works. Section 3 presents the proposed approach and its derivation. Section 4 is devoted to the proposed algorithm and its computational complexity study. In section 5, we compare `GCC` with the state-of-the-art in terms of clustering and evaluate its performance in terms of embedding. Finally, Section 6 presents our conclusions.

2. Related Work

Our contributions lie in the intersection of several research topics, graph representation learning, graph clustering and graph convolutional neural networks.

Unsupervised Graph Representation learning. Unsupervised Graph Representation learning is generally done either through contrastive learning or via autoencoders.

The contrastive methods learn representations in a self-supervised way. They commonly rely on maximizing mutual information. DGI [Velickovic, 2019], for example, maximizes mutual information between node and graph representations. InfoGraph [Sun, 2020] expands the previous concept to graph substructures of different scales rather than just the node-level e.g. edges, triangles.

Autoencoder-based models learn embeddings by trying to reconstruct some property of the graph, generally the adjacency matrix. Variational Graph Autoencoders (VGAE) [Kipf,

[2] From now on, in order to distinguish between a model and its derived algorithm, we will use *typewriter font* for an algorithm. Consequently, GCC is the model and `GCC` its derived algorithm.

[3] https://github.com/chakib401/graph_convolutional_clustering

2016] extend the concept of variational autoencoders to the graph context, it uses a GCN based encoder and a dot product decoder. Linear variational Graph autoencoders [Salha, 2020] simplify VGAE by defining the encoder to be a linear transformation with a one-hop propagation matrix.

Graph Clustering. Graph clustering is the process of grouping nodes into clusters depending on the structure of the graph and/or node-level features. By only considering node attributes, classical clustering algorithms can be used to cluster the graph. Algorithms that rely on graph structure exclusively include the spectral clustering algorithm [Ng, 2001] that optimizes the ratio and normalized-cut criteria. Graclus [Dhillon, 2007] is mathematically equivalent to the spectral clustering algorithm but is faster due to not having to compute the eigenvalues of the graph Laplacian.

Finally, approaches that leverage both graph structure and node attributes commonly learn representations before applying classical clustering algorithms on them [Wu, 2019; Zhu, 2021]. However, some recent works explored integrating the clustering loss directly into the objective.

Clustering-friendly Graph Representation Learning. Literature on joint representation learning and clustering claims that doing the two tasks simultaneously can improve clustering quality. DCN [Yang, 2017] proposed to include the k-means clustering loss to the autoencoder loss as a regularization. Deep k-means [Fard, 2020] followed on this concept by proposing a fully differentiable formulation of this problem. In the context of attributed graph clustering, joint clustering and embedding is starting to receive some attention. GEMSEC [Rozemberczki, 2019] maximizes the information between labels and visual input data indices in order to self-label the data. AGC [Zhang, 2019] proposes to exploit high-order neighborhoods in the clustering process through an adaptive rule for neighborhood order selection. Deep Modularity Network [Anton Tsitsulin, 2020] clusters the graph by maximizing spectral modularity. Graph InfoClust (GIC) [Mavromatis, 2021] computes clusters by maximizing the mutual information between nodes contained in the same cluster.

3. Proposed Method

In this section we describe how we formulate the simultaneous node embedding and clustering problem. Then we propose a new model for solving it (as depicted in figure 5.1).

3.1. Preliminaries and Notations

Let $\mathcal{G} = (V, \mathbf{A}, \mathbf{X})$ be an attributed undirected graph where V represents the vertex set consisting of nodes $\{v_1, ..., v_n\}$, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a symmetric adjacency matrix where a_{ij} denotes the edge weight between nodes v_i and v_j, and $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a node-level feature matrix. Tr denotes the trace of a matrix. In what follows k represents the number of clusters. f is the

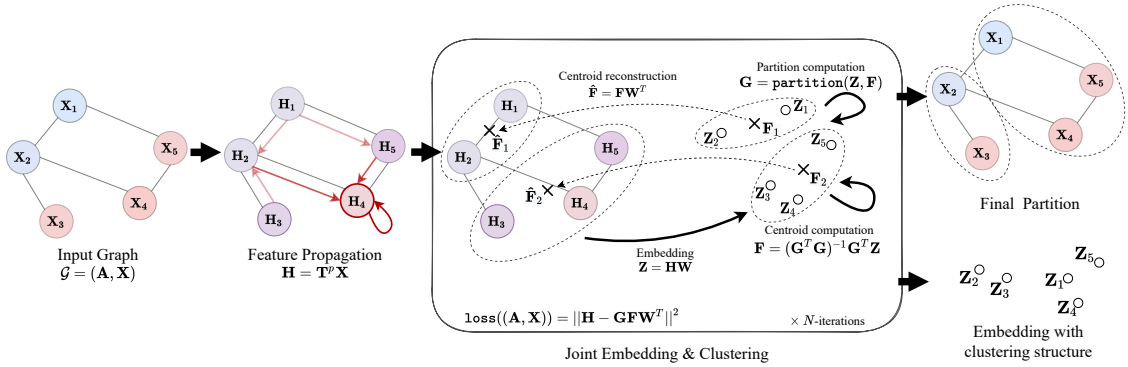


Figure 5.1: Schema of the GCC model: GCC creates an initial representation of the graph before iteratively learning to embed and cluster the data. The graph signal is represented by the colors of the node. Feature propagation results in a smoother signal.

embedding dimension. $\mathbf{1}_m$ represents a column vector of m ones. $\mathbf{I}_m$ represents an identity matrix of dimension m. If $\mathbf{G}$ is a matrix then $\mathbf{m}_i$ is its i-th row vector, $\mathbf{m}'_j$ is its j-th column vector and m_{ij} is the j-th element of the i-th row.

3.2. Joint Graph Representation Learning and Clustering

We formulate the simultaneous node embedding and clustering problem as follows

$$\begin{aligned} \min_{\theta_1,\theta_2,\mathbf{G},\mathbf{F}} \quad & \underbrace{\left|\left| \texttt{dec}_{\theta_2}\Big(\texttt{enc}_{\theta_1}\big(\texttt{agg}(\mathbf{A},\mathbf{X})\big)\Big) - \texttt{agg}(\mathbf{A},\mathbf{X}) \right|\right|^2}_{\text{reconstruction term}} \\ & + \alpha \underbrace{\left|\left| \texttt{enc}_{\theta_1}\big(\texttt{agg}(\mathbf{A},\mathbf{X})\big) - \mathbf{G}\mathbf{F} \right|\right|^2}_{\text{clustering regularization term}} \\ \text{s.t.} \quad & \mathbf{G} \in \{0,1\}^{n\times k},\ \mathbf{G}\mathbf{1}_k = \mathbf{1}_n \end{aligned} \tag{5.1}$$

where $\texttt{enc}_{\theta_1}$ is the encoding function, $\texttt{dec}_{\theta_2}$ is the decoding function, $\texttt{agg}(\mathbf{A},\mathbf{X})$ is some aggregate of $\mathbf{A}$ and $\mathbf{X}$ which represents the information contained in the graph (structure and node-features), $\mathbf{G} \in \{0,1\}^{n\times k}$ the binary classification matrix, $\mathbf{F} \in \mathbb{R}^{k\times d}$ play the role of centroids in the embedding space and α is a coefficient that regulates the trade-off between seeking reconstruction and clustering.

The clustering regularizer is the k-means clustering loss [Lloyd, 1982] on the encoded observations. It penalizes transformations that do not result in a clustering-friendly representation.

3.3. Linear Graph Embedding

Linear graph autoencoders (LGAE) [Salha, 2020] have shown that a linear encoder with an inner product decoder can be powerful enough to reach competitive results w.r.t more

complex GCN-based models on the link prediction and node clustering tasks. Consequently, we also define our encoder to be a simple linear transformations i.e.

$$\texttt{enc}(\texttt{agg}(\mathbf{A}, \mathbf{X}); \mathbf{W}_1) = \texttt{agg}(\mathbf{A}, \mathbf{X})\mathbf{W}_1$$

In LGAE, the decoder attempts to reconstruct the adjacency matrix $\mathbf{A}$ rather than an aggregation of $\mathbf{A}$ and $\mathbf{X}$. This means that this type of decoder is not suitable for our problem. Therefore, we also define the decoder as a simple linear transformation

$$\texttt{dec}(\mathbf{Z}; \mathbf{W}_2) = \mathbf{Z}\mathbf{W}_2$$

where $Z = \texttt{agg}(\mathbf{A}, \mathbf{X})\mathbf{W}_1$.

3.4. Normalized Simple Graph Convolution

Our choice for the aggregate function is inspired by the simple graph convolution proposed in SGC [Wu, 2019]. We set

$$\texttt{agg}(\mathbf{A}, \mathbf{X}) = \mathbf{T}^p\mathbf{X} \tag{5.2}$$

but rather than have $\mathbf{T}$ be the symmetric normalized adjacency matrix with added self-loops, we define it to be

$$\mathbf{T} = {\mathbf{D}_\mathbf{T}}^{-1}(\mathbf{I} + \tilde{\mathbf{S}}) \tag{5.3}$$

where $\tilde{\mathbf{S}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ with $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ (resp. $\mathbf{D}_\mathbf{T}$) being the diagonal matrix of degrees of $\tilde{\mathbf{A}}$ (resp. $\mathbf{I} + \tilde{\mathbf{S}}$).

The GCN, and by extension the SGC, do graph signal filtering with matrix $\mathbf{I} - \tilde{\mathbf{S}} = \mathbf{I} - \tilde{\mathbf{D}}^{-1/2}(\mathbf{I} - \tilde{\mathbf{L}})\tilde{\mathbf{D}}^{-1/2}$ where $\tilde{\mathbf{L}}$ is the Laplacian of $\tilde{\mathbf{A}}$. The frequency response function of this filter is $h(\tilde{\lambda_l}) = 1 - \tilde{\lambda_l}$ where λ_l is a frequency of the graph. In the GCN stacking K-layers, or equivalently raising $\tilde{\mathbf{S}}$ to power K in SGC, implies doing the filtering with frequency response function $h_K(\tilde{\lambda_l}) = (1 - \tilde{\lambda_l})^K$. This filter is low-pass on $[0, 1]$ but not $[0, 1.5]$. We then propose to further add self-loops and row normalize matrix $\tilde{\mathbf{S}}$. This has the following effects

- From the spectral perspective: The proposed normalization further shrinks the spectrum of the matrix to lie in $[0, 1]$, as can be seen in figure 5.2, which makes the filter truly low-pass.
- From the spatial perspective: Each transformed vertex becomes a weighted-average of the neighbors which is more intuitive but it also takes into account column degree information unlike direct random walk adjacency normalization.

We further motivate this choice in the experiments section. Thereby, with this aggregation

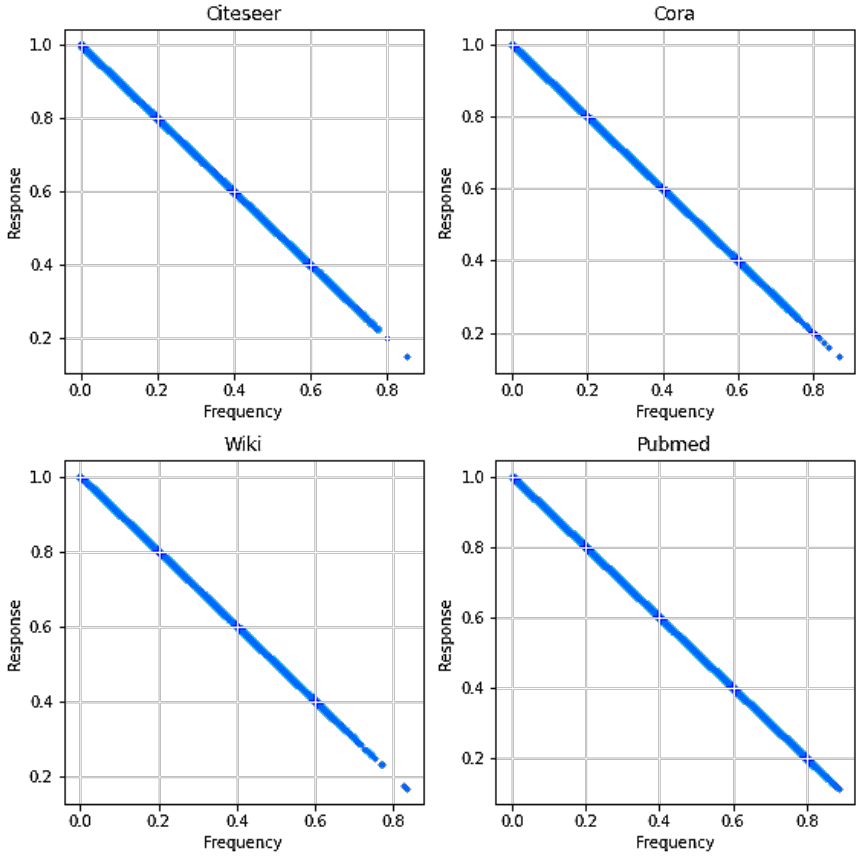


Figure 5.2: frequency response of the proposed GCN filter plotted against the frequency on four real-world datasets

function, our problem turns into

$$\begin{aligned} \min_{\mathbf{G},\mathbf{F},\mathbf{W_1},\mathbf{W_2}} \quad & || \mathbf{T}^p\mathbf{X} - \mathbf{T}^p\mathbf{X}\mathbf{W}_1\mathbf{W_2} ||^2 + \alpha || \mathbf{T}^p\mathbf{X}\mathbf{W}_1 - \mathbf{G}\mathbf{F} ||^2 \\ \text{s.t.} \quad & \mathbf{G} \in \{0,1\}^{n\times k},\ \mathbf{G}\mathbf{1}_k = \mathbf{1}_n \end{aligned} \tag{5.4}$$

Both terms of (5.4) make it possible to express a connection between the two tasks, the first term plays the role of linear autoencoder and the second the role of clustering in the embedding space. We decide in the following to give the same weight for the two terms ($\alpha = 1$).

3.5. Graph Convolutional Clustering

To obtain a mutual supplementation between embedding and clustering, we assume $\mathbf{W} = \mathbf{W_1} = \mathbf{W_2}^\top$ and add an orthogonality constraint on $\mathbf{W}$ in (5.4). It gives rise to the following problem

$$\begin{aligned} \min_{\mathbf{G},\mathbf{F},\mathbf{W}} \quad & || \mathbf{T}^p\mathbf{X} - \mathbf{T}^p\mathbf{X}\mathbf{W}\mathbf{W}^\top ||^2 + || \mathbf{T}^p\mathbf{X}\mathbf{W} - \mathbf{G}\mathbf{F} ||^2 \\ \text{s.t.} \quad & \mathbf{G} \in \{0,1\}^{n\times k},\ \mathbf{G}\mathbf{1}_k = \mathbf{1}_n,\ \mathbf{W}^\top\mathbf{W} = \mathbf{I}_k \end{aligned} \tag{5.5}$$

Similar to [Yamamoto, 2014], solving this problem can be proven to be equivalent to

$$\begin{aligned} \min_{\mathbf{G},\mathbf{F},\mathbf{W}} \quad & || \mathbf{T}^p\mathbf{X} - \mathbf{G}\mathbf{F}\mathbf{W}^\top ||^2 \\ \text{s.t.} \quad & \mathbf{G} \in \{0,1\}^{n\times k},\ \mathbf{G}\mathbf{1}_k = \mathbf{1}_n,\ \mathbf{W}^\top\mathbf{W} = \mathbf{I}_k \end{aligned} \tag{5.6}$$

Proof. We first decompose the reconstruction term

$$\begin{aligned} ||\mathbf{T}^p\mathbf{X} - \mathbf{T}^p\mathbf{X}\mathbf{W}\mathbf{W}^\top||^2 &= ||\mathbf{T}^p\mathbf{X}||^2 + ||\mathbf{T}^p\mathbf{X}\mathbf{W}\mathbf{W}^\top||^2 - 2||\mathbf{T}^p\mathbf{X}\mathbf{W}||^2 \\ &= ||\mathbf{T}^p\mathbf{X}||^2 - ||\mathbf{T}^p\mathbf{X}\mathbf{W}||^2 \quad \text{due to } \mathbf{W}^\top\mathbf{W} = \mathbf{I}_k. \end{aligned}$$

Similarly, the clustering regularization term can be decomposed as follows

$$||\mathbf{T}^p\mathbf{X}\mathbf{W} - \mathbf{G}\mathbf{F}||^2 = ||\mathbf{T}^p\mathbf{X}\mathbf{W}||^2 + ||\mathbf{G}\mathbf{F}||^2 - 2\mathrm{Tr}((\mathbf{T}^p\mathbf{X}\mathbf{W})^\top\mathbf{G}\mathbf{F})$$

Summing the two resulting expressions we get

$$\begin{aligned} ||\mathbf{T}^p\mathbf{X}||^2 + ||\mathbf{G}\mathbf{F}||^2 - 2\mathrm{Tr}((\mathbf{T}^p\mathbf{X}\mathbf{W})^\top\mathbf{G}\mathbf{F}) &= ||\mathbf{T}^p\mathbf{X} - \mathbf{G}\mathbf{F}\mathbf{W}^\top||^2 \\ &\text{due to } ||\mathbf{G}\mathbf{F}\mathbf{W}^\top|| = ||\mathbf{G}\mathbf{F}|| \end{aligned}$$

Thus, optimizing (5.5) is equivalent to optimizing (5.6) □

Before tackling the resolution of this problem in section 4, we will first look at how our proposed GCC approach is related to some other methods.

3.6. Connections to Existing Work

Simple Graph Convolution Variants. Similarly to SGC, the computation of $\mathbf{T}^p\mathbf{X}$ can be considered to be a pre-processing step with a different propagation matrix $\mathbf{T}$. This representation is then used for a downstream task. In the original work that task was classification where the representation is fed to a linear regression model corresponding to a fully connected neural network layer with sigmoid activations. Other variants of the simple graph convolution can also be used as the aggregation function e.g. for the simple spectral graph convolution (S²GC) [Zhu, 2021] we have $\mathtt{agg}(\mathbf{A},\mathbf{X}) = \frac{1}{p}\sum_{i=1}^{p}\mathbf{T}^i\mathbf{X}$.

Graph Autoencoder and Linear Graph Autoencoder. Our model can be seen as a case of the non-probabilistic variant of the VGAE model adapted to graph clustering. Like VGAE, the encoder we use is a form of GCN but rather than an inner-product decoder, we use a linear decoder. The original graph autoencoder was used for link-prediction, i.e., it tried to reconstruct a completed version of the adjacency matrix $\mathbf{A}$. In our case we reconstruct the convolved matrix $\mathbf{T}^p\mathbf{X}$

Deep Clustering Network. From $\mathbf{X}$, the DCN algorithm [Yang, 2017] also performs unsupervised clustering using a deep autoencoder; it uses an optimization objective that is a

weighted combination of a reconstruction error and a clustering error. The DCN cost function is given by

$$\min_{\theta_1,\theta_2,\mathbf{G},\{\mathbf{f}_i\}} \ell(g_{\theta_2}(f_{\theta_1}(\mathbf{x}_i)),\mathbf{x}_i) + \frac{\lambda}{2}\sum_i \mid\mid f_{\theta_1}(\mathbf{x}_i) - \mathbf{G}\mathbf{f}_i \mid\mid_2^2$$
$$\text{s.t.} \quad s_{ij} \in \{0,1\},\ \mathbf{1}^\top \mathbf{f}_i = 1$$

where f is the encoder and g is the decoder, $\{\mathbf{f}_i\}$ are the centroids, $\mathbf{G}$ is a membership matrix and ℓ is a loss function. If we take $\lambda = 2$, the encoder and decoder to be linear functions $f(\mathbf{x};\mathbf{W}) = \mathbf{x}\mathbf{W}$ and $g(\mathbf{x};\mathbf{W}^\top) = \mathbf{x}\mathbf{W}^\top$ with a semi-orthogonality constraint on $\mathbf{W}$, the loss function to be the mean squared error and by considering the observations to be the rows of $\mathbf{T}^p\mathbf{X}$ we get the problem as formulated in (5.5). As for the optimization process, the update rule is the same for the cluster assignment while it differs in the centroids, encoder and decoder updates.

4. Optimization and Algorithm

Directly solving optimization problem in (5.6) is tricky so we use the following alternating iterative approach. The algorithm alternately fixes two of the matrices $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{W}$ and solves for the third one.

4.1. Optimization Procedure

For each matrix, through fixing the two other matrices we obtain a formula which can be solved directly. The solutions to these modified problems are guaranteed to decrease the overall cost function monotonically. The initialization and update rules are described in what follows.

Initialization. We initialize $\mathbf{W}$ with the first f components obtained from applying a randomized *Principal Component Analysis* (PCA) on $\mathbf{T}^p\mathbf{X}$. Matrices $\mathbf{F}$ and $\mathbf{G}$ are then obtained via a k-means on $\mathbf{T}^p\mathbf{X}\mathbf{W}$.

Update Rule for F. By fixing $\mathbf{G}$ and $\mathbf{W}$ and solving for $\mathbf{F}$ we obtain a linear least squares problem. By setting the derivative to zero, we obtain the normal equation which is the optimal solution to the given problem. The update rule is then

$$\mathbf{F} = (\mathbf{G}^\top\mathbf{G})^{-1}\mathbf{G}^\top\mathbf{T}^p\mathbf{X}\mathbf{W}. \tag{5.7}$$

Intuitively, each row vector $\mathbf{f}_i$ is set to the average of the embeddings $\mathbf{X}\mathbf{W}$ that are assigned to cluster i. In the k-means algorithm this corresponds to the centroid update step.

Update Rule for W. By fixing $\mathbf{G}$ and $\mathbf{F}$ and solving for $\mathbf{W}$, the update rule is as

follows

$$\mathbf{W} = \mathbf{U}\mathbf{V}^\top \quad \text{s.t.} \quad [\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] \quad = \texttt{SVD}((\mathbf{T}^p\mathbf{X})^\top \mathbf{G}\mathbf{F}) \tag{5.8}$$

where $\mathbf{\Sigma} = (\sigma_{ii})$, $\mathbf{U}$, and $\mathbf{V}$ are respectively the singular values, the left and right singular vectors of the matrix $(\mathbf{T}^p\mathbf{X})^\top \mathbf{G}\mathbf{F}$.

Proof. Fixing $\mathbf{F}$ and $\mathbf{G}$ in (5.6) leads to the following generalized Procrustes problem

$$\min_{\mathbf{W}} \quad || \mathbf{T}^p\mathbf{X} - \mathbf{G}\mathbf{F}\mathbf{W}^\top ||^2 \quad \text{s.t.} \quad \mathbf{W}^\top\mathbf{W} = \mathbf{I}_k. \tag{5.9}$$

As $||\mathbf{T}^p\mathbf{X} - \mathbf{G}\mathbf{F}\mathbf{W}^\top||^2 = ||\mathbf{T}^p\mathbf{X}||^2 + ||\mathbf{G}\mathbf{F}\mathbf{W}^\top||^2 - 2\text{Tr}(\mathbf{W}\mathbf{F}^\top\mathbf{G}^\top\mathbf{T}^p\mathbf{X})$. and since $||\mathbf{G}\mathbf{F}\mathbf{W}^\top||^2 = ||\mathbf{G}\mathbf{F}||^2$ (5.9) is equivalent to

$$\max_{\mathbf{W}} \quad \text{Tr}(\mathbf{W}\mathbf{F}^\top\mathbf{G}^\top\mathbf{T}^p\mathbf{X}) \quad \text{s.t.} \quad \mathbf{W}^\top\mathbf{W} = \mathbf{I}_k.$$

By taking $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \texttt{SVD}(\mathbf{F}^\top\mathbf{G}^\top\mathbf{T}^p\mathbf{X})$, we have

$$\begin{aligned}
\text{Tr}(\mathbf{W}\mathbf{F}^\top\mathbf{G}^\top\mathbf{T}^p\mathbf{X}) &= \text{Tr}(\mathbf{W}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top) \\
&= \sum_{i=1}^{f} \sigma_{ii} < \mathbf{w}_i'\mathbf{U}, \mathbf{v}_i' > \\
&\leq \sum_{i=1}^{f} \sigma_{ii} ||\mathbf{w}_i'\mathbf{U}|| \times ||\mathbf{v}_i'|| = \sum_{i=1}^{f} \sigma_{ii} = \text{Tr}(\mathbf{\Sigma}).
\end{aligned}$$

This implies that an upper bound for (5.9) is attained when $\text{Tr}(\mathbf{W}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top) = \text{Tr}(\mathbf{\Sigma})$ or equivalently when $\mathbf{V}^\top\mathbf{W}\mathbf{U} = \mathbf{I}$ meaning that the maximum is attained at $\mathbf{W} = \mathbf{V}\mathbf{U}^\top$. □

Update Rule for G. By fixing $\mathbf{F}$ and $\mathbf{W}$ and solving for $\mathbf{F}$, we get a problem that can be optimized with the assignment step of the k-means algorithm. The update rule is, then, given as

$$g_{ij^*} \leftarrow \begin{cases} 1 & \text{if } j^* = \arg\min_j || (\mathbf{T}^p\mathbf{X}\mathbf{W})_i - \mathbf{f}_j ||^2 \\ 0 & \text{otherwise.} \end{cases} \tag{5.10}$$

4.2. The GCC Algorithm

The steps in the GCC algorithm are outlined in Algorithm 3. The convergence of GCC is guaranteed, but it will only reach a local optimum according to the initial conditions. A possible strategy to overcome this is to run GCC several times and to select the best result relative to the objective function. The selection of the propagation order p is integral to the overall performance of the algorithm. A smaller p can mean insufficient neighborhood information is being propagated while a larger p can cause over-smoothing of the graph signal. Figure 5.3 shows projections of the Cora dataset using the t-SNE algorithm [Maaten, 2008] for different values of p (with a perplexity of 50).

Algorithm 3: `GCC`

Input : - Adjacency matrix $\mathbf{A}$
- Feature matrix $\mathbf{X}$
- Propagation order p
- Number of clusters k
- Embedding dimension f
- Tolerance ϵ
- Maximum number of iterations max_iter

Output: - Membership indicator $\mathbf{G} \in \{0,1\}^{n \times k}$
- Embedded centers $\mathbf{F} \in \mathbb{R}^{k \times f}$
- Embedding matrix $\mathbf{W} \in \mathbb{R}^{d \times f}$

Compute $\mathbf{T}$ from $\mathbf{A}$;
Initialize $\mathbf{W}$ with a randomized PCA on $\mathbf{T}^p\mathbf{X}$;
Initialize $\mathbf{G}$ with a k-means on $\mathbf{T}^p\mathbf{X}\mathbf{W}$;
while $|| \mathbf{T}^p\mathbf{X} - \mathbf{G}\mathbf{F}\mathbf{W}^\top ||$ $>$ ϵ **or** max_iter *not reached* **do**
  Update $\mathbf{F}$ using formula (5.7);
  Update $\mathbf{W}$ using formula (5.8);
  Update $\mathbf{G}$ using formula (5.10);
end

With AGC in [Zhang, 2019], the authors proposed to first select an interval of possible values for p and then retain the first p that is a local minimum of an intra-cluster metric. Since our loss function contains information about the clustering performance, it can serve as a metric for the selection of p. Thus, similarly to AGC, we select the p via our loss function as follows: We stop and select $p = p^*$ if the change in square-root of the loss function $||\mathbf{T}^p\mathbf{X} - \mathbf{G}\mathbf{F}\mathbf{W}^\top||$ between p^* and $p^* - 1$ is less than $\frac{d}{n}$ for $p \in \{0, \ldots, 150\}$. The detailed rule is described in Algorithm 4.

As our loss function w.r.t p is always decreasing for every dataset in the interval we chose. We stop when the change in the loss is lower than a constant that is a function of the input dimensions rather than wait for a local minimum.

4.3. Complexity Analysis

In what follows, we analyze the computational complexity of each operation in the `GCC` algorithm as well as the overall one.

Computing $\mathtt{agg}(\mathbf{A}, \mathbf{X})$. The computational complexity of the p-th order simple graph convolution is $\mathcal{O}(p|E|d)$ as each multiplication costs $|E|d$ and p such multiplications are needed.

Initializing W and G. Initializing $\mathbf{W}$ with PCA costs $\mathcal{O}(nd\log(k))$ operations as claimed in [Halko, 2011]. For $\mathbf{G}$, computing $\mathbf{T}^p\mathbf{X}\mathbf{W}$ takes $\mathcal{O}(ndf)$ while k-means applied on $\mathbf{T}^p\mathbf{X}\mathbf{W}$ is in $\mathcal{O}(tnkf)$ where t is the number of iterations of k-means; ergo, the overall complexity of initialization is $\mathcal{O}(nd\log(k) + ndf + tnkf)$.

Updating F. In (5.7), the cost of computing the embeddings matrix $\mathbf{T}^p\mathbf{X}\mathbf{W}$ is $\mathcal{O}(ndf)$.

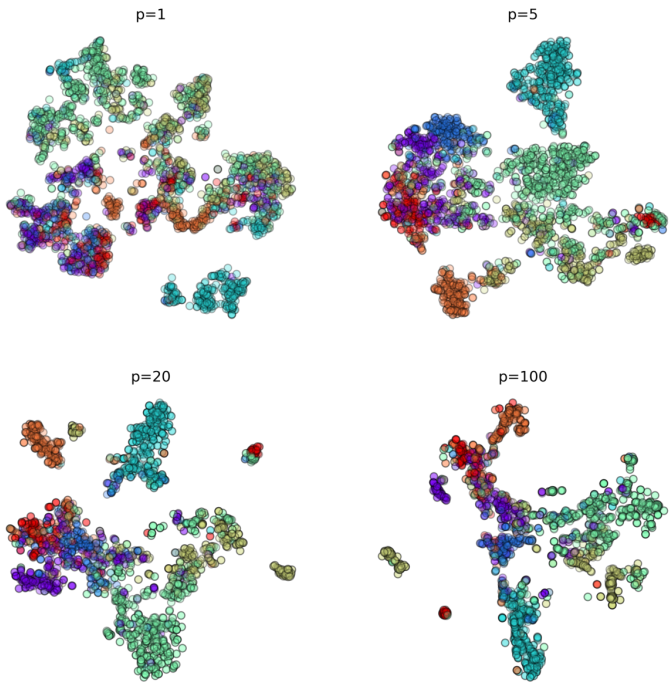


Figure 5.3: Visualization of the Cora GCC-embeddings using t-SNE for different values of p.

Since $\mathbf{G}$ is an indicator matrix, it can be stored as a vector rather than a matrix and multiplications that include it can be replaced by indexing operations. Thus, computing the transformation $(\mathbf{G}^\top\mathbf{G})^{-1}\mathbf{G}^\top$ takes $\mathcal{O}(n+k)$ and applying it on the embeddings $\mathbf{T}^p\mathbf{X}\mathbf{W}$ costs $\mathcal{O}(nf)$. Since $n > k$, the total complexity of the update of $\mathbf{F}$ is then $\mathcal{O}(ndf)$.

Updating W. In (5.8), computing $(\mathbf{T}^p\mathbf{X})^\top\mathbf{G}\mathbf{F}$ for the SVD costs $\mathcal{O}(ndf)$ because of G being indices. The SVD operation itself costs $\mathcal{O}(df^2)$. As for calculating $\mathbf{V}\mathbf{U}^\top$, it is also in $\mathcal{O}(df^2)$. This brings us to a total of $\mathcal{O}(ndf + df^2)$ operations.

Updating G. $\mathbf{T}^p\mathbf{X}\mathbf{W}$ having already being computed, in (5.10) the complexity of computing $\mathbf{G}$ comes from searching each embedded vector's closest centroid, this takes $\mathcal{O}(nkf)$.

Loss computation. Is in $\mathcal{O}(dkf + nd)$ or $\mathcal{O}(ndf)$ depending on the order of the multiplication and the indexing operation in the product $\mathbf{G}\mathbf{F}\mathbf{W}^\top$.

Overall complexity. The totality of the previous operations cost $\mathcal{O}\big(p|E|d+(t'+t)nkf+t'(ndf+df^2+dkf)+nd\log(k)\big)$ where t' is the number of iterations of our algorithm (generally converges within 5-15 iteration).

For simplicity's sake we assume $t' = t$, we can also assume that $k, f \leq \min(n, d)$, which is often the case in graph datasets (this condition can always be satisfied by adding duplicate nodes or constant features), this allows us to set $f = k$. Consequently, the total complexity is given as $\mathcal{O}(p|E|d + tndk)$.

Algorithm 4: Propagation order selection rule

Input : - Adjacency matrix $\mathbf{A}$
- Feature matrix $\mathbf{X}$
- Number of clusters k
- Embedding dimension f

Output: Propagation order p^*

for $p \in \{2, \ldots, 100\}$ **do**
  $\mathbf{G}, \mathbf{F}, \mathbf{W} \leftarrow \texttt{GCC}(\mathbf{A}, \mathbf{X}, p, k, f)$;
  $\text{loss}_p \leftarrow \; || \mathbf{T}^p\mathbf{X} - \mathbf{GFW}^\top ||$;
  if $| loss_p - loss_{p-1} | < \frac{d}{n}$ **then**
    $p* \leftarrow p - 1$
  end
end

Table 5.1: Dataset statistics.

Dataset	#Nodes	#Edges	#Features	#Classes
CiteSeer [Sen, 2008]	3327	4732	3703	6
Cora [Sen, 2008]	2708	5429	1433	7
PubMed [Sen, 2008]	19717	44338	500	3
Wiki [Yang, 2015]	2405	17981	4973	17

In comparison, computing $\mathbf{T}^p\mathbf{X}$ and applying a k-means on it takes $\mathcal{O}(p|E|d + tndk)$, the same as our method. In practice, however, our algorithm is significantly faster than the k-means algorithm as its most theoretically heavy computations are matrix multiplications which can be efficiently performed on GPUs.

5. Experiments

To evaluate our proposed model, we conduct experiments on four datasets and compare it against a number of state-of-the-art approaches for the node clustering task.

5.1. Datasets

We evaluate GCC on four widely-used attributed network datasets (Cora, Citeseer, Pubmed and Wiki). The nodes in Cora and Citeseer are associated with binary word vectors, while the ones in Pubmed and Wiki with tf-idf weighted word vectors. The summary statistics of the datasets are shown in table 5.1.

Table 5.2: Clustering performance on four datasets averaged over 20 runs. AGE was averaged over 3 runs. AGE, LAE and LVAE failed to scale to Pubmed; OOM denotes 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**

Table 5.3: Wall-clock time in seconds for different methods on the four datasets averaged over 20 runs (3 runs for AGE).

Method	CiteSeer	Cora	Pubmed	Wiki
Sph. k-means	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

5.2. A Fair Comparison with Baseline Methods

In our work we focus on clustering and related methods. Below we look at how our GCC algorithm performs in comparison with state-of-the art unsupervised methods. Approaches that use information from the actual labels, be it supervised or semi-supervised, are not considered such as [Velickovic, 2019]. The baseline methods are categorized as follows:

1. **Methods that use node-level features only.** Spherical k-means [Hornik, 2012] is k-means applied on data projected on the unit sphere. It will serve as the node features clustering baseline along with DCN [Yang, 2017].

2. **Methods that use graph structure only.** Spectral is the spectral clustering algorithm with the normalized Laplacian as the input similarity matrix.

3. **Methods that use both.** LVAE [Salha, 2020] is the linear graph variational autoencoder and LAE is its non-probabilistic version. GIC [Mavromatis, 2021], AGE [Cui, 2020] proposes a Laplacian smoothing filter that acts as a low-pass filter applied in adaptive learning scheme. S²GC proposes a new method for the aggregation of K-hop neighborhoods that is a trade-off of low- and high-pass filter bands, it then applies spectral clustering on the output of that operation.

In the experiments we use the implementations that are publicly available on Github repositories of the authors.

5.3. Experimental Settings

To evaluate the clustering results, we employ three performance metrics: clustering Accuracy (Acc), Normalized Mutual Information (NMI) and clustering macro F1-score (F1). Larger values imply better performance. We report the mean values of the three metrics for each algorithm over 20 executions except for AGE which we average over three runs because of high execution time.

For our model, we set the embedding dimension $f = k$ for each dataset. The propagation parameter p is selected via the heuristic rule described prior; we obtain $p = 5$ for citeseer, $p = 12$ for Cora, $p = 150$ for Pubmed and $p = 4$ for Wiki. We row normalize the feature vectors and use tf-idf normalization on the binary word vectors of Citeseer and Cora so that all datasets are in tf-idf.

For other methods, we employ the parameters recommended by the authors for every dataset, For S²GC we expand the possible propagation order to $\{1, \ldots, 150\}$, the same as `GCC` for a fair comparison. All models were run on the same machine with a 12GB memory GPU and a RAM of 12GB. Note that we could not run AGE, LAE and LVAE on Pubmed due to out of memory (OOM) issues.

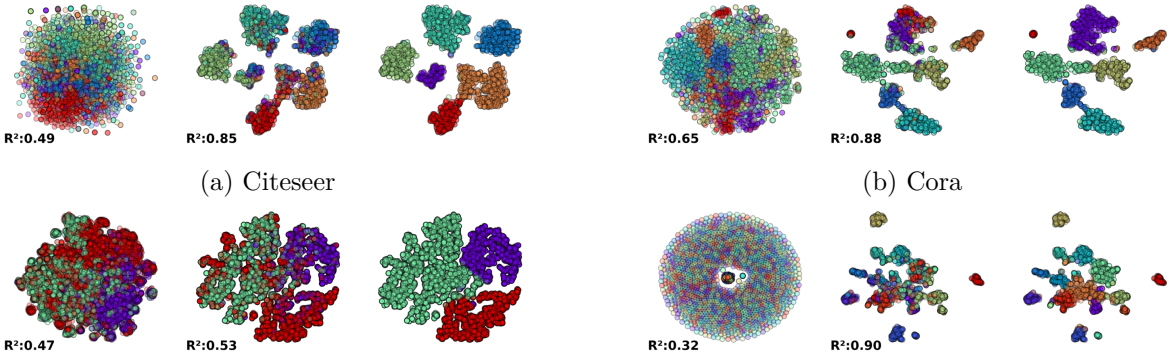
5.4. Clustering Results

Clustering performances of the different methods are reported in table 5.2. Best performances are shown in bold. It is clear that the methods that use both $\mathbf{A}$ and $\mathbf{X}$ perform significantly better than those that use either of them individually. We see how `GCC` outperforms other attributed graph clustering methods in terms of accuracy, F1 and NMI except for Pubmed where S²GC and `GCC` are comparable. The algorithm is also stable as its standard deviation on the accuracy is 0.13 on Citeseer, 0.02 on Cora, 0.00 on Pubmed, and 1.58 on Wiki.

We also report the average running time of each algorithm in table 5.3. Notice how the running times of our algorithm are the lowest on the four datasets when compared to the state-of-the-art especially when comparing with the AGE algorithm. We mentioned earlier how our algorithm has a similar theoretical complexity to that of k-means. We can see here that in practice our algorithm is faster on most datasets due to the fact that it can be efficiently run on a GPU.

5.5. Embedding and Visualization

The GCC model offers the ability to display the cluster-based structure inherent to multivariate data. Figure 5.4 presents the lower-dimensional representations produced by our



(a) Citeseer (b) Cora

(c) Pubmed (d) Wiki

Figure 5.4: *Left column:* t-SNE projection of the original features colored according to the real labels. *Middle column:* t-SNE projection of the GCC embeddings colored according to the real labels. *right column:* t-SNE projection of the GCC embeddings colored according to the predicted labels. R-squared is used to measure of class separability for real classes (left and middle column), e.g., 0.49 vs 0.85 for Citeseer.

model projected on a 2-d space by t-SNE (with a perplexity of 50). We can see a clear difference in the structures of the projections of the raw data and those of the generated embeddings. To further judge the quality of the embedding, we use the R-squared measure, i.e., $R^2=\frac{\mathrm{Tr}(S_b)}{\mathrm{Tr}(S_t)}$, where S_b is the between-class scatter matrix and S_t is the total scatter matrix. We report this measure in Figure 5.4 on the true labels plots to quantify separability. The R-squared is larger for 2-d projections of the GCC embeddings on all four datasets. On Pubmed, the structure is less pronounced (0.47 vs 0.53) but we can still see the formation of three clusters.

This shows how our model can be efficiently used for data visualization to generate more interpretable embeddings or for a dimensionality reduction step before feeding the output representations to more complex clustering algorithms down the line. Note also that such visualisations can help the user in assessing the number of clusters.

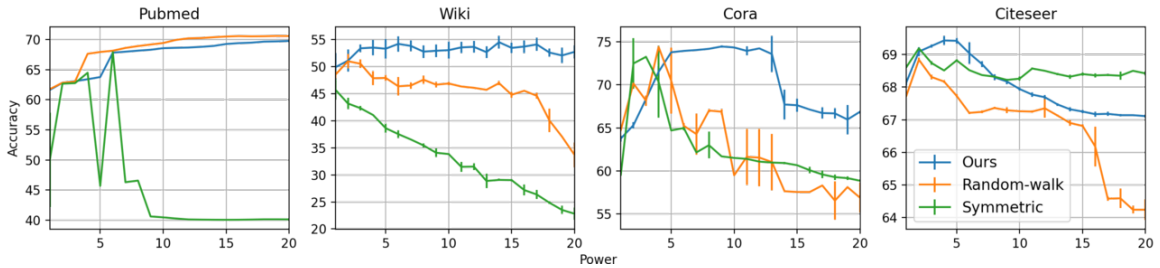


Figure 5.5: Accuracy with GCC using different propagation matrices averaged over 20 runs

5.6. Choice of Propagation Matrix

We conduct experiments to further motivate our choice of propagation matrix. We compare the following propagation matrices: *i)* Augmented symmetric norm. $\mathbf{A}_{\mathrm{sym}}$ =

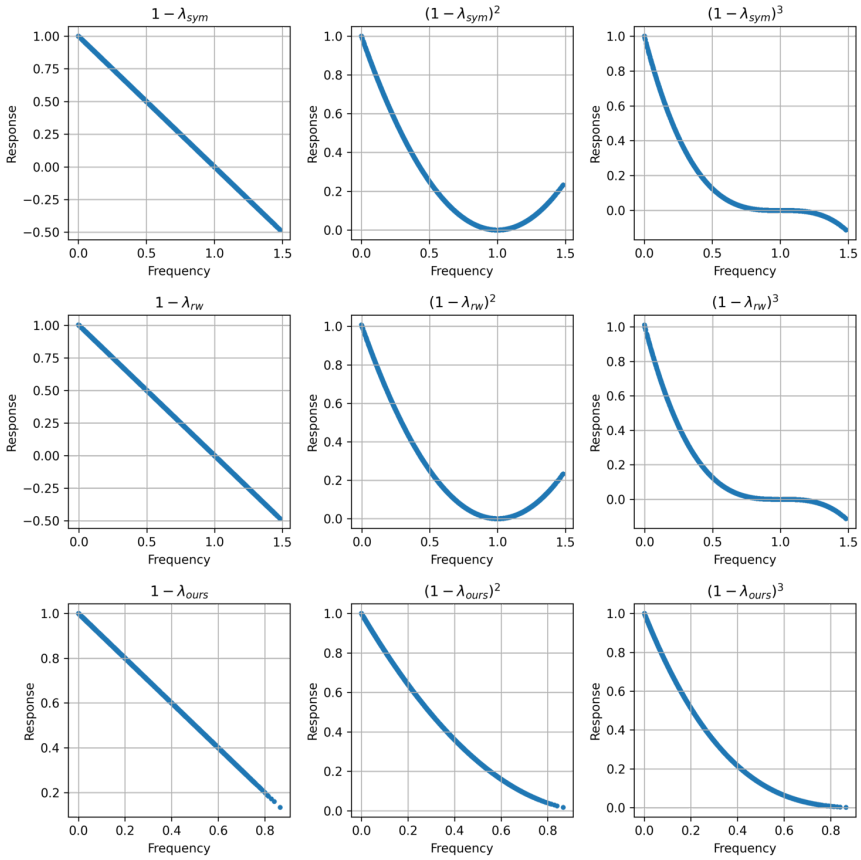


Figure 5.6: Frequency response plotted against the frequency for different propagation matrices on Cora. *Left column*: frequency response is $1-\lambda$. *Middle column*: frequency response is $(1-\lambda)^2$. *Right column*: frequency response is $(1-\lambda)^3$.

$\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$. *ii)* Augmented random walk norm. $\mathbf{A}_{\text{rw}} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$. *iii)* Our norm. $\mathbf{A}_{\text{ours}} = \mathbf{T} = {\mathbf{D}_{\mathbf{T}}}^{-1}(\mathbf{I} + \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2})$. We do an analysis on the clustering accuracy of our model with these normalizations for propagation orders $p \in \{1, .., 20\}$. These results are averaged over 20 runs and the same parameters are used for all normalizations.

We see in figure 5.5 how our proposed propagation matrix offers the maximum accuracy on three out of the four datasets ($\mathbf{A}_{\text{rw}}$ slightly outperforms it on Pubmed). We also see that it is more stable and well-behaved compared to the other two on all four datasets. The symmetric normalization especially is prone to large changes even for consecutive propagation orders. These results can be explained by the fact that the GCN when using the symmetric and random walk normalizations is not strictly low-pass. Figure 5.6 shows the frequency response functions for the GCN with the three propagation matrices for propagation order $p \in \{1, 2, 3\}$. We see how the absolute value of the frequency response function is not always decreasing w.r.t the frequencies for $\mathbf{A}_{\text{rw}}$ and $\mathbf{A}_{\text{sym}}$ as opposed to $\mathbf{A}_{\text{ours}}$.

6. Conclusion

In this work, we harnessed the simple formulation of the graph convolutional network to obtain an efficient model that addresses both node embedding and clustering in a unified framework. First, we provided a normalization that makes the GCN encoder act as a low pass filter in the strict sense. Secondly, we proposed a novel approach where the objective function to be optimized leverages information from both the GCN embedding reconstruction loss and the cluster structure of these embeddings. Thirdly, we derived `GCC` whose complexity has been rigorously studied. In doing so, we showed how `GCC` achieves better performances compared to other graph clustering algorithms in a more efficient manner. Note that all the compared methods are unsupervised in nature in order to have a fair comparison with our model. Our experiments demonstrated the interest of our approach. We also showed how GCC is related to other methods including some GCN variants.

The proposed model is flexible and can be extended in several directions, thus opening up opportunities for future research. For instance, in our approach we have assumed that the α coefficient which regulates the trade-off between seeking reconstruction and clustering is equal to one, it would be interesting to investigate the choice of this value. On the other hand, while our focus in this work is clustering, so it would be worthwhile to extend the problem, e.g., to co-clustering, which is useful in a wide range of real-world scenarios like document clustering.

The following chapter considers a specific type of attributed graphs, namely, bipartite attributed graphs. This is the same type of graph we considered in this chapter but with an additional specificity which is that its vertices are divided into two disjoint and independent subsets.

Chapter 6

Attributed Graph Subspace Clustering

Objective

This chapter was published as [Fettal, 2023b]. Our goal was to create an efficient yet effective approach for the clustering attributed-graph nodes based on Laplacian smoothing, as well as subspace clustering through learning factor matrices, and using nonnegative explicit kernel feature maps.

Contents

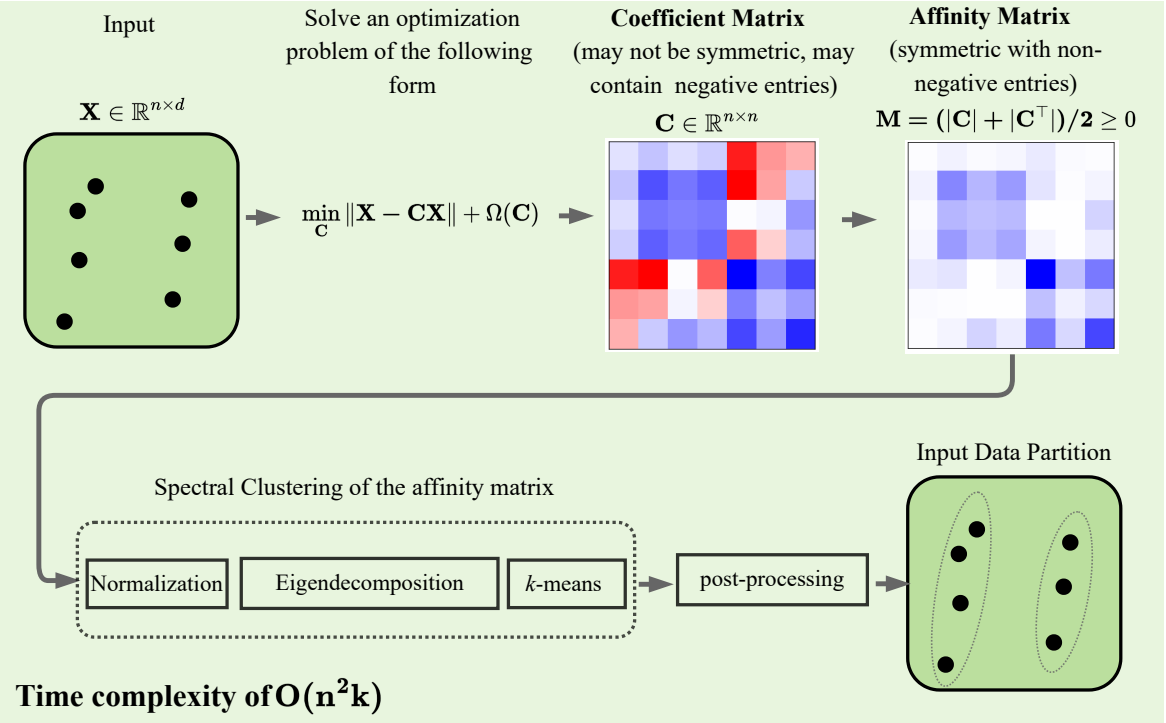


Figure 6.1: The traditional subspace clustering pipeline. A coefficient matrix $\mathbf{C}$ is initially learned. An affinity matrix $\mathbf{M}$ is then generated based on the magnitudes of $\mathbf{C}$, e.g., a common choice for the affinity is $\mathbf{M} = \frac{|\mathbf{C}|+|\mathbf{C}^\top|}{2}$. Finally, a partition of the data is carried out via spectral clustering on the affinity matrix $\mathbf{M}$.

1. Introduction

An attributed-graph is a type of graph that contains two information sources, a topology or structure and node- and/or edge-level features. It is a convenient representation for many real-world networks, with applications in the fields of recommender systems [Fan, 2019; Ying, 2018], computer vision [Satorras, 2018; Yang, 2018a], Natural language processing [Marcheggiani, 2017] and physical systems [Hoshen, 2017].

With the advent of the Graph Convolutional Network (GCN) [Defferrard, 2016; Kipf, 2017], graph related tasks such as graph representation learning [Wu, 2019; Zhu, 2021] and graph clustering [Anton Tsitsulin, 2020] have received a lot of attention. We observe, however, that for the task of graph clustering, few approaches [Cai, 2020] based on the subspace clustering principle have been proposed despite it being, at first sight, well-suited to attributed-graph data. We argue that this is mostly due to subspace clustering models suffering from high spatial and/or computational complexity. In a nutshell, the goal of subspace clustering is to group data points according to the subspaces in which they lie within a dataset. For example, subspace clustering models that use the self-expressive property, whereby every data point can be represented as an approximate linear combination of other points, have to learn a square matrix called the *coefficient* or *self-representation* matrix. This coefficient matrix has a size that is quadratic in the number of points. Once this matrix is learned, an affinity

matrix is constructed from it and spectral clustering is performed on said affinity matrix. We can see the classical subspace clustering pipeline in figure 6.1.

In this work, we argue that subspace clustering is well-suited to attributed-graph representations generated with GCN-based models due the neighborhood averaging making the data points closer and thus helping with the self-expressiveness of the data points. To leverage this property and in order to avoid the complexity problems associated with traditional subspace clustering, we propose an efficient variant to learn an initial representation of the graph before applying an efficient self-expressive subspace clustering procedure via learning a factored coefficient matrix and then projecting these factors into a new feature space in such a way as to generate a valid affinity matrix (symmetric with non-negative entries) on which to perform implicit spectral clustering. A schema for our model is available in figure 6.3. To showcase the efficacy and efficiency of our proposal, we perform extensive experimentation on six widely used attributed-networks. We can see a preview of the results in figure 6.2, these are the clustering results of our model on the arXiv open graph benchmark, our model yields a 14% improvement over the second best model in terms of performance and 16% improvement in terms of speed. Code for our work can be found in [1].

This work is organized as follows: Section 2 reviews related works. Section 3 presents the necessary previous work. Section 4 is devoted to the proposed model and its computational complexity study. In section five, we carry out our experimental study. Finally we present our conclusion in section 6.

2. Related Work

2.1. Subspace Clustering

Subspace clustering methods based on the self-expressive property are commonly used on image data and have set state-of-the-art results on the task of image clustering. One of the earlier approaches was the Least-Square Regression subspace clustering (LSR) that leverages a grouping effect in the data. Newer models that make up the state-of-the-art include the Elastic-net Subspace Clustering (EnSC) [You, 2016a] that uses a mix of l1- and l2-norm regularization, and the Subspace Clustering through the Orthogonal Matching Pursuit (SSC-OMP) [You, 2016b] which possesses a subspace-preserving affinity under broad conditions. There are also deep learning approaches like the deep Subspace clustering network [Ji, 2017] and but these models have received some critique to the effect that their good performances are the result of an ad-hoc post processing step instead of the actual self-representation learning process [Haeffele, 2021]. More recently, a new efficiency trend has appeared, and some scalable models have also been proposed e.g. k-Factorization Subspace Clustering (k-FSC) [Fan, 2021] which was put forward as a scalable subspace clustering model that factorizes data into subsets via structured sparsity.

[1] https://github.com/chakib401/sagsc

2.2. Attributed-Graph Clustering

In this work, attributed-graph clustering refers to the process of grouping nodes into clusters according to the graph topology and node features. We can classify attributed-graph clustering models into two subsets. A first one, where the goal is to learn graph representations and then use traditional clustering models such as k-means. Examples of models that use this approach include Simplified Graph Convolution (SGC) [Wu, 2019] which proposes a neighborhood averaging process that corresponds to a fixed low-pass filter, and the Simple Spectral Graph Convolution (S^{2}GC) which uses a new method for the aggregation of K-hop neighborhoods that is a trade-off of low- and high-pass filter bands. [Zhu, 2021]. On the other hand, the second class of attributed-graph clustering models proposes to include the clustering objective into the representation learning process to learn better results, e.g., Graph InfoClust (GIC) [Mavromatis, 2021] which generates clusters by maximizing mutual information between nodes contained in the same cluster, and Graph Convolutional Clustering (GCC) [Fettal, 2022c] that performs clustering by minimizing the difference between convolved node representations and their reconstructed cluster representatives.

3. Preliminaries

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ be an undirected attributed-graph where V is the vertex set consisting of nodes $\{v_1, \ldots, v_n\}$, E is the set of edges that connects the nodes, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a symmetric adjacency matrix where a_{ij} denotes the edge weight between nodes v_i and v_j, if $a_{ij} = 0$ then there is no edge between v_i and v_j, and $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a node-level feature matrix. Our goal is to partition this graph into k independent subsets in an unsupervised manner.

3.1. Graph Convolutional Networks

The graph Convolutional Network consists in a sequence of propagation layers. It can be formalized recursively as

$$\begin{aligned} &\mathbf{H}^{(l+1)} \leftarrow \sigma\left(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \\ &\text{with} \quad \mathbf{H}^{(0)} = \mathbf{X} \end{aligned} \tag{6.1}$$

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loop the, $\hat{\mathbf{D}}$ is its diagonal matrix of degrees, σ is some activation function and $\mathbf{W}^{(l)}$ is the weight matrix of the l-th layer. These weight matrices are optimized for some downstream task like semi-supervised classification, link prediction, etc.

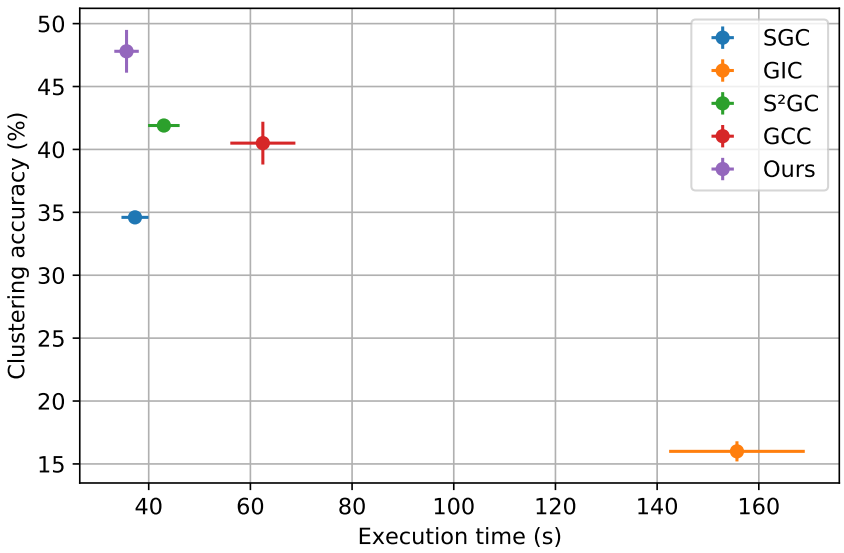


Figure 6.2: Clustering accuracy scores (%) plotted against the execution time (s) for our method and the state-of-the-art attributed-graph clustering models on the OGBN-arXiv dataset.

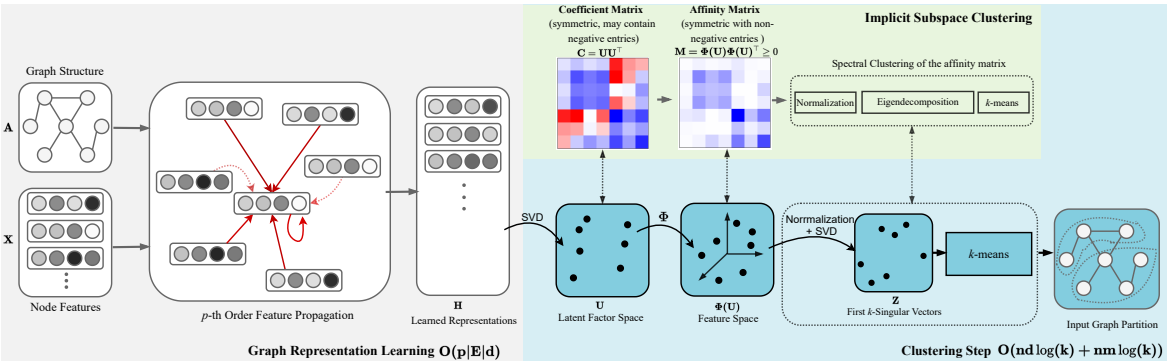


Figure 6.3: Diagram of our proposal. We have as input an attributed-graph characterized by an adjacency matrix $\mathbf{A}$ and a feature matrix $\mathbf{X}$. An initial representation $\mathbf{H}$ of the attributed-graph is learned through neighborhood propagation. Then, subspace clustering is performed using a latent factor matrix $\mathbf{U}$ where $\mathbf{M} = \mathbf{U}\mathbf{U}^\top$ is the subspace coefficient matrix that we project using a quadratic kernel feature map Φ so that $\mathbf{D}^{-1/2}\Phi(\mathbf{U})\Phi(\mathbf{U})^\top\mathbf{D}^{-1/2} \geq 0$. With this we obtain the final partition by using the k-means algorithm on $\mathbf{Z}$, the first k singular vectors (not counting the first one) of $\mathbf{D}^{-1/2}\Phi(\mathbf{U})^\top$.

3.2. Simplified Graph Convolutional Networks

Authors in [Wu, 2019] argued that the non-linearities in the GCN are superfluous and that most of its performance comes from the feature propagation. With this, the recursive of a p-layer GCN collapses into this single formula

$$\mathbf{H} \leftarrow \mathbf{S}^p\mathbf{X}\mathbf{W}$$

where $\mathbf{S} = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}$ is called the propagation matrix. Here, we can see how the weights matrices collapsed into a single weight matrix $\mathbf{W}$ while the graph propagation steps collapsed into the p-th power of the propagation matrix $\mathbf{S}$.

3.3. Subspace Clustering

The goal of subspace clustering is to group data points according to the subspaces that support them. A popular formulation uses the self-expressive property where it is assumed that a data point can be written as a linear combination of the data points that belong in the same subspace. A possible formulation is

$$\min_{\mathbf{C}\in\mathcal{C}} \quad \|\mathbf{X} - \mathbf{C}\mathbf{X}\|^2 + \Omega(\mathbf{C}) \tag{6.2}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is a matrix of d-dimensional data points, $\mathbf{C} \in \mathbb{R}^{n\times n}$ is known as the *self-representation* or *coefficient matrix*, $\Omega(\mathbf{C})$ is a regularization term introduced to establish certain properties for $\mathbf{C}$ e.g. to avoid trivial solutions (such as $\mathbf{C} = \mathbf{I}$), and $\mathcal{C}$ is the feasible region.

Once a solution $\mathbf{C}$ is found, an affinity matrix is generated based on the magnitudes of the entries of $\mathbf{C}$, a popular choice for this is $|\mathbf{C} + \mathbf{C}^\top|/2$. Finally, a clustering of the data points is obtained using some graph clustering algorithm such as the spectral clustering algorithm [Shi, 2000].

4. Proposed Approach

In this work, we propose the following generic formulation for the attributed-graph subspace clustering problem

$$\min_{\mathbf{C}\in\mathcal{C}} \quad \|\mathsf{agg}(\mathbf{A},\mathbf{X}) - \mathbf{C}\,\mathsf{agg}(\mathbf{A},\mathbf{X})\| + \Omega(\mathbf{C}) \tag{6.3}$$

where agg is an aggregation function whose role is to merge the two information sources: the topology information and the feature information present in the graph.

4.1. Simple Graph Convolutional Encoder

We propose to use a GCN-based encoder. More particularly, we use the convolution operation proposed in the simplified graph convolutional network along with the normalization of the adjacency matrix used in [Fettal, 2022c]

$$\min_{\mathbf{C}\in\mathcal{C}} \quad \|\mathbf{S}^p\mathbf{X} - \mathbf{C}\,\mathbf{S}^p\mathbf{X}\| + \Omega(\mathbf{C}). \tag{6.4}$$

Now that we have our initial graph representation, we can present our clustering step.

4.2. Efficient Subspace Clustering

4.2.1 Learning the implicit coefficient matrix

We set constraints on $\mathbf{C}$ in order to obtain a decomposition of $\mathbf{C}$ into the Gramian product $\mathbf{U}\mathbf{U}^\top$ where $\mathbf{U} \in \mathbb{R}^{n\times k}$ is a semi-orthogonal matrix i.e. $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$. This will allow us to significantly speed up the subspace clustering process. Thus, our problem becomes

$$\min_{\mathbf{U}} \quad \|\mathbf{S}^p\mathbf{X} - \mathbf{U}\mathbf{U}^\top\,\mathbf{S}^p\mathbf{X}\| \quad \text{such that} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{I}. \tag{6.5}$$

As we can see, we have no need for any form of regularization. This problem can be efficiently solved through a singular value decomposition of the convolved features $\mathbf{S}^p\mathbf{X}$. With this we obtain a solution $\mathbf{C}$ which corresponds to a subspace coefficient matrix from which we can derive a clustering of the nodes.

4.2.2 Learning the implicit affinity matrix

Once we have a coefficient matrix $\mathbf{C} = \mathbf{U}\mathbf{U}^\top$, we have to derive a nonnegative matrix that reflects the magnitudes of the entries of $\mathbf{C}$. As already mentioned the common way is to compute $(|\mathbf{C}|+|\mathbf{C}^\top|)/2$ but this will result in a spectral clustering step which has a quadratic complexity in the number of nodes. In this work, we propose to use a nonnegative kernel with feature map Φ to embed $\mathbf{U}$ into its feature space explicitly

$$\mathbf{M} = \Phi(\mathbf{U})\Phi(\mathbf{U})^\top \geq 0. \tag{6.6}$$

Here the feature map is applied row-wise, for example, in our experiments, we used the quadratic kernel

$$\mathbf{M} = \mathbf{C}^{\circ 2} = (m_{ij}) = (c_{ij}^2). \tag{6.7}$$

It is also possible to introduce a bias term b to the kernel such as

$$m_{ij} = (c_{ij} + b)^2.$$

Hence, we have implicitly derived a Gramian decomposition through $\Phi(\mathbf{U})$ similarly to what was done for $\mathbf{C}$. This will allow us to efficiently perform the last step which corresponds to spectral clustering. Note that $\mathbf{M}$ is symmetric by construction.

4.2.3 Spectral clustering the implicit affinity matrix

Through the previous step we can now efficiently perform the NJW spectral clustering [Ng, 2001] on matrix $\mathbf{M}$ by:

- Projecting the factor $\mathbf{U}$ using feature map Φ, i.e., $\mathbf{Q} \leftarrow \Phi(\mathbf{U})$
- Computing $\tilde{\mathbf{Q}} \leftarrow \mathbf{Q}\mathbf{D}^{-\frac{1}{2}}$ where $\mathbf{D}$ is a diagonal matrix such that d_{ii} is the sum of $\mathbf{M}$ i-th row.
- Constructing $\mathbf{Z}$ using the left singular vectors corresponding to the second to $k+1$-largest singular values of $\tilde{\mathbf{Q}}$.
- Performing a clustering of the rows of $\mathbf{Z}$ and assigning node i to cluster j if the i-th row of $\mathbf{Z}$ was assigned to cluster j.

4.3. Complexity Analysis

Our overall algorithm is presented in algorithm 5. In what follows, we will analyze the computational complexity of our proposal

Table 6.1: Complexity of the different models. For k-FSC, m referes to the dimension of subspaces. For k-FSC, many possible complexities are possible depending on the chosen algorithm, please see [Fan, 2021] for a discussion on its complexity. For simplicity, we suppose that the embedding dimension in SGC, S²GC and GCC is in $\mathcal{O}(k)$.

Method	**Time complexity**	**Space complexity**
k-means	$\mathcal{O}(ndk)$	$\mathcal{O}(n(k+d))$
LSR	$\mathcal{O}(n^2k)$	$\mathcal{O}(n^2)$
EnSC	$\mathcal{O}(n^2k)$	$\mathcal{O}(n^2)$
SSC-OMP	$\mathcal{O}(n^2k)$	$\mathcal{O}(n^2)$
SGC	$\mathcal{O}(p\|E\|d+ndk)$	$\mathcal{O}(n(k+d))$
S²GC	$\mathcal{O}(p\|E\|d+ndk)$	$\mathcal{O}(n(k+d))$
GCC	$\mathcal{O}(p\|E\|d+ndk)$	$\mathcal{O}(n(k+d))$
SAGSC	$\mathcal{O}(p\|E\|d+n(d+m)\log(k)+nk^2)$	$\mathcal{O}(n(k+d+m))$

Graph representation learning step To compute the p-th order graph convolution, we need $\mathcal{O}(p|E|d)$ operations.

Learning the implicit coefficient matrix Getting the left singular values of the convolved data requires $\mathcal{O}(nd\log(k))$ operations using the randomized singular value decomposition [Halko, 2011].

Learning the implicit affinity matrix The projection of the data using a feature kernel of dimensionality m takes $\mathcal{O}(nm)$. The computation of the diagonal matrix $\mathbf{D}$ and its multiplication with $\mathbf{Q}$ takes $\mathcal{O}(nm)$ operations. The truncated singular value decomposition of $\hat{\mathbf{Q}}$ is in $\mathcal{O}(nm\log(k))$. Finally, the k-means algorithm applied on $\mathbf{Z}$ costs roughly $\mathcal{O}(nk^2)$. The overall computation time of this step $\mathcal{O}(nm\log(k)+nk^2)$.

Overall complexity. The totality of our algorithm costs
$\mathcal{O}\left(p|E|d+n(m+d)\log(k)+nk^2\right)$. Generally, we have that $k << d$. The dimension m generally depends on k, for example in the case of the quadratic kernel $m=\binom{k+2}{2}=\frac{(k+2)(k+1)}{2}$. In other cases, when wishing to use nonnegative infinite dimensional kernels such as the RBF kernel or higher order polynomial kernels, feature map approximation techniques such as Nyströem method [Zhang, 2008] or the polynomial count sketch [Pham, 2013] can be used and m becomes a variable hyper-parameter.

In table 6.1, we can see how the complexity of our algorithm compares with that of the other models. Despite our model being using subspace clustering, it is significantly more efficient than the other subspace clustering models both in terms of computational and spatial complexity. When comparing with the SOTA attributed-graph clustering models, we can see that when $m \in \mathcal{O}(d)$ then our model has the same complexity as them. Which means that when taking a smaller m, e.g., $m \in \mathcal{O}(k)$, then our model should be more computationally efficient.

Algorithm 5: Scalable Attributed-Graph Subspace Clustering.

Input : $\mathbf{X}$ data matrix, $\mathbf{S}$ propagation matrix, p propagation order, k number of clusters, Φ nonnegative kernel feature map.
Output: π partition of the nodes.
$\mathbf{H} \leftarrow \mathbf{S}^p\mathbf{X}$;
Form the matrix $\mathbf{U}$ containing the first k left singular vectors of $\mathbf{H}$ in its rows;
$\mathbf{Q} \leftarrow \Phi(\mathbf{U})$;
$\mathbf{r} \leftarrow \mathbf{Q}^\top\mathbf{1}$;
$\mathbf{D} \leftarrow \texttt{diag}(\mathbf{Qr})$;
$\hat{\mathbf{Q}} \leftarrow \mathbf{QD}^{-\frac{1}{2}}$;
Form the matrix $\mathbf{Z}$ containing left singular vectors corresponding to the second to $k+1$-th largest singular values of $\hat{\mathbf{Q}}$ in its rows;
Apply a clustering algorithm on the rows of $\mathbf{Z}$ to obtain π a partition of the data;

5. Experiments

In this section, we conduct experimentation to showcase the effectiveness and efficiency of our LGCSC model.

Table 6.2: The datasets statistics. The imbalance is quantified via the ratio between the majority and minority classes.

Dataset	**Nodes**	**Edges**	**Features**	**Classes**	**Imbalance**
ACM	3025	16,153	1870	3	1.1
Wiki	2405	14,001	4973	17	45.1
DBLP	4057	2,502,276	334	4	1.6
Amazon Computers	13,381	259,159	767	10	17.5
Pubmed	19,717	64,041	500	3	1.9
OGBN-arXiv	169,343	1,327,142	128	40	942.1

Table 6.3: Clustering performance of the different models over ACM, DBLP and Wiki. Best results are highlighted in bold font and second best results are underlined.

Method	**Input**	**ACM**			**DBLP**			**Wiki**		
		CA	NMI	ARI	CA	NMI	ARI	CA	NMI	ARI
k-means	$\mathbf{X}$	87.8±0.9	61.7±1.5	67.4±2.1	67.9±0.0	37.3±0.0	31.5±0.1	47.6±1.4	48.6±0.2	26.6±0.2
LSR	$\mathbf{X}$	78.6±0.0	43.1±0.0	48.3±0.0	69.4±0.1	34.7±0.1	36.4±0.2	17.8±0.5	2.8±1.7	0.3±0.2
EnSC	$\mathbf{X}$	83.8±0.0	53.0±0.0	58.6±0.0	30.0±0.1	0.8±0.2	0.1±0.0	47.5±0.0	45.2±0.2	30.2±0.1
SSC-OMP	$\mathbf{X}$	82.1±0.0	49.4±0.1	55.3±0.0	29.4±0.1	0.4±0.1	-0.1±0.0	37.8±8.5	34.4±9.1	21.2±7.9
k-FSC	$\mathbf{X}$	59.7±7.2	25.2±7.1	27.2±7.2	51.3±11.1	17.4±7.3	17.3±9.6	38.2±5.1	35.6±3.9	17.7±4.4
SC	$\mathbf{A}$	36.5±0.2	1.0±0.2	0.7±0.1	91.0±0.0	73.0±0.1	78.3±0.1	30.7±1.1	24.0±0.8	6.0±0.2
SGC	$\mathbf{A},\mathbf{X}$	83.7±0.0	55.7±0.0	58.8±0.0	88.8±0.0	69.5±0.0	73.2±0.0	51.9±0.8	49.6±0.2	28.6±0.1
GIC	$\mathbf{A},\mathbf{X}$	90.1±0.3	68.2±0.6	73.2±0.6	90.2±0.2	72.4±0.4	77.4±0.3	48.0±0.7	48.4±0.3	31.0±0.3
S²GC	$\mathbf{A},\mathbf{X}$	84.1±0.1	56.8±0.1	59.6±0.2	88.3±0.0	69.2±0.0	71.9±0.0	52.1±1.0	52.2±0.1	33.0±0.4
GCC	$\mathbf{A},\mathbf{X}$	91.3±0.0	71.2±0.1	76.0±0.1	91.8±0.0	74.5±0.0	80.5±0.0	53.7±1.4	**53.5±0.5**	31.6±1.1
SAGSC	$\mathbf{A},\mathbf{X}$	**93.3±0.1**	**75.1±0.2**	**80.9±0.1**	**93.1±0.1**	**78.1±0.2**	**83.2±0.2**	**56.0±2.1**	53.2±1.2	**34.1±2.7**

5.1. Datasets and Metrics

In our experiments, We use six commonly used benchmark datasets to compare the different models including three citation network datasets (ACM, DBLP [Wang, 2019]; PubMed [Sen, 2008]; and Wiki [Yang, 2015]), an Amazon sales dataset (Computers) [Shchur, 2018] and one large scale dataset (OGBN-arXiv) [Hu, 2020]. The summary statistics of the datasets are shown in table 6.2.

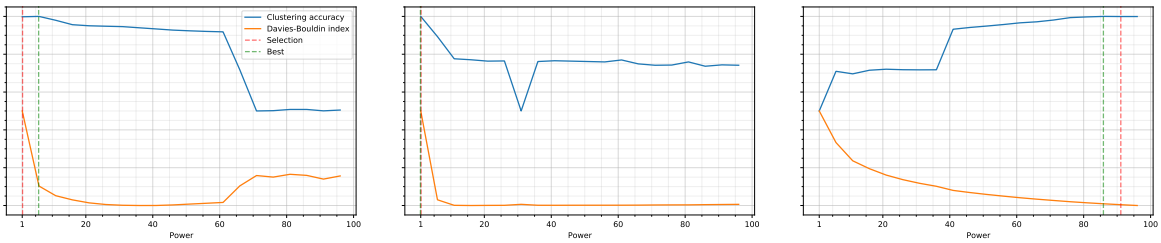
We adopt three popular clustering evaluation metrics: clustering accuracy (CA), normalized mutual information (NMI) [Strehl, 2002], adjusted rand index (ARI) [Hubert, 1985].

Table 6.4: Clustering performance of the SOTA models over the larger networks; Amazon Computers, Pubmed and OGBN-arXiv. Best results are highlighted in bold font and second best results are underlined.

Method	**Input**	**Amazon Computers**			**PubMed**			**OGBN-arXiv**		
		CA	NMI	ARI	CA	NMI	ARI	CA	NMI	ARI
SGC	$\mathbf{A},\mathbf{X}$	65.5±0.0	52.2±0.0	45.7±0.0	69.6±0.0	29.3±0.0	29.9±0.0	34.6±0.4	39.2±0.1	25.2±0.6
GIC	$\mathbf{A},\mathbf{X}$	46.8±2.2	47.5±0.9	31.3±3.5	64.5±0.4	26.2±0.3	23.8±0.4	16.0±0.8	17.9±0.5	5.8±0.2
S²GC	$\mathbf{A},\mathbf{X}$	65.4±0.0	55.4±0.0	**49.5±0.0**	71.0±0.0	**32.9±0.0**	33.7±0.0	41.9±0.3	45.9±0.1	36.9±0.5
GCC	$\mathbf{A},\mathbf{X}$	67.6±0.0	56.0±0.0	46.5±0.0	70.5±0.0	32.2±0.0	33.1±0.0	40.5±1.7	46.8±0.2	35.1±2.0
SAGSC	$\mathbf{A},\mathbf{X}$	**69.0±1.0**	**58.2±0.4**	48.2±1.8	**71.1±0.0**	**32.9±0.0**	**34.1±0.0**	**47.8±1.7**	**47.1±0.5**	**38.4±1.6**

Table 6.5: Execution time of all methods in seconds. Best results are highlighted in bold.

Method	ACM	DBLP	Wiki	Pubmed	Computers	OGBN-arXiv
k-means	4.29±0.7	6.05±0.7	24.25±0.3	-	-	-
LSR	20.14±0.26	5.2±0.64	46.55±1.61	-	-	-
EnSC	590.31±63.93	120.66±0.28	232.58±3.04	-	-	-
SSC-OMP	201.78±34.8	37.1±2.87	293.78±9.63	-	-	-
k-FSC	3.72±0.87	8.45±0.73	34.29±1.86	-	-	-
SC	2.15±0.32	18.54±1.30	2.86±0.44	-	-	-
SGC	0.56±0.13	0.19±0.04	1.68±0.14	1.18±0.39	1.00±0.11	37.30±2.66
GIC	3.67±0.16	268.96±63.17	5.96±0.66	12.0±1.50	22.38±1.51	155.7±13.34
S²GC	0.44±0.04	0.23±0.06	1.56±0.10	0.82±0.10	1.33±0.28	42.98±3.10
GCC	1.73±2.96	0.33±0.10	1.66±0.14	1.26±0.11	1.92±0.13	62.45±6.40
SAGSC	**0.40±0.05**	**0.18±0.05**	**1.07±0.09**	**0.79±0.05**	**0.88±0.12**	**35.64±2.41**



(a) ACM. (b) DBLP. (c) Pubmed.

Figure 6.4: Plot of the clustering accuracy (%) and the Davies-Bouldin index [Davies, 1979] against the propagation power.

5.2. Baseline Models and algorithms

The following are the baselines we used in our experiments:

- k-**Means** will serve as the simplest baseline.
- **LSR** is a subspace clustering model with an l2-norm regularization.
- **EnSC** is a subspace clustering model with an elastic net regularization (mix of l1- and l2-norm regularization).
- **SSC-OMP** has a subspace-preserving affinity under broad conditions.
- **k-FSC** is a scalable subspace clustering model that factorizes data in subsets via structured sparsity.
- **SC** refers to the classical spectral clustering algorithm applied on the original adjacency matrix of the graph.
- **SGC** proposes a neighborhood averaging process that corresponds to a fixed low-pass filter.
- **GIC** generates clusters by maximizing mutual information between nodes contained in the same cluster.

- **S²GC** proposes a new method for the aggregation of K-hop neighborhoods that is a trade-off of low- and high-pass filter bands.
- **GCC** performs clustering by minimizing the difference between convolved node representations and their reconstructed cluster representatives.

We use the implementations of the authors when possible.

5.3. Experimental Settings

All experiments were implemented in TensorFlow and conducted on a standard computer with a 12GB memory GPU and a RAM of 12GB. In all experiments, we ran the models ten times, and reported the average performance along with the corresponding standard deviation. We use the implementations of the authors when possible but optimized them to run on GPU. We used hyper-parameters prescribed by authors when possible. For fairness, for the remaining hyper-parameters, we ran grid searches and reported the results corresponding to the best accuracy for all models. For k-FSC, we use the LARGE implementation. All results are the averages of ten runs.

For our model, we use a quadratic kernel feature map with a bias term equal to $\frac{1}{\sqrt{2}}$. This leads to the following kernel feature map:

$$\begin{aligned} \varphi : \mathbb{R}^k &\to \mathbb{R}^{\binom{k+2}{2}} \\ \mathbf{x} &\mapsto \langle x_k^2, \dots, x_1^2, x_k x_{k-1}, \dots, x_k x_1, x_{k-1} x_{k-2}, \\ &\quad \dots, x_{k-1} x_1, \dots, x_2 z_1, x_k, \dots, x_1, \tfrac{1}{\sqrt{2}} \rangle \end{aligned} \tag{6.8}$$

For the power hyper-parameter, similarly to the other benchmarks, we use a grid search over the accuracy and report the best results. We do however propose a heuristic to adaptively select this hyper-parameter.

5.4. Node Clustering Results

Performance Clustering performances of the different methods are reported in tables 6.3 and 6.4. Best performances are highlighted in bold while second best results are underlined. In table 6.3, there is a general trend that the methods that use both $\mathbf{A}$ and $\mathbf{X}$ perform better than those that use $\mathbf{A}$ or $\mathbf{X}$ individually, except on DBLP where they perform well. Our model has the best performance over the three datasets with respect to all three metrics. The GCC has the second best results in all but one case, i.e., ARI on Wiki where it is outperformed by S²GC. In table 6.4, the three datasets are of larger sizes, our model has the best results in eight out of nine cases although our model has the second best result on the remaining case (ARI over Amazon Computers), note that for NMI over PubMed we have a tie between S²GC and our model. On the largest dataset OGBN-arXiv, our model shows a 14% improvement over the second best model, S²GC.

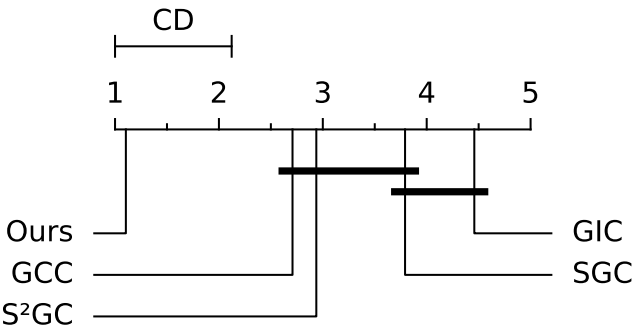


Figure 6.5: Results of the Nemenyi test where each rank represents the average rank over the CA, NMI, ARI and clustering F1-score; and the six datasets. We see that our model achieves the best rank, and is alone in the best performing group. We can also see the formation of two other groups.

Efficiency In table 6.5, we report the training times of all the baselines over ACM, DBLP and Wiki, and report those of the SOTA over PubMed, Computers and OGBN-arXiv. Our model is the fastest one on all datasets. Two main observations can be made. First, our model is significantly faster than other subspace clustering models including the more efficient ones like k-FSC. Second, our model is as fast as the SOTA attributed-graph clustering models despite it being based on subspace clustering which is known to be computationally heavy.

Analysis Overall, our model is as fast as the fastest attributed-graph clustering models while consistently yielding the best overall performance on all six datasets. This shows the cost-effective nature of our model with respect to the state of the art.

To back up this claim statistically, we use the Nemenyi post-hoc test [Nemenyi, 1963] to find groups of models that perform similarly in a statistically meaningful manner, to do this we rank the performances of the different models w.r.t to each metric (CA, NMI, ARI, and clustering F1-score) for each dataset. This yields 24 different rankings. We then carry-out the Nemenyi test with a confidence level of 90%. Results are illustrated in figure 6.5. We see the formation of three groups. The first one contains the best performing model, SAGSC; a second one, containing GCC, S²GC and SGC; and a third one containing SGC and GIC.

5.5. Selection of the Power Hyper-Parameter

The selection of the power parameter is integral to the performance of our model. A power that is too small can lead to not enough neighborhood information being propagated and a power that is too large can lead to the oversmoothing phenomenon [Chen, 2020]. Since in the unsupervised context, it impossible to know for certain which power will lead to the best performance, several heuristics for the selection of this hyper-parameter have been proposed, e.g., in [Zhang, 2019], authors proposed to use internal criteria based on the information intrinsic to the data while in [Fettal, 2022c] authors proposed to choose a cutoff threshold on the change of their loss function between successive powers. Here, we propose to use

an approach similar to the elbow method [Ketchen, 1996] which is used for the selection of the number of clusters in the k-means algorithm. We start by choosing an interval for the powers we wish to consider e.g. the multiples of five plus one between one and a hundred i.e. $\{1, 6, \ldots, 96\}$. Then we choose the power that precedes the appearance of the first pronounced 'elbow' in the graph. if there is no elbow, we choose the upper bound of the interval.

For example, in figure 6.4, we can see a clear elbow for ACM, DBLP when the power is equal to six so we have the power to one. In the case of Pubmed, no such elbow appears and so we set the power 96. We see that with very simple rules, we reach an accuracy that is almost the same as the best one. For DBLP, we retrieve the best power, while for ACM and PubMed, the differences between the accuracy of the power we retrieved and the best one are 0.23 and 0.02, respectively, which is negligible. Of course, after this initial selection, a more granular selection can be performed since here we used an interval with a crude spacing of five between consecutive powers. Note that this selection process can be easily automated.

6. Conclusion

In this work, we leveraged subspace clustering for attributed-graphs through the means of an efficient algorithm whereby after learning an initial representation of the graph through a simple yet effective neighborhood propagation step. We learn a factored coefficient matrix through orthogonal constraints, these factors are then embedded into a new feature space in such a way as to create a symmetric and nonnegative affinity matrix on which an implicit spectral clustering algorithm is performed. We additionally showed how this overall clustering process corresponds to an implicit subspace clustering algorithm. The experimentation we conducted showed the effectiveness and efficiency of our proposal with respect to the state of the art attributed-graph clustering algorithms.

In the next chapter, we show how taking into consideration the clustering objective in the graph representation learning task can lead to better results on both the clustering and embedding tasks.

Chapter 7

Attributed Bipartite Graph Subspace clustering

Objective

This chapter is based on [Fettal, 2024a] an extended version of [Fettal, 2022d]. Our goal was to create an efficient yet effective approach for the co-clustering of attributed-graph features based on Laplacian smoothing, as well as subspace co-clustering through learning factor matrices, and using nonnegative explicit kernel feature maps.

Contents

1. Introduction

As the datasets become more and more larger and can be in addition more sparse, adaptations to existing clustering algorithms are required to maintain cluster quality. In fact, this quality can greatly suffer in the high dimensional data where many dimensions are often irrelevant. This appears in many applications including recommendation systems, microarray or even textual data represented as document-term matrices. In the text area, the task of text clustering is always important and has many use cases including fake news detection, sentiment analysis, information retrieval and so on; see for instance [Aggarwal, 2012] for more applications. Thereby *subspace clustering* [Parsons, 2004] and *biclustering/co-clustering* [Dhillon, 2001; Riverain, 2022] techniques have shown that can leveraged to uncover the complex relationships found in such data.

Subspace clustering is an unsupervised learning method in which points are to be grouped according to the subspaces in which they lie. A variety of approaches have been used to solve this problem, and a number of these approaches are based on a self-expressive formulation where it is assumed that each element can be written as a linear combination of the elements in the subspace. Based on a self-expressive formulation, subspace clustering methods have been widely used to cluster image datasets, given that image datasets will often be drawn from multiple low-dimensional subspaces, and state-of-the-art clustering results have in many cases been obtained. Regarding text data, however, to the best of our knowledge no self-expressive subspace clustering approaches have been proposed that are specifically tailored to text, although the assumption made in relation to image data applies equally to text data. We argue that this discrepancy between image and text stems from two causes:

- *Complexity.* Document-term datasets are usually very large, much more so than figs/scc, which makes it prohibitive to use subspace clustering algorithms whose computational complexity is usually in $\mathcal{O}(n^3)$, and whose spatial complexity is in $\mathcal{O}(n^2)$.
- *Sparsity.* Document-term datasets are sparser than image datasets, and each individual data point may thus potentially lie in a unique subspace, making it difficult for subspace clustering algorithms to group points meaningfully.

In this work we propose a subspace clustering model tailored for networked (and non-networked) text data based on the principle of *co-clustering* (or *biclustering*), that is to say using the interplay between rows and columns [Govaert, 2013]. In the context of text, this consists in harnessing the interplay between the set of documents and the set of terms to jointly generate a partitioning for both of them. This leads to reorganize the initial data matrix into a new one that is reorganized into homogeneous co-clusters/biclusters. The choice of the co-clustering approach is justified for at least four reasons a) co-clustering overcomes the curse of dimensionality and sparsity; by alternatingly updating the row partition given the column partition and vice versa the clustering can be performed in lower dimensional space and therefore more parsimonious than one-sided clustering performed on the row or column sets separately [Govaert, 2008] b) the clusters/co-clusters tend to be more easily interpretable,

allowing the user to better direct further study [Dhillon, 2001; Salah, 2018; Fettal, 2022b] c) fuzzy co-clustering can be easily deduced from many co-clustering methods [Govaert, 2013] allowing, for instance, to assign a document/term to several classes, and finally d) in terms of topic modelling, the obtained results from co-clustering are in line with the human judgment, outperforming, in general, the conventional LDA (Allocation Dirichlet Allocation) method [Jelodar, 2019]; see for instance [Ailem, 2017]. In this regard, other researchers emphasize that sparse datasets are not suitable for LDA [Chen, 2019b].

In our approach, we address the two major issues that are inherent to subspace clustering on text data: complexity and sparsity. To this end, we use factorized representation matrices and nonnegative kernel feature maps, as well as a bilateral graph convolution that includes a weighted Laplacian smoothing preprocessing step, having similarities with a simple graph convolutional network [Defferrard, 2016; Kipf, 2017; Wu, 2019]. Second, combining subspace and co-clustering, we propose an efficient extension to [Fettal, 2023b] to co-clustering and expand the proposal presented in [Fettal, 2022d]. This makes our model particularly well suited to clustering attributed graphs whose nodes and/or edges have attributes or features. This leads to tackle networked text data/text attributed graphs that are used to model a wide variety of real-world networks such as in recommender systems [Fan, 2019], citation networks [Sen, 2008] and so on. We summarize our contributions as follows:

- We study how the Laplacian smoothing operation boosts the grouping effect of subspace clustering approaches that possess this property.
- Unlike the iterative process generally adopted in co-clustering, we show here that in our formulation of the self-expressive subspace co-clustering problem the optimal solution can be derived from a single truncated singular value decomposition, which makes it efficient (linear complexity in the number of nodes).
- We carry out extensive experimentation on text attributed graphs where the graphs exist on the one hand and when the graphs are generated from the node features on the other using k-nearest neighbor graphs. This allows to demonstrate the flexibility and value of the proposed model, in terms of document /word clustering capability and computational efficiency.
- Making the code available for download to ensure reproducibility of the results [1].

The remainder of this work is structured as follows. Section 2 discusses related works. Section 3 develops our proposed method SC^3, and section 4 discusses the algorithm and its complexity. Section 5 contains a detailed description of our experiments. Finally, we give our conclusion in section 6.

[1] https://github.com/chakib401/sc3

2. Related Works

Our contributions can be seen as being at the intersection between subspace clustering, co-clustering, and attributed graph clustering.

2.1. Self-expressive Subspace Clustering

Among the earliest approaches was Least Squares Regression (LSR) subspace clustering [Lu, 2012], which leverages a grouping effect based on the correlation of data. More sophisticated approaches that represent today's state of the art were later proposed, such as Elastic-net Subspace Clustering (EnSC) [You, 2016a], subspace clustering by Orthogonal Matching Pursuit (SSC-OMP) [You, 2016b], the ℓ^0-norm regularized subspace clustering (ℓ^0-SCC) [Yang, 2018b] and its recent extension that deals with noisy data (Noisy-DR-ℓ^0-SCC-LR) [Yang, 2022]. Some recent works have proposed efficient methods such as K-Factorization Subspace Clustering (K-FSC) [Fan, 2021] and others were created to deal with multi-view data [Wang, 2015; Wang, 2018c; Wang, 2018b]. Some models like the GAN-Based Enhanced Deep Subspace Clustering Networks [Yu, 2020] explicitly make the assumption of dealing with image data. Note that deep self-expressive subspace clustering models have received criticism over the necessity of a neural network component [Haeffele, 2021].

2.2. Co-clustering

Co-clustering seeks to form *co-clusters* which are sets of homogeneous sets of rows and columns. It harnesses the inherent duality between the rows and columns of data tables, which can lead to improvements in partitioning for both dimensions. For example, in the case of document-term matrices, co-clustering incorporates term space information that is used in the document partitioning, and vice versa. A popular method is by alternatingly finding a clustering for the rows while taking into consideration the current clustering of columns, and inversely. One of the first co-clustering approaches was the spectral co-clustering algorithm [Dhillon, 2001]. Directional Co-clustering with Constraints (DCC) [Salah, 2017a] is based on a regularized von Mises-Fisher mixture model that makes it suitable for balanced text datasets. Regularized Dual-PPMI Co-clustering (RDPCo) [Affeldt, 2021] extends DCC to incorporate both word-word semantic relationships and document-document similarities into the procedure (see [Govaert, 2013] for a review). Finally, Consensus Factorization for Co-Clustering Networked Data (CFOND) [Guo, 2018] is a consensus factorization model that factorizes information simultaneously from three sources: network topology structures, instance-feature content relationships, and feature-feature correlations.

2.3. Attributed Graph Clustering

Attributed Graph clustering involves grouping nodes into clusters depending on the structure of the graph and node-level features. In Graph-InfoClust (GIC)[Mavromatis, 2021] clustering is done by maximizing the mutual information between nodes contained in the same cluster. Simple Spectral Graph Convolution (S²GC) [Zhu, 2021] is a method for the aggregation of K-hop neighborhoods that is a trade-off between low- and high-pass filter bands. Graph Convolutional Clustering (GCC) [Fettal, 2022c] is a procedure that simultaneously clusters and learns clustering-friendly representations of nodes. In addition, CFOND, mentioned above in relation to co-clustering, is also considered to be an attributed graph clustering model.

3. Preliminaries and Background

Matrices are denoted using boldface uppercase, and vectors using boldface lowercase letters. Given a matrix $\mathbf{X}$, its i-th row is denoted as $\mathbf{x}_i$ and its j-th column as $\mathbf{x}'_j$. $\mathbf{I}_n$ is the identity matrix of size n. The Frobenius norm is denoted as $\|.\|$. $\texttt{rk}$ gives the rank of a matrix. Function $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \texttt{SVD}(\mathbf{X})$ gives the compact singular value decomposition of matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, where $\mathbf{U} \in \mathbb{R}^{n \times \texttt{rk}(\mathbf{X})}$ and $\mathbf{V} \in \mathbb{R}^{d \times \texttt{rk}(\mathbf{X})}$ have the left and right singular vectors in their columns and $\mathbf{\Sigma} \in \mathbb{R}^{\texttt{rk}(\mathbf{X}) \times \texttt{rk}(\mathbf{X})}$ is the diagonal matrix containing the singular values, sorted in decreasing order. We also define function $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \texttt{TruncatedSVD}(\mathbf{X}, k)$ that returns the first (largest) k singular values, and the left and right singular vectors. Function $\texttt{diag}$ creates a diagonal matrix from a vector input, while $\mathbf{1}$ denotes a vector of ones.

3.1. Self-Expressive Subspace Clustering

Given a $\mathbf{X} \in \mathbb{R}^{n \times d}$ a matrix of d-dimensional data points. The self-expressive subspace clustering is typically formulated as

$$\min_{\mathbf{R}} \quad \| \mathbf{X} - \mathbf{R}\mathbf{X} \|^2 + \Omega(\mathbf{R}) \quad \text{such that} \quad \mathbf{R} \in \mathcal{R} \tag{7.1}$$

where $\mathbf{R} \in \mathbb{R}^{n \times n}$ is known as the self-representation matrix, $\Omega(\mathbf{R})$ serves as a regularization term designed to establish certain properties for $\mathbf{R}$ so as to avoid trivial solutions (such as $\mathbf{R} = \mathbf{I}$), and $\mathcal{R}$ is the feasible region. After an optimal solution $\mathbf{R}^*$ has been obtained, an affinity matrix is first generated based on the magnitudes of the entries in $\mathbf{R}^*$, usually using $|\mathbf{R}^* + \mathbf{R}^{*\top}|/2$, and a partition of the points is then generated using a graph clustering method such as the spectral clustering algorithm [Shi, 2000].

3.2. Block seriation

Co-clustering can be posed as a *block seriation* problem [Marcotorchino, 1987] whose objective is finding a block diagonal matrix $\mathbf{R} \in \mathbb{R}^{n \times d}$ that identifies co-clusters.

The block seriation problem can be stated as this integer program:

$$\max_{\mathbf{R}} \quad \sum_{ij} x_{ij} r_{ij} \tag{7.2}$$

subject to:

$$\forall\, i, j \qquad r_{ij} \in \{0, 1\} \qquad \text{Binarity}$$

$$\left.\begin{array}{ll} \forall\, i, j, i', j' & r_{ij} + r_{ij'} + r_{i'j'} - r_{i'j} \leq 2 \\ & r_{i'j'} + r_{i'j} + r_{ij} - r_{ij'} \leq 2 \\ & r_{i'j} + r_{ij} + r_{ij'} - r_{i'j'} \leq 2 \\ & r_{ij'} + r_{i'j'} + r_{i'j} - r_{ij} \leq 2 \end{array}\right\} \qquad \text{No triads}$$

$$\left.\begin{array}{ll} \forall\, j & \sum_i r_{ij} \geq 1 \\ \forall\, i & \sum_j r_{ij} \geq 1 \end{array}\right\} \qquad \text{Assignment}$$

The solution $\mathbf{R}$ is always block diagonal up to a permutation of the rows and columns. An equivalent formulation consists in factorizing $\mathbf{R}$ using two assignment matrices $\mathbf{Z}$ and $\mathbf{W}$:

$$\max_{\mathbf{Z}, \mathbf{W}} \quad \sum_{ij} x_{ij} \mathbf{z}_i^\top \mathbf{w}_j \equiv \operatorname{trace}(\mathbf{Z}^\top \mathbf{X} \mathbf{W}) \tag{7.3}$$

$$\text{s.t.} \quad \mathbf{Z} \in \{0, 1\}^{n \times k}, \qquad \mathbf{Z}\mathbf{1} = \mathbf{1} \tag{7.4}$$

$$\mathbf{W} \in \{0, 1\}^{d \times k}, \quad \mathbf{W}\mathbf{1} = \mathbf{1} \tag{7.5}$$

For example, a simple heuristic for solving this problem consists in using block coordinate descent, to iteratively solve for the row assignments while the column assignments, and vice versa. This approach shows, how given a clustering of the rows and columns, it is possible to obtain a diagonal co-clustering of the data matrix [Laclau, 2017a].

3.3. Neighborhood Propagation & Graph Convolutional Networks

let $\mathcal{G} = (\mathbf{A}, \mathbf{X})$ be a node-attributed graph, with $\mathbf{A}$ the adjacency and $\mathbf{X}$ the node features. The graph convolutional network (GCN) consists of the following step repeated for each layer

$$\mathbf{H}^{(l+1)} \leftarrow \sigma\left(\mathbf{S}\mathbf{H}^{(l)}\mathbf{W}\right) \quad \text{such that} \quad \mathbf{H}^{(0)} \leftarrow \mathbf{X}$$

where σ is some activation function, $\mathbf{W}$ are learnable weights that depend on the task at hand, and $\mathbf{S}$ is a normalized version of $\mathbf{A}$ with self-loops added. The simplified version, on the other hand, uses a GCN with linear activations and no weights. For example, the simple equivalent of a GCN with p-layers is

$$\mathbf{H} \leftarrow \mathbf{S}^p\mathbf{X}$$

These representations can then be used for some downstream task.

4. Proposed Method

Here, we derive a model that combines the concepts of self-expressive subspace clustering, co-clustering and neighborhood propagation. The resulting model surpasses the performance of state of the art methods for all three categories on document-term matrices.

4.1. Self-Expressive Subspace Co-clustering

Based on the Block Seriation model for co-clustering, given a document-term matrix $\mathbf{X} \in \mathbb{R}_+^{n\times d}$, the self-expressive subspace co-clustering problem can be formulated as

$$\begin{aligned} \min_{\mathbf{R},\mathbf{C}} \quad & \| \mathbf{X} - \mathbf{R}\mathbf{X}\mathbf{C} \|^2 + \Omega(\mathbf{R}, \mathbf{C}) \\ \text{such that} \quad & \mathbf{R} \in \mathcal{R},\ \mathbf{C} \in \mathcal{C}. \end{aligned} \tag{7.6}$$

where $\mathbf{R} \in \mathbb{R}^{n\times n}$ and $\mathbf{C} \in \mathbb{R}^{d\times d}$ are respectively the row and column self-representation matrices, $\Omega(\mathbf{R}, \mathbf{C})$ is the regularization term where the regularization of $\mathbf{R}$ and $\mathbf{C}$ can be either independent (i.e., $\Omega(\mathbf{R}, \mathbf{C}) = \Omega_{\mathbf{R}}(\mathbf{R}) + \Omega_{\mathbf{C}}(\mathbf{C})$) or dependent, and $\mathcal{R}$ and $\mathcal{C}$ are the feasible regions. Note that unlike the Block Seriation model, the generic case of our problem does not require k and g, the numbers of row and column clusters respectively, to be equal.

4.2. Promoting the Grouping Effect Through a Bilateral Graph Convolution

The performance of subspace clustering methods [Lu, 2012; Hu, 2014; Lu, 2013] is due to the grouping effect. While the authors in [Lu, 2012; Lu, 2013] optimized this property implicitly, [Hu, 2014] sought to enforce it explicitly through the regularization term $\Omega(\mathbf{R})$. We use the definition of the grouping effect given in [Hu, 2014].

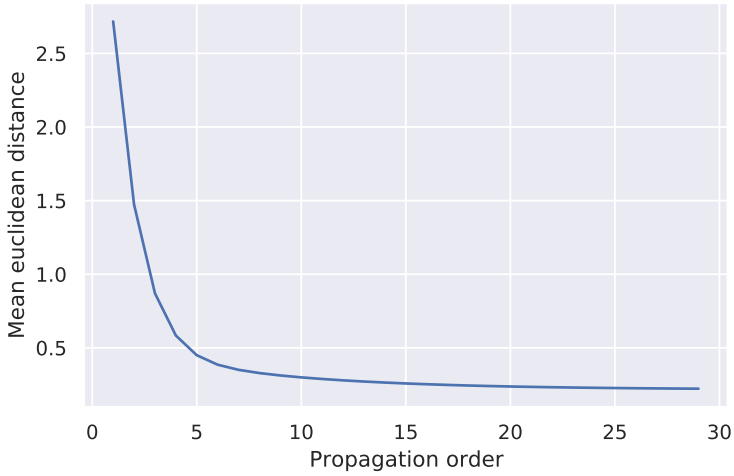


Figure 7.1: Mean pairwise euclidean distance of the columns of $\mathbf{S}^p\mathbf{X}$ as p increases on DBLP. The columns get mutually closer as more information propagates over the rows.

Definition 4.1. *(Grouping Effect) Given a data matrix* $\mathbf{X}$*, a self-representation matrix* $\mathbf{R}$ *has a grouping effect if*

$$\forall i, j,\ \|\mathbf{x}_i - \mathbf{x}_j\| \to 0 \implies \|\mathbf{r}_i - \mathbf{r}_j\| \to 0. \tag{7.7}$$

In the case of text, the grouping effect will not necessarily be beneficial, because of its high dimensionality and sparsity. Data points may not be sufficiently "close" (as regards the self-expressive property) to be grouped in a meaningful way. This means that even if a subspace clustering approach has the grouping effect, in practice $\|\mathbf{x}_i - \mathbf{x}_j\| \to 0$ is not likely, making the property useless. The implication is that data points need some sort of smoothing to make some of the points closer and consequently to help subspace clustering algorithms find common subspaces.

Here, we propose to solve this problem through a bilateral graph convolution preprocessing step, based on simple graph convolution. This requires two similarity matrices to act as graphs on the rows $\mathbf{S_R}$ and columns $\mathbf{S_C}$. These matrices can either be constructed through some similarity measure on the data, e.g. using the k-nn approach using the Euclidean or cosine distance, or be provided *a priori* such as in the case of attributed graphs (on the rows at least).

The reasoning, intuitively, is that the rows and columns of $\mathbf{S}_\mathbf{R}^p\mathbf{X}\mathbf{S}_\mathbf{C}^q$, as propagation orders p, q grow, become smoother by being averaged up to their p-th and q-th neighbors respectively, analogously to Laplacian smoothing. The rows and columns therefore become more similar as powers increase. An illustration of this is shown in figure 7.1.

Proposition 6. *Given a row-normalized adjacency matrix* $\mathbf{S}$ *on the rows of* $\mathbf{X}$*, we have that* $\mathbf{S}$ *is a non-expansive mapping on the columns of* $\mathbf{X}$

$$\forall i \neq j, \quad \|\mathbf{S}\mathbf{x}'_i - \mathbf{S}\mathbf{x}'_j\| \leq \|\mathbf{x}'_i - \mathbf{x}'_j\|$$

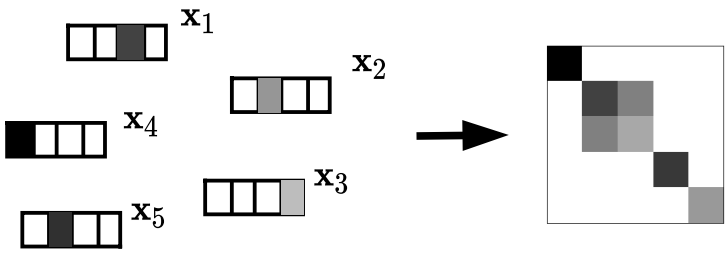
The same holds for the rows of $\mathbf{X}$ *for* $\mathbf{S}$ *an adjacency matrix over the columns of* $\mathbf{X}$*.*

$$\forall i \neq j, \quad \|\mathbf{x}_i\mathbf{S} - \mathbf{x}_j\mathbf{S}\| \leq \|\mathbf{x}_i - \mathbf{x}_j\| \tag{7.8}$$

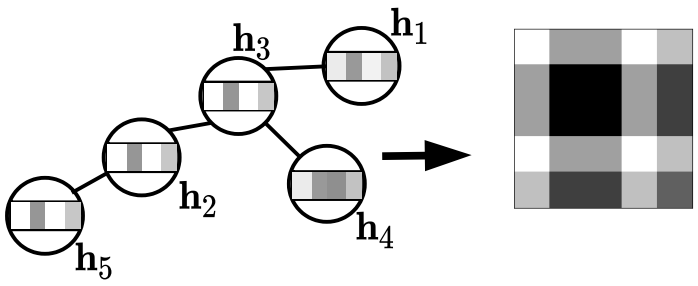
Proof. Since $\mathbf{S1} = \mathbf{1}$ and $s_{ij} \geq 0$, by the *Gershgorin circle theorem*, the spectral radius of $\mathbf{S}$ is $\rho(\mathbf{S}) = 1$. We therefore have

$$\begin{aligned} \|\mathbf{S}\mathbf{x}'_i - \mathbf{S}\mathbf{x}'_j\| &= \|\mathbf{S}(\mathbf{x}'_i - \mathbf{x}'_j)\| \\ &\leq \|\mathbf{S}\|_2\|\mathbf{x}'_i - \mathbf{x}'_j\| \\ &= \|\mathbf{x}'_i - \mathbf{x}'_j\| \end{aligned} \tag{7.9}$$

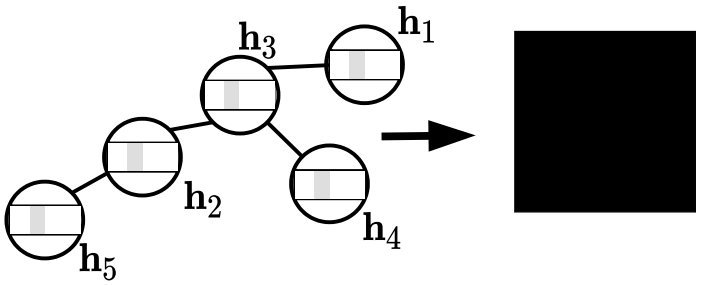
since $||\mathbf{S}||_2 = \rho(\mathbf{S})$ where $||.||_2$ is the spectral norm. The proof is the same for the rows. □



(a) No propagation.



(b) 10-hop propagation with a k-nn graph.



(c) 1000-hop propagation with a k-nn graph.

Figure 7.2: The resulting self-representation matrices using different levels of propagation over synthetic data with the LSR subspace clustering algorithm.

The grouping property also implies that in addition to the features $\mathbf{X}$, the self-representation vectors, i.e., the rows (or, by symmetry, the columns) of $\mathbf{R}$ and $\mathbf{C}$ should also be getting more similar, leading to a more meaningful partitioning when applying spectral clustering on $|\mathbf{R}|$ and $|\mathbf{C}|$ where the absolute value is applied element-wise. In figure 7.2 we show how propagating the features using a row k-nn graph generated from the data using the Euclidean distance impacts the learned self-representation matrix using LSR subspace clustering. We see how the matrix obtained after 10-hop propagation has more nonzero entries than the matrix with no propagation, which indicates, if the self-representation matrix is seen as a graph, that there are more adjacent nodes. The difficulty is identifying the appropriate propagation order, since large values can cause over-smoothing. As we can see from figure 7.2, the self-representation matrix after 1000-hop propagation is entirely uniform, since node features have converged to uninformative representations.

Proposition 7. *Given a row-normalized adjacency matrix with added self-loops* $\mathbf{S}$*,* $\lim_{p\to\infty}\mathbf{S}^p$ *exists and its rank is the same as that of the number of connected components of* $\mathbf{A}$*.*

Proof. Since $\mathbf{S}$ is row-stochastic, we have that $\rho(\mathbf{S}) = 1$. Furthermore, since it has added self-loops it cannot be the adjacency matrix of a bipartite graph, and consequently $-1 \notin \sigma(\mathbf{S})$ the spectrum of $\mathbf{S}$.

$$\begin{aligned}\lim_{p\to\infty}\mathbf{S}^p &= \lim_{p\to\infty}\mathbf{U}\boldsymbol{\Sigma}^p\mathbf{U}^{-1}\\ &= \lim_{p\to\infty}\mathbf{U}\texttt{diag}([1^p,\dots,1^p,r_1^p,...,r_{n-c}^p])\mathbf{U}^{-1}\\ &= \mathbf{U}\texttt{diag}([1,\dots,1,0...,0])\mathbf{U}^{-1}\end{aligned} \tag{7.10}$$

It follows that $\texttt{rk}(\lim_{p\to\infty}\mathbf{S}^p) = c$, where $\forall i\, r_i \in \sigma(\mathbf{S})$, $|r_i| < 1$ and c is the number of connected components. □

Regarding the bilateral graph convolution of rows and columns of order p and q respectively, we define the generic self-expressive subspace co-clustering with bilateral graph convolution problem as

$$\begin{aligned}\min_{\mathbf{R},\mathbf{C}} \quad & \|\,\mathbf{S}_\mathbf{R}^p\mathbf{X}\mathbf{S}_\mathbf{C}^q - \mathbf{R}\,(\mathbf{S}_\mathbf{R}^p\mathbf{X}\mathbf{S}_\mathbf{C}^q)\,\mathbf{C}\,\|^2 + \Omega(\mathbf{R},\mathbf{C})\\ & \text{such that} \quad \mathbf{R}\in\mathcal{R},\ \mathbf{C}\in\mathcal{C}.\end{aligned} \tag{7.11}$$

In what follows, we will refer to the row- and column-smoothed matrix as

$$\mathbf{H} = \mathbf{S}_\mathbf{R}^p\mathbf{X}\mathbf{S}_\mathbf{C}^q$$

since this operation can be considered as a sort of preprocessing step, independent of the co-clustering model that we are about to introduce.

4.3. Subspace Co-clustering through LSR

We propose an initial variant based on the LSR subspace clustering model, where the regularization term is defined as follows: $\Omega(\mathbf{R}, \mathbf{C}) = \lambda_{\mathbf{R}}\|\mathbf{R}\|^2 + \lambda_{\mathbf{C}}\|\mathbf{C}\|^2$, where $\lambda_{\mathbf{R}}$ and $\lambda_{\mathbf{C}}$ are parameters that regulate the trade-off between the reconstruction term and the regularizer. We formulate the LSR subspace co-clustering problem as

$$\min_{\mathbf{R},\mathbf{C}} \quad \|\mathbf{H} - \mathbf{RHC}\|^2 + \lambda_{\mathbf{R}}\|\mathbf{R}\|^2 + \lambda_{\mathbf{C}}\|\mathbf{C}\|^2. \tag{7.12}$$

Fixing $\mathbf{R}$ and solving for $\mathbf{C}$ and inversely, a closed form solution can be obtained for both matrices, providing a clear illustration of how our model uses information from the columns for the row space partitioning, and vice versa

$$\begin{aligned} \mathbf{R} &= \mathbf{HC}^\top\mathbf{H}^\top\left(\mathbf{HCC}^\top\mathbf{H}^\top + \lambda_{\mathbf{R}}\mathbf{I}\right)^{-1} \\ \mathbf{C} &= \left(\mathbf{H}^\top\mathbf{R}^\top\mathbf{RH} + \lambda_{\mathbf{C}}\mathbf{I}\right)^{-1}\mathbf{H}^\top\mathbf{R}^\top\mathbf{H}. \end{aligned} \tag{7.13}$$

However, solving the problem requires an iterative process where, in alternation, one of $\mathbf{R}$ and $\mathbf{C}$ is fixed and the other updated, until convergence. The overall computational complexity is roughly in $\mathcal{O}(n^3 + d^3 + tnd^2 + tn^2d)$ due to the inversion and the necessary spectral clustering step, where t is the number of iterations, and the spatial complexity is $\mathcal{O}(n^2 + d^2)$, which is very often prohibitive for real-world applications, especially those pertaining to text. We therefore try to improve the efficiency of the solving scheme by adding further constraints, as described below.

4.4. SC^3: A More Efficient Formulation Through Orthogonality Constraints

To address the issue of complexity we introduce factor matrices $\mathbf{Z} \in \mathbb{R}^{n\times k}$ and $\mathbf{W} \in \mathbb{R}^{d\times g}$, which we constrain to be semi-orthogonal, i.e., $\mathbf{Z}^\top\mathbf{Z} = \mathbf{I}_k$ and $\mathbf{W}^\top\mathbf{W} = \mathbf{I}_g$, such that $\mathbf{R} = \mathbf{ZZ}^\top$ and $\mathbf{C} = \mathbf{WW}^\top$. With these constraints the LSR co-clustering problem becomes simpler, since $\|\mathbf{Z}\|^2 = \mathtt{rk}(\mathbf{Z})$ and $\|\mathbf{W}\|^2 = \mathtt{rk}(\mathbf{W})$ are constant. The new formulation of the problem, that we have termed SC^3, is

$$\begin{aligned} \min_{\mathbf{Z},\mathbf{W}} \quad & \|\mathbf{H} - \mathbf{ZZ}^\top\mathbf{HWW}^\top\| \\ \text{such that} \quad & \mathbf{Z}^\top\mathbf{Z} = \mathbf{I}_k \quad \mathbf{W}^\top\mathbf{W} = \mathbf{I}_g \end{aligned} \tag{7.14}$$

At first glance this problem also requires an alternating solving scheme using two update rules that we obtain by fixing $\mathbf{W}$ and solving for $\mathbf{Z}$, and vice versa:

$$\begin{aligned} \mathbf{Z} = \mathbf{U} \quad \text{such that} \quad [\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \mathtt{TruncatedSVD}(\mathbf{HW}, k) \\ \mathbf{W} = \mathbf{U} \quad \text{such that} \quad [\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \mathtt{TruncatedSVD}(\mathbf{H}^\top\mathbf{Z}, g). \end{aligned}$$

This entails that $g = k$, making this a block clustering problem reminiscent of the block seriation co-clustering model. The detailed pseudo-code for this method is given in algorithm

6, whose spatial complexity is the same as for LSR, but the computational complexity is in $\mathcal{O}(n^3 + d^3 + tndk)$. Although this method is faster, it remains inefficient owing to bottlenecks, to the iterative nature of the algorithm and, more importantly, because of the spectral clustering step.

Algorithm 6: Naive $\texttt{SC}^3$

Input : $\mathbf{X}$ feature matrix, $\mathbf{S_R}$ row propagation matrix, $\mathbf{S_C}$ column propagation matrix, k number of co-clusters, p, q row and column propagation orders, ϵ tolerance.

Output: $\mathbf{Z}$, $\mathbf{W}$ row and column factors, $\pi_{\mathbf{R}}$, $\pi_{\mathbf{C}}$ row and column partitions.

$\mathbf{H} \leftarrow \mathbf{S}_{\mathbf{R}}^{p}\mathbf{X}\mathbf{S}_{\mathbf{C}}^{q}$;

$[\_, \_, \mathbf{V}] = \texttt{TruncatedSVD}\,(\mathbf{H}, k)$;

$\mathbf{W}^{(0)} \leftarrow \mathbf{V}$;

while $\|\mathbf{Z}^{(k)} - \mathbf{Z}^{(k-1)}\| + \|\mathbf{W}^{(k)} - \mathbf{W}^{(k-1)}\| > \epsilon$ **do**

  $[\mathbf{Z}^{(k)}, \_, \_] = \texttt{TruncatedSVD}(\mathbf{H}\mathbf{W}^{(k-1)}, k)$;

  $[\mathbf{W}^{(k)}, \_, \_] = \texttt{TruncatedSVD}(\mathbf{H}^{\top}\mathbf{Z}^{(k)}, k)$;

end

$\mathbf{R}, \mathbf{C} \leftarrow \mathbf{Z}\mathbf{Z}^{\top}, \mathbf{W}\mathbf{W}^{\top}$;

$\pi_{\mathbf{R}} \leftarrow \texttt{spectral\_clustering}(|\mathbf{R}|)$;

$\pi_{\mathbf{C}} \leftarrow \texttt{spectral\_clustering}(|\mathbf{C}|)$;

Optionally deduce a block diagonal bicluster matrix from $\pi_{\mathbf{Z}}$ and $\pi_{\mathbf{W}}$;

4.4.1 Efficiently Solving for $\mathbf{Z}^*$ and $\mathbf{W}^*$

The problem stated above can be solved efficiently using a single truncated SVD. This is a consequence of the following proposition:

Proposition 8. *The alternating process defined in system of equations 4.4 converges to* $\mathbf{Z}$ *and* $\mathbf{W}$ *containing the* k *left and right singular vectors of* $\mathbf{H}$ *respectively corresponding to the largest* k *singular values.*

Proof. Suppose, without loss of generality, that $k \leq \texttt{rk}(\mathbf{H})$. We have that $k = \texttt{rk}(\mathbf{Z}) = \texttt{rk}(\mathbf{W})$, implying that

$$\begin{aligned} \texttt{rk}\left(\mathbf{Z}\mathbf{Z}^{\top}\mathbf{H}\mathbf{W}\mathbf{W}^{\top}\right) &\leq \min\{\texttt{rk}(\mathbf{Z}), \texttt{rk}(\mathbf{W}), \texttt{rk}(\mathbf{H})\} \\ &= k. \end{aligned}$$

This means that we are looking for the best k-rank approximation of $\mathbf{H}$ for the Frobenius norm. Given $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \texttt{SVD}(\mathbf{H})$, and setting $\mathbf{Z} = \mathbf{U}_k = [\mathbf{u}'_1, \ldots, \mathbf{u}'_k]$ and $\mathbf{W} = \mathbf{V}_k =$

$[\mathbf{v}'_1, \ldots, \mathbf{v}'_k]$, we have

$$\begin{aligned} \|\mathbf{H} - \mathbf{Z}\mathbf{Z}^\top \mathbf{H}\mathbf{W}\mathbf{W}^\top\|^2 &= \|\mathbf{H} - \mathbf{U}_k\mathbf{U}_k^\top \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \mathbf{V}_k\mathbf{V}_k^\top\|^2 \\ &= \|\mathbf{H} - \mathbf{U}_k[\mathbf{I}_k, \mathbf{0}]\mathbf{\Sigma}[\mathbf{I}_k, \mathbf{0}]^\top \mathbf{V}_k^\top\|^2 \\ &= \|\mathbf{H} - [\mathbf{U}_k, \mathbf{0}]\mathbf{\Sigma}[\mathbf{V}_k, \mathbf{0}]^\top\|^2 \\ &= \|\mathbf{H} - \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top\|^2. \end{aligned} \tag{7.15}$$

From the Eckart–Young–Mirsky theorem we have that $\tilde{\mathbf{H}} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top$ is the best rank-k approximation of $\mathbf{H}$. □

This leads to a more efficient algorithm because the iterative step is circumvented. The interplay between the rows and columns is still implicitly present, however, since this solution is also the analytic solution to the alternating optimization problem referred to above, in which the row-column interaction is explicit.

The above result enables us to show that our approach has a grouping effect.

Proposition 9. *Given matrix* $\mathbf{H}$*, the solutions* $\mathbf{R}$ *and* $\mathbf{C}$ *in SC*^{3} *display a grouping effect on the rows and columns respectively of matrix* $\mathbf{H}$*.*

Proof. To this end, we show that $\mathbf{R}$ and $\mathbf{C}$ display a grouping effect on $\mathbf{H}$. We give the proof for $\mathbf{R}$ only since it is similar for $\mathbf{C}$. Let the full singular value decomposition of $\mathbf{H}$ be $\mathbf{H} = \mathbf{U}\mathbf{\Sigma}_k\mathbf{V}^\top$. We have that

$$\mathbf{Z} = \begin{pmatrix} \mathbf{I}_k & 0 \\ 0 & 0 \end{pmatrix} \mathbf{U}$$

and that $\mathbf{u}_i = \mathbf{\Sigma}^{-1}\mathbf{V}^\top \mathbf{h}_i$. Given these two equations, it is possible to write

$$\begin{aligned} \|\mathbf{z}_i - \mathbf{z}_j\| &= \left\| \begin{pmatrix} \mathbf{I}_k & 0 \\ 0 & 0 \end{pmatrix} \mathbf{u}_i - \begin{pmatrix} \mathbf{I}_k & 0 \\ 0 & 0 \end{pmatrix} \mathbf{u}_j \right\| \\ &= \left\| \begin{pmatrix} \mathbf{I}_k & 0 \\ 0 & 0 \end{pmatrix} \mathbf{\Sigma}^{-1}\mathbf{V}^\top \mathbf{h}_i - \begin{pmatrix} \mathbf{I}_k & 0 \\ 0 & 0 \end{pmatrix} \mathbf{\Sigma}^{-1}\mathbf{V}^\top \mathbf{h}_j \right\| \\ &\leq \| \mathbf{h}_i - \mathbf{h}_j \| \left\| \begin{pmatrix} \mathbf{I}_k & 0 \\ 0 & 0 \end{pmatrix} \mathbf{\Sigma}^{-1}\mathbf{V}^\top \right\| \\ &= \| \mathbf{h}_i - \mathbf{h}_j \| \, \text{const.} \end{aligned} \tag{7.16}$$

We also have that since $\mathbf{R} = \mathbf{Z}\mathbf{Z}^\top$, then $\mathbf{r}_i = \mathbf{Z}\mathbf{z}_i$. Therefore, it holds that

$$\begin{aligned} \|\mathbf{r}_i - \mathbf{r}_j\| &= \| \mathbf{Z}(\mathbf{z}_i - \mathbf{z}_j)\| && (7.17) \\ &= \| \mathbf{z}_i - \mathbf{z}_j \| . && \text{since } \mathbf{Z}^\top\mathbf{Z} = \mathbf{I} \end{aligned}$$

From equations 7.16 and 7.17, we have

$$\forall i, j \quad \|\mathbf{h}_i - \mathbf{h}_j\| \to 0 \implies \|\mathbf{r}_i - \mathbf{r}_j\| \to 0$$

implying that there is a grouping effect on the rows of $\mathbf{H}$ for $\mathbf{R}$. □

This gives us an efficient way to obtain $\mathbf{Z}^*$ and $\mathbf{W}^*$, together with theoretical guarantees regarding the quality of these solutions. The main remaining complexity bottleneck is the spectral clustering step with its cubic computational complexity and its quartic space complexity in the number of rows and columns. However, using the structure of our $\mathbf{R}$ and $\mathbf{C}$ we may obtain a spectral clustering algorithm.

4.5. Efficient Spectral Clustering of the Kernel Self Representation Matrices

4.5.1 Nonnegative feature map

The optimal self-representation matrix $\mathbf{R}^* = \mathbf{Z}^*\mathbf{Z}^{*\top}$ is symmetric by construction, but its entries are not necessarily nonnegative. An element-wise absolute value would therefore be required in order for us to obtain a valid affinity matrix. However, this would remove all the information already held on the decomposition of $\mathbf{R}^*$ into $\mathbf{Z}^*\mathbf{Z}^{*\top}$, since generally there is no relation between the spectrum of a matrix and its spectrum after applying an entry-wise function. We circumvent this problem by instead considering an affinity matrix constructed through some nonnegative kernel, i.e.,

$$\mathbf{K}_\mathbf{R} = \langle \varphi(\mathbf{Z}^*), \varphi(\mathbf{Z}^*) \rangle \quad \text{such that} \quad k_{ij} \geq 0$$

where φ is the feature map of the kernel applied row-wise that we need to calculate explicitly. Two possible approaches are available:

Exact feature maps. The kernel trick can provide a means of operating in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space. Computing explicit feature maps is challenging for most commonly used kernel functions, which tend to increase the dimensionality of the original inputs to such an extent that it becomes impossible to use them in realistic scenarios. For instance, in the case of the second degree polynomial kernel $k(\mathbf{z}_i, \mathbf{z}_j) = (\mathbf{z}_i^\top \mathbf{z}_j + c)^2$ where c is some constant that we can see as a bias term. For example, when $c = 1$ then the feature map is

$$\begin{aligned}\varphi\colon \mathbb{R}^k &\to \mathbb{R}^{\binom{k+2}{2}} \\ \mathbf{z} &\mapsto \langle z_k^2, \ldots, z_1^2, \sqrt{2}z_k z_{k-1}, \ldots, \sqrt{2}z_k z_1, \\ &\quad \sqrt{2}z_{k-1}z_{k-2}, \ldots, \sqrt{2}z_{k-1}z_1, \ldots, \sqrt{2}z_2 z_1, \\ &\quad \sqrt{2}z_k, \ldots, \sqrt{2}z_1, 1\rangle\end{aligned}$$

More generally, for a polynomial kernel of degree d the feature map is a function $\varphi : \mathbb{R}^k \mapsto \mathbb{R}^{\binom{k+d}{k}}$. The other possibility is that the explicit feature map is infinite-dimensional and thus impossible to compute, such as in the case of the Radial Basis Function (RBF) kernel.

Approximate feature maps A number of methods using approximations of the feature maps of the desired kernel have been proposed over the years. These include the Nyström method [Drineas, 2005], Tensor Sketch [Pham, 2013], and the use of random features [Rahimi, 2007].

4.5.2 Efficient Spectral Clustering

Since the eigenvectors of $\mathbf{K_R}$ are the same as the left singular vectors of $\varphi(\mathbf{Z}^*)$, the process of spectral clustering becomes much faster, since the affinity matrix does not need to be computed explicitly. For our purposes we adapt the spectral algorithm proposed in [Ng, 2001] as follows:

1. We efficiently compute the diagonal matrix $\mathbf{D}$ containing the sums of the rows of $\mathbf{K_R}$.
2. We construct matrix $\hat{\mathbf{Z}} = \mathbf{D}^{-1/2}\varphi(\mathbf{Z})$, on which we do an SVD yielding the eigenvectors of $\mathbf{D}^{-1/2}\mathbf{K_R}\mathbf{D}^{-1/2}$, referred to as $\mathbf{U}$.
3. We obtain the final assignments of the rows of $\mathbf{H}$ according to the assignment of vectors $[\mathbf{u}_1, ...\mathbf{u}_n]^\top$ using the k-means algorithm.

This is equivalent to doing the spectral clustering on the normalized Laplacian of $\mathbf{K_R}$ i.e. $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{K_R}\mathbf{D}^{-1/2}$ rendering the complexity of the spectral clustering linear in the number of samples instead of cubic. To obtain a spectral clustering of the affinity matrix of $\mathbf{C}^*$, the operations are the same.

5. Algorithm and Complexity

The pseudo-code for the efficient version of our algorithm is given in algorithm 7. We now discuss the computational complexity in detail.

Feature Propagation Step Assuming that the propagation matrix is sparse, the complexity of this operation is in $\mathcal{O}(p\|\mathbf{S_R}\|_0 + q\|\mathbf{S_C}\|_0)$, where $\|.\|_0$ is the 0-norm that gives the number of non-zero entries of its input.

Truncated SVD on H The computational complexity of this step using randomized SVD [Halko, 2011] is $\mathcal{O}(nd\log(k))$.

Truncated SVD on $\varphi(\mathbf{Z})$ and $\varphi(\mathbf{W})$ This depends on the dimensionality of the feature map. Since we consider the affine kernel feature map as the main choice, the complexity using a randomized SVD on it is $\mathcal{O}(nk\log(k) + dk\log(k))$.

Efficient spectral clustering. This operation has a complexity in $\mathcal{O}(tnk^2 + tdk^2)$, where t is the number of iterations of k-means.

Overall computational complexity The overall computational complexity of the proposed SC^3 algorithm is thus in $\mathcal{O}\left(p\|\mathbf{S_R}\|_0 + q\|\mathbf{S_C}\|_0 + nd\log(k) + tnk^2 + tdk^2\right)$.

Algorithm 7: Efficient $\texttt{SC}^3$

Input : $\mathbf{X}$ feature matrix, $\mathbf{S_R}$ row propagation matrix, $\mathbf{S_C}$ column propagation matrix, k number of co-clusters, p, q row and column propagation orders.

Output: $\mathbf{Z}$, $\mathbf{W}$ row and column factors, $\pi_{\mathbf{R}}$, $\pi_{\mathbf{C}}$ row and column partitions.

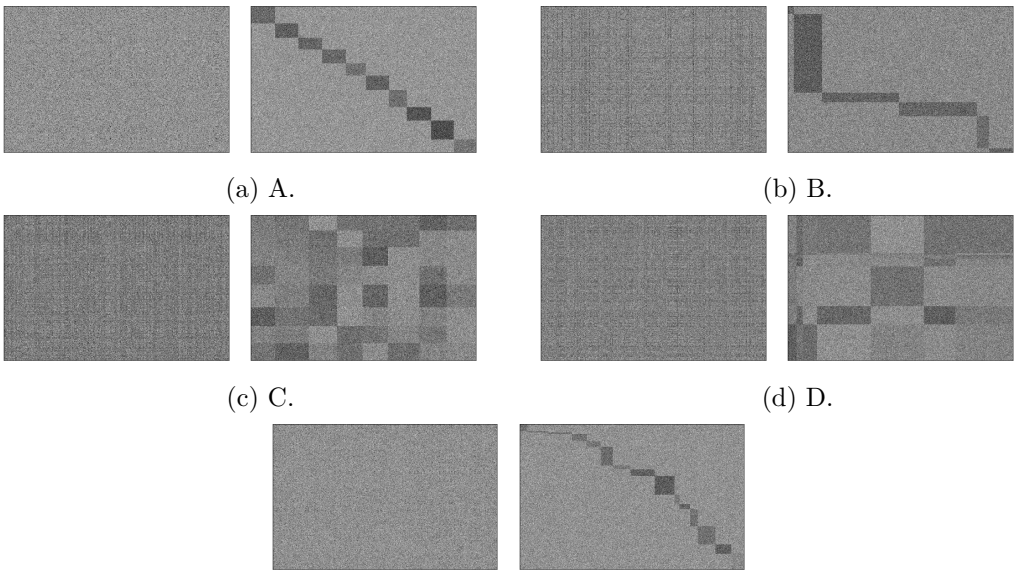
$\mathbf{H} \leftarrow \mathbf{S}_{\mathbf{R}}^{p}\mathbf{X}\mathbf{S}_{\mathbf{C}}^{q}$;
$[\mathbf{Z}, \_, \mathbf{W}] = \texttt{TruncatedSVD}\,(\mathbf{H}, k)$;
$\mathbf{M}_{\varphi} \leftarrow \varphi(\mathbf{Z})$;
$\mathbf{D} \leftarrow \texttt{diag}(\mathbf{Z}_{\varphi}(\mathbf{Z}_{\varphi}^{\top}\mathbf{1}))$;
$\hat{\mathbf{Z}}_{\varphi} \leftarrow \mathbf{D}^{-\frac{1}{2}}\mathbf{Z}_{\varphi}$;
$\mathbf{U} \leftarrow \texttt{TruncatedSVD}\left(\hat{\mathbf{Z}}_{\varphi}, k\right)$;
$\pi_{\mathbf{Z}} \leftarrow k\texttt{-means}(\mathbf{U})$;
$\mathbf{M}_{\varphi} \leftarrow \varphi(\mathbf{W})$;
$\mathbf{D} \leftarrow \texttt{diag}(\mathbf{W}_{\varphi}(\mathbf{W}_{\varphi}^{\top}\mathbf{1}))$;
$\hat{\mathbf{W}}_{\varphi} \leftarrow \mathbf{D}^{-\frac{1}{2}}\mathbf{W}_{\varphi}$;
$\mathbf{U} \leftarrow \texttt{TruncatedSVD}\left(\hat{\mathbf{W}}_{\varphi}, k\right)$;
Discard the column of $\mathbf{U}$ corresponding to the largest singular value;
$\pi_{\mathbf{W}} \leftarrow k\texttt{-means}(\mathbf{U})$;
Optionally deduce a block diagonal bicluster matrix from $\pi_{\mathbf{Z}}$ and $\pi_{\mathbf{W}}$;

Overall spatial complexity The space taken by the created matrices is in $\mathcal{O}\,(nk + dk)$.

As seen in Algorithm 7, k-means is used to propose a hard clustering. Instead of k-means the user can perform any other clustering method including fuzzy clustering methods such as soft k-means. This implies that each data point can belong to more than one cluster; for each of them a set of coefficients gives the degree of being in each cluster. It is also the case of the EM algorithm [Dempster, 1977] where in E-step posterior probabilities of each data point are computed.

6. Experiments

In this section we present the experimental setup and results. We consider two tasks which are co-clustering and document clustering. We start by introducing the models used for comparison and the overall experimental settings. Then, for each task we present the datasets and evaluation metrics as well as the results that were obtained. We follow with some comparisons between the different possible nonnegative kernels, and finally present the results for neighborhood propagation when a k-nn graph is used instead of the ground truth graph.



(a) A. (b) B.

(c) C. (d) D.

(e) E.

Figure 7.3: Synthetic datasets before and after rearrangement with respect to the true partitions.

6.1. Experimental Setup

6.1.1 Baselines

We compare our model against clustering and co-clustering models that either use only the input node feature matrix $\mathbf{X}$ or that use both $\mathbf{A}$ and $\mathbf{X}$, that is to say *attributed* graph clustering/co-clustering models.

Clustering models :

- *Vanilla models*: We take k-means as the simplest baseline for our comparison.
- *Subspace Clustering Models*: We compare our model against the subspace clustering models previously mentioned: LSR, EnSC, ℓ^0-SCC and Noisy-DR-ℓ^0-SCC-LR, SSC-OMP and the mini-batch version of K-FSC.
- *Attributed graph clustering models*: We use the GIC, S^{2}GC and GCC models.

Co-clustering models :

- *Vanilla models*: We compare our model against the spectral co-clustering algorithm (SpecCo), DCC and RDPCo.
- *Attributed graph co-clustering Models*: The only existing model of this kind is CFOND.

6.1.2 Experimental Settings

For our method, we normalize the word count datasets using tf-idf and unit-normalize across the rows. We use the official implementation (with the recommended or default param-

Algorithm 8: Propagation Order Selection Rule

Input : $\mathbf{X}$ document-term matrix, $\mathbf{S_R}$ row propagation matrix, $\mathbf{S_C}$ column propagation matrix, q column propagation order, k number of co-clusters.
Output: Propagation order p^*.

$p \leftarrow 1$;
while $p*$ *not found* **do**
  $\mathbf{Z}, \mathbf{W}, \_, \_ \leftarrow \texttt{SC}^3(\mathbf{X}, \mathbf{S_R}, \mathbf{S_C}, p, q, k)$;
  $\text{loss}_p \leftarrow \left\| \mathbf{S}_\mathbf{R}^p \mathbf{X} - \mathbf{Z}\mathbf{Z}^\top \mathbf{S}_\mathbf{R}^p \mathbf{X}\mathbf{W}\mathbf{W}^\top \right\|$;
  if $| loss_p - loss_{p-1} | < \frac{d}{n\lceil\sqrt{k}\rceil}$ ***or*** $p = 100$ **then**
    $p* \leftarrow p$
  end
  $p \leftarrow p + 1$
end

eters) for each of them, apart from CFOND ($\rho = \beta = \alpha = 1$), LSR ($\lambda = 1$), and RDPCo (with S_r and S_c), ℓ^0-SCC and Noisy-DR-ℓ^0-SCC-LR (see [Yang, 2018b; Yang, 2022] for parameter setting) which, to the best of our knowledge, have not been made publicly available. We also use the recommended parameters for each dataset whenever possible. For models that use a parameter p like ours, we run their selection rule until convergence with a maximum p of 100 for fairness. All models were run on the same machine with 12GB memory GPU and a RAM of 12GB. All experiments for the different models were run on the same machine with 12GB of RAM and a 2.3Ghz Xeon Processor. We perform 20 runs for each model.

6.1.3 Choice of Propagation Matrices

In our model, we use a k-nn graph generated from the rows of matrix $\mathbf{X}$ using the euclidean distance with $k = 3$. We use a propagation order of ten, $p = 10$. For the columns, we do not use a k-nn graph, but rather a graph based on NNPMI, which is a nonnegative version of Pointwise Mutual Information (PMI) traditionally used to quantify word-relatedness. The column propagation order is set to $q = 1$. The column NNPMI graph is defined as

$$\mathbf{S_C} = (s_{\mathbf{C}ij}) = \left(\max \left\{ \log \left(\frac{y_{..}}{y_{i.} y_{.j}} y_{ij} \right), 0 \right\}_{ij} \right) \tag{7.18}$$

where $y_{ij} = \mathbf{x}_i^\top \mathbf{x}_j$. We use the nonnegative version so that the resulting matrix after propagation can still be seen as a document-term matrix. Both matrices are then normalized using the normalization proposed in [Fettal, 2022c].

6.2. Co-clustering

6.2.1 Synthetic Datasets

Due to the absence of datasets with labels along both rows and columns. We propose to use synthetic datasets for the evaluation of the co-clustering performance of our model. The

five datasets are depicted in figure 7.3 before and after rearrangement with respect to the ground truth partition. The characteristics of these datasets are available in table 7.1. These datasets have one of two possible types of structures, checkerboard [Kluger, 2003] and block diagonal [Dhillon, 2001].

Table 7.1: Synthetic datasets' characteristics.

Dataset	Rows	Columns	Biclusters	Proportions	Structure
A	500	500	10	equal	Block diagonal
B	800	1000	6	unequal	Block diagonal
C	800	800	8	equal	Checkerboard
D	2000	1200	7	unequal	Checkerboard
E	2500	2500	15	unequal	Checkerboard

6.2.2 Evaluation Metrics

To compare our algorithms we rely on the *clustering accuracy* that is computed by rearranging the rows of the confusion matrix so as to obtain the greatest possible trace using the Hungarian method. Clustering accuracy then corresponds to this trace divided by the number of elements. However, here we use a metric that simultaneously quantifies the clustering accuracy over rows and columns which we call co-clustering accuracy. Given $c(\pi_r)$, the accuracy over the rows; and $c(\pi_c)$, the accuracy over the columns. The co-clustering accuracy *cca* [Govaert, 2008] is given by

$$cca(\pi_r, \pi_c) = c(\pi_r) + c(\pi_r) - c(\pi_r) \times c(\pi_r)$$

Table 7.2: Co-clustering performance on the synthetic datasets averaged over 20 runs of the different co-clustering models. Our model finds the ground truth partition in almost all cases and has the best performance (or is tied for best) over all datasets.

Dataset	A	B	C	D	E
SpecCo	89.53±1.39	99.54±0.0	94.32±0.0	99.95±0.02	78.43±1.5
DCC	98.61±1.0	99.96±0.1	45.0±15.0	72.06±14.1	98.87±1.0
RDpCo	85.1±6.3	86.74±6.7	28.18±0.0	64.19±14.9	22.5±0.0
CFOND	**100.0±0.0**	97.9±2.0	**100.0±0.0**	99.25±0.7	98.16±0.8
SC^3	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**	**99.87±0.2**

6.2.3 Performance

Table 7.2 shows the co-clustering results of our algorithm with respect to other co-clustering algorithms. We see that the models that use a topological information generally outperform the other models. Our model finds the ground truth partition in four out of five cases and has the best performance (or is tied for best) over the five datasets.

6.3. Document Clustering

6.3.1 Datasets

We use four attributed graph citation networks, which are graphs characterized by an adjacency matrix $\mathbf{A}$ and a node document-term matrix $\mathbf{X}$. The summary statistics are given in table 7.3. The nodes in ACM and Citeseer correspond to word count vectors, and those in PubMed and Wiki to tf-idf weighted word vectors. We use the ACM dataset, whose graph is not very informative, to compare the robustness of the models that use this information.

Table 7.3: Document datasets' statistics.

Dataset	#Nodes	#Edges	#Features	#Classes
ACM [Wang, 2019]	3025	9150593	1870	3
CiteSeer [Sen, 2008]	3327	4732	3703	6
PubMed [Sen, 2008]	19717	44338	500	3
Wiki [Yang, 2015]	2405	17981	4973	17
Computers [Shchur, 2018]	13381	259159	767	10

6.3.2 Evaluation Metrics

To assess the performance of different algorithms on the document clustering task, we use three commonly used clustering performance metrics: clustering accuracy (Acc), Adjusted Rand Index (ARI) [Hubert, 1985], and normalized mutual information (NMI) [Cai, 2008].

6.3.3 Choice of Propagation Matrices

Here, we set the row graph to be the adjacency matrix provided in each attributed graph dataset $\mathbf{S_R} = \mathbf{A}$, but later we will test our model with a k-nn graph generated from the rows of matrix $\mathbf{X}$. The row propagation order p is selected using the selection rule given in algorithm 8. The column propagation matrix and normalizations used are the same as for the synthetic data.

6.3.4 Performance

We consider three versions for our method corresponding to three different nonnegative kernel functions, linear, quadratic and radial basis (rbf) functions. Note that the linear kernel does not really result in a proper affinity matrix but it has a similar behaviour to the other valid kernels when using a bias term, so it can be a more efficient surrogate for them. Tables 7.4,7.5 show the row/document clustering performance of the different models. We report results for the power selected using rule 8. Overall, we see that methods using both graph structure and features outperform methods using features only; this is the case for all datasets, with the exception of ACM, where the graph is nearly strongly connected. We included the ACM dataset to show the robustness of our model in the face of an uninformative

Table 7.4: Clustering results on ACM, Citeseer and Wiki.

Method	Input	Co-clustering	Subspace clustering	ACM			CiteSeer			Wiki		
				Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
k-means	$\mathbf{X}$	✗	✗	62.8±4.8	37.2±9.2	34.5±10.4	62.5±1.6	36.7±1.9	35.5±2.5	47.3±6.0	46.3±6.9	26.4±8.1
LSR	$\mathbf{X}$	✗	✓	80.3±0.0	47.0±0.0	51.9±0.0	21.1±0.0	0.2±0.1	0.0±0.0	21.1±3.3	9.0±5.9	2.6±2.0
EnSC	$\mathbf{X}$	✗	✓	79.5±0.0	46.8±0.0	50.3±0.0	55.6±0.0	14.8±0.0	14.6±0.0	45.5±2.0	45.7±1.7	28.8±1.3
SSC-OMP	$\mathbf{X}$	✗	✓	78.8±0.1	43.4±0.1	48.3±0.1	24.0±1.1	3.5±0.4	1.8±0.1	52.7±4.4	48.1±2.3	33.3±1.5
ℓ^0-SSC	$\mathbf{X}$	✗	✓	80.7±0.0	48.4±0.0	52.8±0.0	55.4±0.0	26.3±0.0	24.5±0.0	45.1±0.0	43.8±0.0	25.6±0.0
DR-ℓ^0SCC-LR	$\mathbf{X}$	✗	✓	75.8±2.1	42.4±2.7	41.5±4.0	58.1±0.6	27.1±0.3	26.2±0.6	45.8±0.1	45.2±0.4	25.3±0.7
K-FSC	$\mathbf{X}$	✗	✓	56.2±7.5	17.1±6.3	18.0±5.7	35.3±6.9	12.4±3.5	10.6±4.0	38.2±5.1	35.6±3.9	17.7±4.4
SpecCo	$\mathbf{X}$	✓	✗	80.6±0.1	48.4±0.1	52.3±0.1	30.3±1.7	10.0±1.3	5.5±1.6	37.8±1.2	38.2±0.3	20.8±0.4
DCC	$\mathbf{X}$	✓	✗	40.5±3.3	7.8±5.1	2.1±2.2	35.1±3.8	11.5±2.4	8.9±2.9	48.3±3.6	47.5±2.6	30.6±3.0
RDPCo	$\mathbf{X}$	✓	✗	35.1±0.0	0.0±0.0	0.0±0.0	46.3±3.2	12.6±6.3	8.3±4.2	18.8±3.1	5.6±7.7	1.4±2.3
GIC	$\mathbf{A}, \mathbf{X}$	✗	✗	34.3±0.4	0.1±0.1	0.0±0.0	68.8±0.8	43.8±1.0	44.6±1.0	46.5±1.4	48.2±0.5	30.2±1.4
SGC	$\mathbf{A}, \mathbf{X}$	✗	✗	83.7±0.0	55.7±0.0	58.8±0.0	64.9±0.1	39.4±0.0	38.8±0.0	51.9±0.8	49.6±0.2	28.6±0.1
S^{2}GC	$\mathbf{A}, \mathbf{X}$	✗	✗	40.5±3.4	1.7±1.2	1.8±1.3	68.1±0.3	42.3±0.2	43.5±0.3	52.7±1.0	49.0±0.3	29.6±0.9
GCC	$\mathbf{A}, \mathbf{X}$	✗	✗	35.4±0.0	0.3±0.0	0.0±0.0	69.4±0.1	**45.0±0.2**	45.4±0.1	54.1±0.8	**55.0±0.2**	**33.3±0.5**
CFOND	$\mathbf{A}, \mathbf{X}$	✓	✗	71.8±0.6	37.2±0.5	38.2±0.7	63.0±1.1	36.6±1.3	36.2±1.2	47.8±3.0	49.5±2.1	30.3±2.5
$\mathbf{SC}^3_{\mathbf{linear}}$	$\mathbf{A}, \mathbf{X}$	✓	✓	**88.4±0.1**	**62.0±0.2**	**68.6±0.1**	69.3±3.8	43.7±2.7	43.9±3.8	**59.7±1.9**	53.9±1.5	31.2±3.1
$\mathbf{SC}^3_{\mathbf{quad}}$	$\mathbf{A}, \mathbf{X}$	✓	✓	**88.4±0.0**	**62.0±0.1**	**68.6±0.1**	**70.7±1.3**	44.7±1.0	**45.5±2.0**	58.2±3.2	53.4±1.8	30.5±4.2
$\mathbf{SC}^3_{\mathbf{rbf}}$	$\mathbf{A}, \mathbf{X}$	✓	✓	**88.4±0.1**	**62.0±0.2**	**68.6±0.1**	70.4±1.1	44.4±0.9	44.8±1.8	57.0±3.0	52.9±2.1	28.4±3.8

Table 7.5: Clustering results on PubMed and Amazon Computers.

Method	Input	Co-clustering	Subspace clustering	PubMed			Computers		
				Acc	NMI	ARI	Acc	NMI	ARI
k-means	$\mathbf{X}$	✗	✗	60.1±0.0	31.4±0.0	28.1±0.0	36.4±1.6	7.0±8.5	-0.2±1.4
LSR	$\mathbf{X}$	✗	✗		OOM		31.6±0.5	22.3±0.5	11.0±0.2
EnSC	$\mathbf{X}$	✗	✗	55.6±0.0	14.8±0.0	14.7±0.0	32.5±1.0	32.7±2.6	15.9±1.1
SSC-OMP	$\mathbf{X}$	✗	✗	60.4±0.0	22.3±0.0	19.4±0.0	46.6±1.3	29.1±0.3	30.0±3.1
ℓ^0-SCC	$\mathbf{X}$	✗	✗		OOM		41.2±0.0	44.5±0.0	22.0±0.0
DR-ℓ^0SCC-LR	$\mathbf{X}$	✗	✗		OOM		38.4±0.4	38.1±0.2	20.3±0.2
K-FSC	$\mathbf{X}$	✗	✓	49.5±9.8	14.8±5.9	12.1±7.1	41.1±2.5	28.9±3.2	16.5±3.5
SpecCo	$\mathbf{X}$	✓	✓	61.2±0.0	24.7±0.0	21.8±0.0	31.2±1.2	29.9±1.5	10.6±1.8
DCC	$\mathbf{X}$	✓	✓	54.3±3.6	16.5±2.9	13.4±4.1	34.3±1.9	3.0±1.5	-2.2±1.4
RDPCo	$\mathbf{X}$	✓	✓	21.4±0.2	1.0±0.5	0.2±0.1	37.1±0.0	0.0±0.0	0.0±0.0
GIC	$\mathbf{A}, \mathbf{X}$	✗	✗	64.3±0.4	26.0±0.5	23.6±0.5	46.8±2.2	47.5±0.9	31.3±3.5
SGC	$\mathbf{A}, \mathbf{X}$	✗	✗	64.7±0.0	54.3±0.0	48.5±0.0	65.5±0.0	52.2±0.0	45.7±0.0
S^{2}GC	$\mathbf{A}, \mathbf{X}$	✗	✗	70.7±0.0	32.9±0.0	33.5±0.0	58.3±0.0	54.6±0.0	40.1±0.0
GCC	$\mathbf{A}, \mathbf{X}$	✗	✗	70.8±0.0	32.3±0.0	33.2±0.0	67.6±0.0	56.0±0.0	46.5±0.0
CFOND	$\mathbf{A}, \mathbf{X}$	✓	✗	60.1±0.0	31.4±0.0	28.1±0.0	23.6±0.8	13.3±1.3	7.0±0.8
$\mathbf{SC}^3_{\mathbf{linear}}$	$\mathbf{A}, \mathbf{X}$	✓	✓	**71.1±0.0**	**33.2±0.0**	**33.9±0.0**	**71.1±0.0**	**59.0±0.0**	**50.6±0.0**
$\mathbf{SC}^3_{\mathbf{quad}}$	$\mathbf{A}, \mathbf{X}$	✓	✓	**71.1±0.0**	**33.2±0.0**	**33.9±0.0**	**71.1±0.0**	**59.0±0.0**	**50.6±0.0**
$\mathbf{SC}^3_{\mathbf{rbf}}$	$\mathbf{A}, \mathbf{X}$	✓	✓	**71.1±0.0**	**33.2±0.0**	**33.9±0.0**	**71.1±0.0**	**59.0±0.0**	**50.6±0.0**

graph structure when compared with state-of-the-art attributed graph clustering models. Our model is seen to be competitive on all five datasets, and to have the best performances on all five datasets. SC^3 has either the best or the second best performance with respect to most metrics on each dataset, while having near-zero standard deviation, which shows that the proposed approach is robust. The performance gap is most striking on Wiki, where our model is seven points ahead of the closest model in terms of accuracy. We also see that the performance of the three variants of our model is similar.

For statistical significance, we perform a Nemenyi post-hoc test [Nemenyi, 1963] with a confidence level of 95% on the ranks of each model in terms of Acc, NMI, and ARI, for each dataset and for each run (20 runs for each dataset) to compare our model with the other best performing models, namely SGC, S^2GC, GIC, GCC and CFOND. This allows us to group models with similar performances. The results are shown in figure 7.4. The best group consisting of the SC^3 variants is seen to have the best performances by a wide margin, followed by the group containing only GCC.

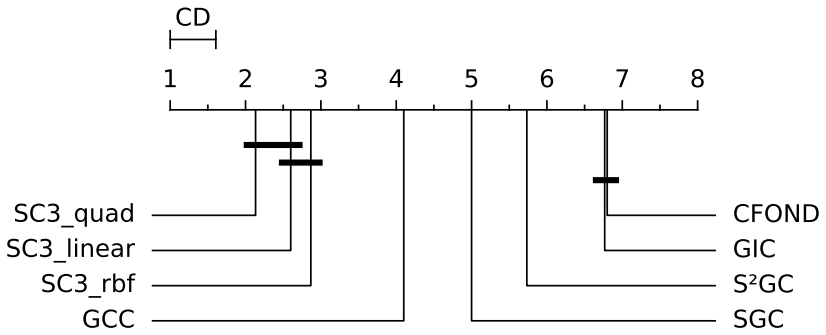
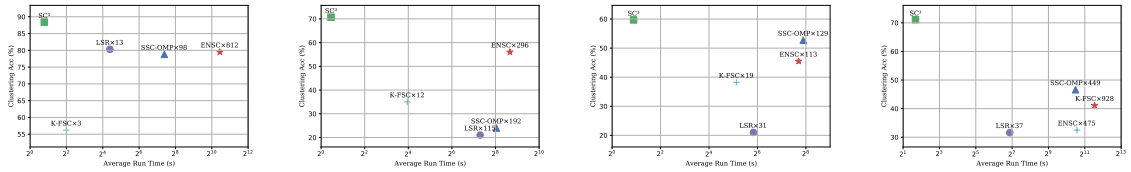


Figure 7.4: Visualization of the results of the Nemenyi test with a confidence level of 95%. We see that SC^3 variants perform similarly while being better than other models.

Compared to co-clustering methods applied on $\mathbf{X}$, note that sometimes, with certain methods we have zero values for NMI and ARI. This is explained by the fact that we are facing the recurrent problem of empty classes when dealing with sparse and unbalanced data. The problem is, however, overcome by SC^3 as illustrated for ACM.

6.3.5 Efficiency

In figure 7.5 we plot the accuracy of the subspace clustering models in relation to their training times, comparing their performance on four datasets to that of linear SC^3. The results correspond to an average over 20 runs. The time required by the propagation step is included in the overall running time of our algorithm. The fastest model on all datasets is linear SC^3, it also achieves the best accuracy score. Note that while the other models give a partition for the rows exclusively, ours gives one for both rows and columns while remaining significantly more efficient.



(a) ACM. (b) CiteSeer. (c) Wiki. (d) Computers.

Figure 7.5: Clustering accuracy plotted against training times on a logarithmic scale of subspace clustering algorithms on the different datasets. Linear SC^3 timing is used as the reference; for instance, on ACM, LSR $\times 13$ means that it is approximately 10 times slower than SC^3. Linear SC^3 consistently gives the best results and training times. PubMed is left out due to OOMs.

Table 7.7: Performance of SC^3 with a quadratic kernel using different column propagation matrices averaged over 20 runs. Best results are highlighted in bold font.

Graph	k	**ACM**			**Citeseer**			**Wiki**			**Pubmed**			**Computers**		
		Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
k-nn (euclidean)	3	88.7	62.8	69.2	68.0	42.8	43.8	52.9	50.3	19.0	**71.3**	**33.4**	**34.2**	70.5	58.7	50.3
	5	88.7	63.1	69.3	68.3	43.6	44.3	56.7	51.4	24.5	**71.2**	33.2	34.1	70.8	59.2	48.9
	10	**88.8**	**63.3**	**69.5**	**68.6**	**44.2**	**44.8**	58.4	52.7	26.4	**71.2**	**33.3**	**34.2**	**72.6**	60.5	**52.6**
k-nn (cosine)	3	88.0	61.1	67.7	67.3	42.3	42.7	54.7	51.7	24.0	**71.2**	33.0	34.0	71.3	60.2	49.7
	5	87.9	61.0	67.3	67.2	42.8	42.8	55.7	52.0	23.8	71.1	32.9	33.9	72.4	60.5	52.2
	10	87.9	60.9	67.4	67.3	42.8	43.0	55.4	51.6	25.3	71.0	32.8	33.8	72.5	**60.6**	52.4
k-nn (correlation)	3	88.6	62.7	69.0	67.3	42.4	43.1	**58.8**	**53.3**	**27.3**	**71.2**	33.1	**34.2**	**72.6**	60.5	**52.6**
	5	88.7	62.9	69.2	68.1	43.4	43.9	55.7	51.1	26.6	71.1	32.9	34.0	**72.6**	60.5	52.5
	10	88.3	62.2	68.5	68.1	43.8	44.2	57.1	52.5	26.2	71.1	32.9	34.0	72.5	60.5	52.4

Table 7.6: The three topics found by SC^3 characterized by their top ten most frequent terms, their size and coherence.

Topic	a	b	c
Most Frequent Terms	patient	cell	rat
	insulin	mice	control
	glucos	islet	activ
	type	iddm	level
	group	gene	increas
	subject	diseas	respons
	lt	develop	signific
	risk	nod	effect
	associ	children	express
	treatment	betacel	plasma
Coherence	-403.4	-365.5	-387.5
Size	280	63	157

6.4. Term Clustering

Since co-clustering models additionally generate a clustering for the terms. We use the PubMed dataset which is the only dataset for which we managed to find the actual terms.

Table 7.6 presents the most frequent terms for each topic found by our model. The PubMed dataset contains scientific works concerning diabetes. We see that topic a contains terms that are related to a presentation of diabetes, e.g, *insulin*, *glucos*, *type*, etc. Topic b has terms that are related to the microscopic effects of diabetes such as *cell*, *islet*, *gene*, *betacel*, and so on. Finally, topic c seems to concern terms that are associated with medical experimentation and analysis of results such as *control*, *increas*, *signific*, etc. We note the coherence of these term clusters since they cluster the PubMed work contents according to three topics. This term clustering can then be used to help characterize document clusters to facilitate interpretation.

In order to quantitatively evaluate the semantic coherence of topics, we use an internal metric [Mimno, 2011] to measure the level of association between terms within each cluster. Essentially, if a majority of the words within a term cluster are interconnected, it can be inferred that the cluster is semantically coherent from a statistical standpoint. Topic coherence measures the degree of semantic similarity between the top words within a topic or term cluster. It takes into account both the intrinsic coherence of the words themselves, as well as their coherence within the context of the topic or cluster as a whole. This metric has been shown to be effective in assessing the quality of topic models and identifying topics that are most representative of the underlying data. We consider the top 10 terms when computing this metric. We also report the sizes of the topics which are considered as a good metric for assessing topic quality.

6.5. Convolution Using k-nn Graphs

Unless we are working with attributed graphs, no ground truth adjacency matrix on the rows is provided, but only the feature matrix. Consequently, we need to generate an adjacency matrix on the rows ourselves. A popular choice is the k-nearest neighbors graph, based on some metric. In table 7.7 we compare results on the five datasets using k-nn graphs generated based on the Euclidean and cosine dissimilarity with a number of neighbors $k \in \{3, 5, 10\}$. We symmetrize the obtained k-nn matrix $\mathbf{A}$ as follows: $(\mathbf{A} + \mathbf{A}^\top)/2$. We see that even without using the ground truth graph, our model continues to outperform subspace clustering and co-clustering models, including CFOND, which uses the actual ground truth graph on all datasets. It also outperforms attributed graph clustering models on Wiki and ACM. On ACM, the results obtained by our model results are better when using the k-nn graph than when using the ground truth graph, since the ACM ground truth graph is not very informative. To sum up, we note that whatever the number of neighbors $k \in \{3, 5, 10\}$ and the similarity measure used, SC^3 remains profitable for data even without a ground truth adjacency matrix on the data points.

7. Conclusion

We proposed SC^3, a new approach to leverage subspace clustering for text data through co-clustering and a bilateral graph convolution. SC^3 circumvents the computational and

spatial complexity issues inherent in subspace clustering by using factor matrices and non-negative kernel feature maps. We showed that the simple model that we propose has a grouping effect, and we demonstrated how the bilateral convolution helps to put this grouping property to good use. Experiments on synthetic and real datasets showed that our model is competitive with the state of the art on the tasks of document and word clustering in the context of document-term attributed graphs, while also being efficient (linear complexity in the number of nodes) and robust in the face of uninformative graph topologies. Even when no ground truth graph is available, the bilateral convolution operation improves performance in comparison to classical subspace clustering approaches. In what comes next, we will consider multi-view attributed graphs. These are attributed graphs with multiple versions of graph structures and/or multiple graphs which describe the same nodes but from distinct perspectives.

Chapter 8

Multi-view Attributed Graph Joint Embedding and Clustering

Objective

This chapter was published as [Fettal, 2023c]. Our goal was to create an efficient yet effective approach for the simultaneous embedding and clustering of multi-view attributed-graph nodes based on the simplified graph convolutional network, as well as the reduced k-means.

Contents

1. Introduction And Related Work

Attributed graphs are graphs that contain features in their nodes. They are used to model a wide variety of structured data with applications in recommender systems [Salah, 2017b; Ying, 2018; Fan, 2019], computer vision, [Xu, 2017; Qi, 2017; Satorras, 2018], and physical systems [Sanchez-Gonzalez, 2018]. However, capturing topological (or structural) information at the same time as capturing information about node-level features presents challenges, and in the last few years a number of methods have been proposed for tackling problems such as attributed graph representation learning and attributed graph clustering.

In some real-world applications, data is collected from a variety of different sources, which means that it can be characterized using different sets of information or *views*. This is the premise of *multi-view learning* [Xu, 2013], an area of considerable interest among the data mining and machine learning communities. In the context of attributed graphs, a multi-view attributed graph is simply a set of attributed graphs; each attributed graph counts as a single view. For example, in the case of a recommender system, the relationship between users can be characterized using a two graphs, one representing their "friendship", and one representing their mutual interests; as for the features, one set of features could represent personal information and another set their past transactions.

The task of multi-view attributed graph clustering has lately received a lot of attention. The methods that have been proposed can be separated into two broad approaches. In the first approach, a consensus graph partition is learned directly from the data without explicitly learning an embedding of the graph. Methods adopting this approach include MvAGC [Lin, 2021b], where a graph filter is proposed to perform the graph clustering, and MAGC [Lin, 2021c], a similar method to MvAGC that uses a graph filter to learn a consensus graph before doing the clustering. The second approach is more flexible; it consists in learning a consensus representation or embedding before applying a simple single-view attributed graph clustering method. For example, DMGI [Park, 2020] is an unsupervised network embedding method for attributed multiplex networks that uses the concept of mutual information, while O2MAC [Fan, 2020] is based on the graph autoencoder [Kipf, 2016], it learns clustering-friendly embeddings through integrating a clustering loss in its objective.

These different methods have their shortcomings. First, the more flexible approaches that learn a consensus representation generally tackle the problems of representation learning and clustering separately, i.e. they learn representations that are not specifically tailored to clustering. Second, they often have unnecessarily complex architectures in comparison to simpler strategies. Finally, some of these methods are not generic, in the sense that they require the multi-view graph to be of a certain type: for example, a multi-view graph with a multiple structures and a single set of features (but not the other way around) like for DMGI, or a graph with exactly two views, etc. As a way of addressing these shortcomings, we propose *LMGEC* (for *Linear Multi-view Graph Embedding and Clustering*), a simple yet effective linear model. LMGEC starts by applying a linear graph filter corresponding to a one-hop neighborhood propagation step in each individual view, and then applies a weighting

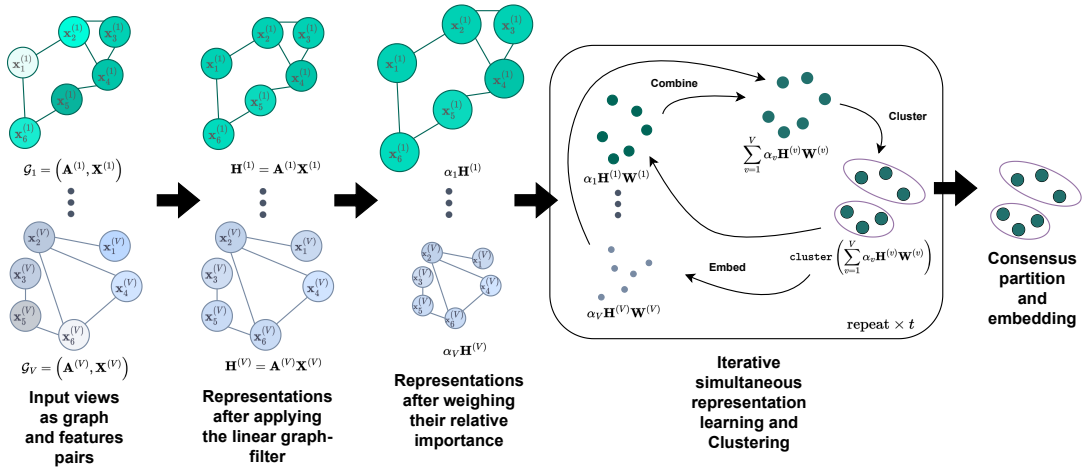


Figure 8.1: Schematic representation of LMGEC.

scheme so that views are attended to in order of their perceived importance. This is followed by an iterative process of simultaneous clustering and representation learning, which gives rise to a consensus embedding and partition of the graph. The model is generic in the sense that it can deal with any number of graph structures and/or any number of feature sets. A high-level schematic representation of the model is shown in figure 8.1. Our contributions may be summarized as follows:

- We introduce a simple yet effective generic linear model for performing multi-view attributed graph representation learning simultaneously with clustering. The model is based on (1) a one-hop neighborhood propagation corresponding to a linear graph filter, (2) a view weighting scheme reminiscent of the attention mechanism in neural networks, and (3) a graph clustering and representation learning linear component that addresses both tasks via a unified framework.

- We carry out a theoretical study of linear graph filtering, formulate the problem that we are seeking to solve, and propose an algorithm that we subject to a detailed computational complexity analysis.

- We showcase the efficiency and effectiveness of this model against the state of the art through extensive experimentation. We show that our model is both competitive and several magnitudes more efficient than current state-of-the-art multi-view attributed graph clustering.

- We release our code for reproducibility[1].

2. Preliminaries

[1] https://github.com/chakib401/LMGEC

2.1. Definitions and Notations

An attributed graph is defined as a quadruple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ where $\mathcal{V}$ represents the vertex set, $\mathcal{E}$ the edge, $\mathbf{X} \in \mathbb{R}^{n \times d}$ its node features matrix and $\mathbf{A}$ its adjacency matrix of size $n \times n$. A multi-view attributed graph is represented as a sequence of attributed graphs $\mathcal{M} := \{\mathcal{G}_v = (\mathcal{V}_v, \mathcal{E}_v, \mathbf{A}_v, \mathbf{X}_v)\}_{v=1}^{v=V}$. Matrices are denoted by boldface uppercase and vectors by boldface lowercase letters, $\mathbf{1}$ represents a column vector of ones. $\mathbf{I}$ denotes the identity matrix. Where $\mathbf{X}$ is a matrix, $\mathbf{x}_i$ is its i-th row. A matrix referenced as $\mathbf{X}_v$ means that it belongs to the v-th attributed graph, and its i-th row is referenced as $\mathbf{x}_i^v$.

2.2. Graph Filters and the Simple Graph Convolutional Network

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ be an attributed graph whose symmetrically normalized Laplacian matrix is $\mathbf{L}^{\text{sym}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ where $\mathbf{D}$ is the diagonal matrix of degrees of the graph such that $d_{ii} = \sum_j a_{ij}$. A graph signal can be seen as a vector $\mathbf{f} = [f(v_1), \ldots, f(v_n)]$ such that $f : \mathcal{V} \to \mathbb{R}$ is a real-valued function on the vertex set $\mathcal{V}$. For any graph signal $\mathbf{f}$, we can quantify its smoothness using the Laplacian quadratic form [Zhou, 2004]

$$\mathcal{S}(\mathbf{f}) = \mathbf{f}^\top \mathbf{L}^{\text{sym}} \mathbf{f} = \frac{1}{2} \sum_{i,j}^{n} a_{ij} \left(\frac{f_i}{\sqrt{d_{ii}}} - \frac{f_j}{\sqrt{d_{jj}}} \right)^2. \tag{8.1}$$

Now let $\mathbf{L}^{\text{sym}} = \mathbf{U}\Lambda\mathbf{U}^\top$ be the eigendecomposition of the Laplacian and $\{\mathbf{u}_l\}_{l=1}^n$ and $\{\lambda_l\}_{l=1}^n$ the sets of eigenvectors and eigenvalues of $\mathbf{L}^{\text{sym}}$. Since $\mathbf{L}^{\text{sym}}$ is symmetric, these eigenvectors form a basis for $\mathbb{R}^n$, and we can therefore write $\mathbf{f} = \mathbf{U}\mathbf{c} = \sum_{l=1}^n c_l \mathbf{u}_l$. This implies that we can write the Laplacian quadratic form as

$$\mathcal{S}(\mathbf{f}) = \mathbf{f}^\top \mathbf{L}^{\text{sym}} \mathbf{f} = \left(\mathbf{c}^\top \mathbf{U}^\top\right) \mathbf{U}\Lambda\mathbf{U}^\top (\mathbf{U}\mathbf{c}) = \mathbf{c}^\top \Lambda \mathbf{c} = \sum_{l=1}^{n} c_l^2 \lambda_l, \tag{8.2}$$

and that diagonal operators applied to the spectrum of the Laplacian modulate the smoothness of the signal; consequently, the eigenvalues can be seen as the frequencies of the signal [Defferrard, 2016]. Accordingly, if we wish to make a graph signal smoother, we should minimize this measure through removing frequencies that correspond to larger eigenvalues. This is done using a *low-pass filter*.

To low-pass filter a graph using a polynomial filter whose frequency-response function is g, we use the graph convolution operation which is defined as

$$\mathbf{f}_{\text{filtered}} = g\left(\mathbf{L}^{\text{sym}}\right)\mathbf{f} = \mathbf{U} g(\Lambda) \mathbf{U}^\top \mathbf{f} \tag{8.3}$$

such that $g(\Lambda) = \text{diag}\left(g(\lambda_1), ..., g(\lambda_n)\right)$. For example, in the case of a graph filter corresponding to a GCN [Kipf, 2017] with p layers, or its simplified version [Wu, 2019] using a p-th order feature propagation, its frequency-response function is given as $g(\lambda) = (1-\lambda)^p$, or in matrix form as $g(\mathbf{L}^{\text{sym}}) = (\mathbf{I} - \mathbf{L}^{\text{sym}})^p = \mathbf{A}^p$, which is a polynomial filter that is low-pass

for odd values of p and somewhat low-pass for even values of p since the filtering function is not strictly decreasing on the interval of definition of the eigenvalues $I = [0, 2]$. Note that the GCN also introduces added self-loops into the adjacency matrix $\mathbf{A}$. For more details about graph filtering and graph signal processing in general, we refer the reader to [Shuman, 2013; Ortega, 2018].

3. Proposed Model

Now that we have introduced the necessary background, we can formulate our problem.

3.1. First-order Neighborhood Propagation and Linear Graph Filtering

As previously mentioned, the graph neighborhood propagation performed in the GCN acts as a filter on the graph signal and removes high-frequency noise. We argue, however, that these steps of neighborhood propagation as performed in the GCN are unnecessary and even counterproductive because of a risk of over-smoothing, which is when the signal becomes uniform over the different nodes. To support our argument, we would point to the performance of the linear graph autoencoder [Salha, 2020], which was competitive w.r.t more complex GCN-based models. In this work we are seeking to show that a first-order (or one-hop) neighborhood propagation, when applied properly, is also sufficient for the task of simultaneous graph clustering and embedding. Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$, let

$$\tilde{\mathbf{A}} \leftarrow \mathbf{A} + \beta \mathbf{I} \tag{8.4}$$

be the adjacency matrix with β added self-loops and $\tilde{D}$ its diagonal matrix of degrees. We define our propagation matrix as

$$\mathbf{S} \leftarrow \tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}}. \tag{8.5}$$

The generalized Laplacian $\mathbf{L}^{\text{rw}}$ that corresponds to this propagation matrix is a random walk normalized Laplacian with added self-loops. The linear propagation operation we propose is

$$\mathbf{H} \leftarrow \mathbf{S}\mathbf{X} \tag{8.6}$$

where $\mathbf{H}$ are the new filtered features. The frequency-response function associated with this filter is $g(\lambda) = 1 - \lambda$ or, in matrix form, $g(\mathbf{S}) = \mathbf{I} - \mathbf{L}^{\text{rw}}$, which is clearly a linear function that is decreasing in the interval of definition of the eigenvalues $I = [0, 2]$.

3.2. Simultaneous Multi-view Attributed Graph Representation Learning and Clustering

Given a multi-view attributed graph represented as a set of attributed graphs $\mathcal{M} := \{\mathcal{G}_v\}_{v=1}^{v=V}$, we define a preliminary version of the problem of simultaneous multi-view graph

representation learning and clustering as

$$\begin{aligned} \min_{\substack{\mathbf{G},\mathbf{F}\\ \mathbf{W}_1,\ldots,\mathbf{W}_V}} \quad & \underbrace{\sum_v \underbrace{(\underbrace{\|\mathbf{H}_v - \mathbf{H}_v\mathbf{W}_v\mathbf{W}_v^\top\|^2}_{\textbf{reconstruction term}} + \underbrace{\|\mathbf{H}_v\mathbf{W}_v - \mathbf{GF}\|^2}_{\textbf{clustering term}})}_{\textbf{individual view loss}}}_{\textbf{multi-view loss}} \\ s.t. \quad & \forall v \quad \mathbf{W}_v\mathbf{W}_v^\top = \mathbf{I}, \quad \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G1} = \mathbf{1} \end{aligned} \tag{8.7}$$

where $\mathbf{W}_v \in \mathbb{R}^{d_v \times f}$ such that d_v is the dimension of the features in view v, f is the dimensionality of the consensus representation we wish to learn and k is the number of clusters. The loss corresponding to each view consists of two terms, namely a reconstruction term and a clustering term:

- **The reconstruction term** can be seen as reconstructing the filtered graph signal of the v-th view $\mathbf{H}_v$ using a semi-orthogonal matrix $\mathbf{W}_v$ similar to what is done in principal component analysis. We may draw a parallel with autoencoders, where multiplying by $\mathbf{W}$ encodes the data and $\mathbf{W}^\top$ decodes it.
- **The clustering term** is similar to the k-means objective applied to the embeddings learned from the reconstruction process $\mathbf{H}_v\mathbf{W}_v$. $\mathbf{G}$ is a partition matrix and $\mathbf{F}$ is the centroids matrix.

Matrices $\mathbf{G}$ and $\mathbf{F}$ are the same for each matrix and represent the consensus partition and centroids respectively. There are, however, exactly V $\mathbf{W}_v$ matrices, due the fact that the features in the different views do not necessarily have the same dimensionality. Problem (8.7) can be rewritten in a way that combines the two terms as follows

$$\begin{aligned} \min_{\mathbf{G},\mathbf{F},\mathbf{W}_1,\ldots,\mathbf{W}_V} \quad & \sum_v \|\mathbf{H}_v - \mathbf{GFW}_v^\top\|^2 \\ s.t. \quad & \forall v \quad \mathbf{W}_v\mathbf{W}_v^\top = \mathbf{I}, \quad \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G1} = \mathbf{1}. \end{aligned} \tag{8.8}$$

This formulation is more intuitive, since we can see it as trying to minimize the discrepancy between each input vector $\mathbf{h}_i^v$ and its corresponding centroid learned in the latent embedding space after reconstructing $\mathbf{g}_i\mathbf{FW}_v^\top$. A proof of this is available in [Yamamoto, 2014].

3.3. Paying Attention to the Individual Views

Not all views have the same importance, and for this reason it is not optimal to directly add the losses from each view without applying some kind of importance weighting scheme. To address this issue we introduce a new set of parameters $\{\alpha_v\}_{v=1}^{v=V}$ such that $\sum_v \alpha_v = 1$ where α_v represents the relative importance of each view v. With this, we obtain the final

formulation of our problem

$$\begin{aligned} \min_{\mathbf{G},\mathbf{F},\mathbf{W}_1,\ldots,\mathbf{W}_V} \quad & \sum_v \alpha_v \|\mathbf{H}_v - \mathbf{G}\mathbf{F}\mathbf{W}_v^\top\|^2 \\ s.t. \quad \forall v \quad & \mathbf{W}_v\mathbf{W}_v^\top = \mathbf{I}, \quad \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G}\mathbf{1} = \mathbf{1}. \end{aligned} \tag{8.9}$$

Note, however, that we do not introduce α_v as a parameter, since this would cause a solution to pay attention to only a single view, i.e., the view with the smallest individual view loss. With this in mind, we define α as the softmax α of the negative inertia of each view, and we add a temperature parameter for greater flexibility. The formula for each α is then

$$\alpha_v \leftarrow \frac{e^{\frac{-I_v}{\tau}}}{\sum_{w=1}^{w=V} e^{\frac{-I_w}{\tau}}} \tag{8.10}$$

where I stands for inertia. For the v-th view I_v is computed as follows: $I_v = \|\mathbf{H}_v - \mathbf{G}_v\mathbf{F}_v\|$ such that $\mathbf{W}_v$ is obtained through a truncated singular value decomposition (SVD) on $\mathbf{X}_v$, and $\mathbf{G}_v$ and $\mathbf{F}_v$ are the results of a k-means applied on the embeddings of the v-th view $\mathbf{X}_v\mathbf{W}_v$. When the temperature τ is sufficiently high, only the best view in terms of inertia is selected, and when it is sufficiently low, all the views have the same weight.

4. Optimization and Complexity

Even though solving LMGEC exactly may be NP-hard, a solution can be computed reasonably efficiently via the use of heuristics. To this end, we propose using a Block Coordinate Descent (BCD) scheme that boils down to iteratively solving sub-problems where we alternately solve for one of $\mathbf{W}_1, \ldots, \mathbf{W}_V$, $\mathbf{G}$, $\mathbf{F}$ while keeping the others fixed. All optimizations are described below.

4.1. Optimizing for G

When solving for $\mathbf{G}$ and fixing the other matrices, we obtain the following problem

$$\min_{\mathbf{G}} \quad \sum_v \alpha_v \|\mathbf{H}_v\mathbf{W}_v - \mathbf{G}\mathbf{F}\|^2 \quad s.t \quad \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G}\mathbf{1} = \mathbf{1}. \tag{8.11}$$

This problem is hard to solve, so instead we propose solving the following relaxation obtained using the Cauchy-Schwarz inequality:

$$\min_{\mathbf{G}} \quad \left\| \sum_v \alpha_v \mathbf{H}_v\mathbf{W}_v - \mathbf{G}\mathbf{F} \right\|^2 \quad s.t. \quad \mathbf{G} \in \{0,1\}^{n\times k}, \quad \mathbf{G}\mathbf{1} = \mathbf{1}. \tag{8.12}$$

In this way we can efficiently minimize the objective of this problem with the assignment step

$$g_{ij} \leftarrow \begin{cases} 1 & \text{if } j = \arg\min_l \| \left(\sum_v \alpha_v \mathbf{H}_v \mathbf{W}_v\right)_i - \mathbf{f}_l \|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{8.13}$$

4.2. Optimizing for F

When optimizing for $\mathbf{F}$ we retrieve the same criterion as for $\mathbf{G}$, the difference lying in the constraints

$$\min_{\mathbf{F}} \quad \left\| \sum_v \alpha_v \mathbf{H}_v \mathbf{W}_v - \mathbf{G}\mathbf{F} \right\|^2. \tag{8.14}$$

This problem is an instance of an ordinary least-squares problem whose exact solution is easy to find and is given as

$$\mathbf{F} = (\mathbf{G}^\top \mathbf{G})^{-1} \mathbf{G}^\top \left(\sum_v \alpha_v \mathbf{H}_v \mathbf{W}_v \right). \tag{8.15}$$

4.3. Optimizing for $\mathbf{W}_1, \ldots, \mathbf{W}_V$

Optimizing for the different $\mathbf{W}_{v=1}^{v=V}$ matrices results in V sub-problems. For a specific $\mathbf{W}_v$, the resulting problem is $\min_{\mathbf{W}_v} \|\mathbf{H}_v - \mathbf{G}\mathbf{F}\mathbf{W}_v^\top\|^2$ which can be solved using a singular value decomposition as follows:

$$\mathbf{W}_v = \mathbf{U}\mathbf{V}^\top \quad s.t. \quad \mathbf{U}, \Sigma, \mathbf{V} = \texttt{SVD}(\mathbf{X}_v^\top \mathbf{G}\mathbf{F}) \tag{8.16}$$

For more details on the derivation we refer the reader to [Yamamoto, 2014; Fettal, 2022c].

4.4. Optimization Algorithm

The main steps are summarized in Algorithm 9. The loss function might not be strictly decreasing due to the use of relaxations for the sub-problems, but it should nevertheless have an overall decreasing trend, as shown in figure 8.2 where we observe that our algorithm does not need many iterations to converge.

4.5. Complexity Analysis

For simplicity, we suppose that $f \in O(k)$, that $d_1, \ldots, d_V \in O(d)$ and that $|\mathcal{E}_1|, \ldots, |\mathcal{E}_V| \in O(|\mathcal{E}|)$. We also suppose that $k << n, d$, which is almost always the case in real-world attributed graph datasets. Note that in what follows, the multiplication of matrix $\mathbf{G}$ with another matrix amounts only to a re-indexing of this other matrix, because of the structure of $\mathbf{G}$.

- **First-order Neighborhood Propagation.** This step consumes roughly $O(v|\mathcal{E}|d)$ oper-

Algorithm 9: Block Coordinate Descent (BCD) for LMGEC

Input : - Sequence of views $\{(\mathbf{A}_v, \mathbf{X}_v)\}_{v=1,\dots,V}$
- Number of clusters k - Embedding dimension f
- Temperature τ - Tolerance ϵ
- Maximum number of iterations max_iter

Output: - Consensus membership indicator $\mathbf{G} \in \{0,1\}^{n \times k}$
- Consensus embedding centers $\mathbf{F} \in \mathbb{R}^{k \times f}$
- Consensus embedding matrices $\mathbf{W} \in \mathbb{R}^{d \times f}$

$\forall v$ $\mathbf{H}_v \leftarrow \mathbf{A}_v \mathbf{X}_v$;
$\forall v$ Initialize $\mathbf{W}_v$ through a truncated SVD on $\mathbf{H}_v$;
$\forall v$ Compute α_v using formula (8.10);
Initialize $\mathbf{G}$ and $\mathbf{F}$ through a k-means on $\sum_v \alpha_v \mathbf{H}_v \mathbf{W}_v$;
while *change in loss* $> \epsilon$ **and** max_iter *not reached* **do**
  $\forall v$ Update $\mathbf{W}_v$ using formula (8.16);
  Update $\mathbf{G}$ using formula (8.13);
  Update $\mathbf{F}$ using formula (8.15);
  $loss \leftarrow \sum_v \alpha_v \|\mathbf{H}_v - \mathbf{G}\mathbf{F}\mathbf{W}_v^\top\|$;
end

ations.

- **Initializing $\{\mathbf{W}_v\}_{v=1}^{v=V}$.** This step takes $O(vnd\log(k))$ when using a randomized SVD algorithm.
- **Computing $\{\alpha_v\}_{v=1}^{v=V}$.** Here it is necessary to compute the inertia as well as the sets $\{\mathbf{G}_v\}_{v=1}^{v=V}$ and $\{\mathbf{F}_v\}_{v=1}^{v=V}$ for each view, totalling around $O(vndk)$ operations.
- **Initializing G and F.** Computing the summation $\sum_v \alpha_v \mathbf{H}_v \mathbf{W}_v$ and the application of k-means on it amounts to $O(ndk)$ operations, where the number of iterations of k-means is held constant.
- **Updating $\{\mathbf{W}_v\}_{v=1}^{v=V}$.** Like their initialization steps, this step takes roughly $O(vnd\log(k))$.
- **Updating G.** The rule associated with this step takes $O(nk^2)$ computations.
- **Updating F.** This rule is computed in $O(vndk)$, as a result of computing the embeddings for the different views $\mathbf{H}_v\mathbf{W}_v$.
- **Objective Value Calculation.** The computation here can be performed in $O(nd)$.
- **Overall Complexity.** Altogether, the total computation time for our algorithm is $O(v|\mathcal{E}|d + tvndk)$, where t is the number of iterations of LMGEC.

5. Experimentation

In this section we present the experimental setup and results. We start by introducing the datasets and the evaluation metrics used in relation to clustering, the methods used for

Table 8.1: Characteristics of the Datasets. For wiki, there are two topologies and two features matrices leading to four possible combinations/views.

Dataset	Multi-view type	#Views	#Nodes	#Features	#Edges	#Clusters
ACM [Wang, 2019]	Topology	2	3,025	1,830	29,281 2,210,761	3
DBLP [Wang, 2019]	Topology	3	4,057	334	11,113 5,000,495 6,776,335	4
IMDB [Wang, 2019]	Topology	2	4,780	1,232	98,010 21,018	3
Amazon Photos [Shchur, 2018]	Features	2	7,487	745 7,487	119,043	8
Wiki [Yang, 2015]	Both	4	2405 2405	4973 4973	24,357 12,025	17

comparison with LMGEC, and the experimental settings. We then present the results in terms of quality of clustering, followed by analysis where we consider embedding, complexity, sensitivity, and robustness in the face of noisy views.

5.1. Datasets and Metrics

In order to demonstrate the generic nature of our model we looked at the three possible types of multi-view attributed graph datasets:

1. **Datasets with Heterogeneous Graph Topology.** These datasets have the same set of features $\mathbf{X}$ but multiple graph topologies, i.e. multiple adjacency matrices $\{\mathbf{A}_v\}_{v=1}^{v=V}$. They include ACM, DBLP and IMDB.

2. **Datasets with Heterogeneous Features.** These datasets have the same graph structure $\mathbf{A}$ but multiple sets of features $\{\mathbf{X}_v\}_{v=1}^{v=V}$. The example we chose was Amazon Photos.

3. **Datasets with Both.** These datasets have multiple graph structures $\{\mathbf{A}_v\}_{v=1}^{v=V}$ as well as multiple sets of features $\{\mathbf{X}_v\}_{v=1}^{v=V}$, The only such dataset is Wiki for which we create the additional views from the initial data, we initially have a single topology and features, we then generate a second topology using a nearest neighbor graph based on the cosine distance and a second set of features by using a log-scale of the original ones.

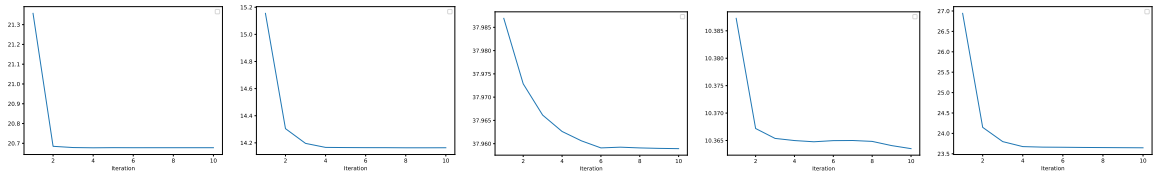
The characteristics of these datasets are given in table 8.1. In quantifying the quality of a clustering we use four metrics: clustering accuracy (CA), clustering F1-score (F1) [Rijsbergen CJ, 1979], normalized mutual information (NMI) [Strehl, 2002] and adjusted rand index (ARI) [Hubert, 1985].

Table 8.2: Clustering results on ACM, DBLP and IMDB. Best results are highlighted in bold font and the second best results underlined.

Model	ACM				DBLP				IMDB			
	CA	F1	NMI	ARI	CA	F1	NMI	ARI	CA	F1	NMI	ARI
LINE-avg	0.6479	0.6594	0.3941	0.3433	0.875	0.866	0.6681	0.7056	0.4719	0.2985	0.0063	-0.009
GAE	0.8216	0.8225	0.4914	0.5444	0.8859	0.8743	0.6925	0.741	0.4298	0.4062	0.0402	0.0473
GAE-avg	0.699	0.7025	0.4771	0.4378	0.5558	0.5418	0.3072	0.2577	0.4442	0.4172	0.0413	0.0491
MNE	0.637	0.6479	0.2999	0.2486	out of memory error				0.3958	0.3316	0.0017	0.0008
PMNE(n)	0.6936	0.6955	0.4648	0.4302	0.7925	0.7966	0.5914	0.5265	0.4958	0.3906	0.0359	0.0366
PMNE(r)	0.6492	0.6618	0.4063	0.3453	0.3835	0.3688	0.0872	0.0689	0.4697	0.3183	0.0014	0.0115
PMNE(c)	0.6998	0.7003	0.4775	0.4431	out of memory error				0.4719	0.3882	0.0285	0.0284
RMSC	0.6315	0.5746	0.3973	0.3312	0.8994	0.8248	0.7111	0.7647	0.2702	0.3775	0.0054	0.0018
PwMC	0.4162	0.3783	0.0332	0.0395	0.3253	0.2808	0.019	0.0159	0.2453	0.3164	0.0023	0.0017
SwMC	0.3831	0.4709	0.0838	0.018	0.6538	0.5602	0.376	0.38	0.2671	0.3714	0.0056	0.0004
O2MA	0.888	0.8894	0.6515	0.6987	0.904	0.8976	0.7257	0.7705	0.4697	0.4229	0.0524	0.0753
O2MAC	0.9042	0.9053	0.6923	0.7394	0.9074	0.9013	0.7287	0.778	0.4502	0.4159	0.0421	0.0564
DMGI	0.8973	0.8985	0.6974	0.7296	0.8722	0.8691	0.6931	0.7034	0.5827	0.4253	**0.1317**	0.1457
MvAGC	0.8975	0.8986	0.6735	0.7212	0.9277	0.9225	0.7727	0.8276	0.5633	0.3783	0.0371	0.0940
MAGC	0.8806	0.8835	0.6180	0.6808	0.9282	0.9237	**0.7768**	0.8267	**0.6125**	**0.4551**	0.1167	**0.1806**
LMGEC	**0.9302**	**0.9311**	**0.7513**	**0.8031**	**0.9285**	**0.9241**	0.7739	**0.8284**	0.5893	0.4267	0.0632	0.1294

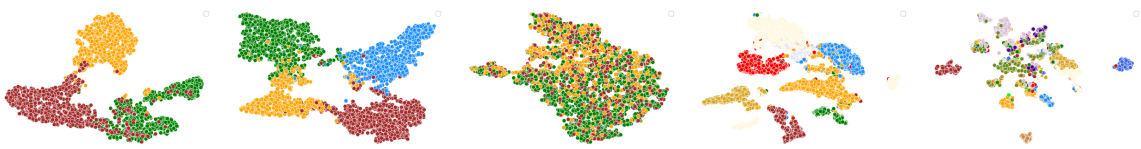
Table 8.3: Clustering results on Amazon Photos and Wiki. Additionally, we report the performance of LMGEC on each individual view (for the other datasets see figure 8.4). Note that Amazon Photos has only two views, while Wiki has four.

Model	Amazon Photos				Wiki			
	CA	F1	NMI	ARI	CA	F1	NMI	ARI
LMGEC (view 1)	0.6726	0.6451	0.5903	0.4865	0.4757	0.4154	0.4772	0.2944
LMGEC (view 2)	0.6835	0.6164	0.5971	0.4896	0.5181	*0.4463*	0.5079	0.3226
LMGEC (view 3)	-	-	-	-	0.5202	0.4333	0.5383	0.3401
LMGEC (view 4)	-	-	-	-	0.5264	0.4384	0.5362	0.3455
MAGC	0.4511	0.3359	0.4297	0.1127	0.4972	0.4084	0.5139	0.2707
MvAGC	0.6775	0.6397	0.5237	0.3968	0.3297	0.2432	0.3531	0.0864
LMGEC	**0.7117**	**0.6500**	**0.6114**	**0.5123**	**0.5333**	**0.4501**	**0.5408**	**0.3496**



(a) ACM (b) DBLP (c) IMDB (d) Photos (e) Wiki

Figure 8.2: Evolution of the loss value across iterations using BCD for LMGEC.

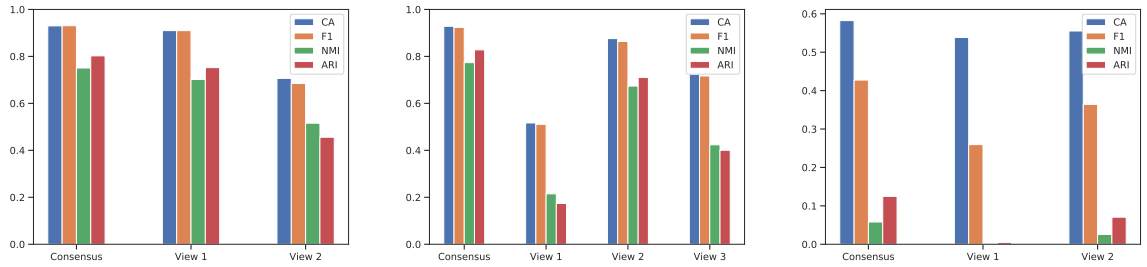


(a) ACM (b) DBLP (c) IMDB (d) Photos (e) Wiki

Figure 8.3: Two-dimensional projections of the `LMGEC` embeddings using t-SNE colored according to the real class labels.

Table 8.4: Training times. Best results are in bold font, second best results are underlined. DMGI is only applicable to datasets with one set of features.

Model	ACM	DBLP	IMDB	Wiki	Amazon Photos
DMGI	943.19	3117.87	843.96	-	-
MvAGC	<u>14.48</u>	<u>26.28</u>	<u>32.97</u>	<u>34.73</u>	<u>139.14</u>
MAGC	139.93	242.99	395.45	150.93	1661.64
LMGEC	**3.49**	**3.07**	**4.96**	**18.06**	**19.42**



(a) ACM (b) DBLP (c) IMDB

Figure 8.4: Performance of LMGEC on each individual view vs. its consensus performance when considering all views on ACM, DBLP and IMDB (for the other datasets see table 8.3).

5.2. Baselines

Below we list all the methods evaluated in our proposal.

- **LINE** [Tang, 2015]: A single-view graph embedding method. It is applied on each view and the best results are reported.
- **GAE** [Kipf, 2016]: Another single-view graph embedding method based on the autoencoder.
- **X-avg**: To utilize multiple views of a network we apply the X method to learn node representations on each single view, then average all learned representations.
- **LINE-avg** and **GAE-avg**: By this we mean that the node representations learned for each view using LINE and GAE are averaged and clustered as such.
- **MNE** [Zhang, 2018]: A scalable multi-view network embedding model. Only the graph structure information (adjacency matrix) of each view is input into this model.
- **PMNE** [Liu, 2017]: Encompasses three multi-view graph embedding methods, including network aggregation PMNE (n), results aggregation PMNE (r) and layer co-analysis PMNE (c).
- **RMSC** [Xia, 2014]: A multi-view spectral clustering method that uses Markov chains and low-rank decomposition.

- **PwMC** and **SwMC** [Nie, 2017]: PwMC is a parameter-weighted multi-view graph clustering method, while SwMC is a self-weighted multi-view graph clustering method.
- **O2MA** and **O2MAC** [Fan, 2020]: O2MAC is also an autoencoder-based multi-view graph clustering method. O2MA is a simplified version of O2MAC with the clustering loss removed from the objective function.
- **DMGI** [Park, 2020]: This is an unsupervised embedding method for attributed multiplex network embedding. This approach is only applicable to datasets with one set of features and multiple graphs.
- **MvAGC** [Lin, 2021b]: Performs graph filtering to do multi-view attributed graph clustering.
- **MAGC** [Lin, 2021c]: A multi-view graph clustering method that utilizes both node attributes and graphs.

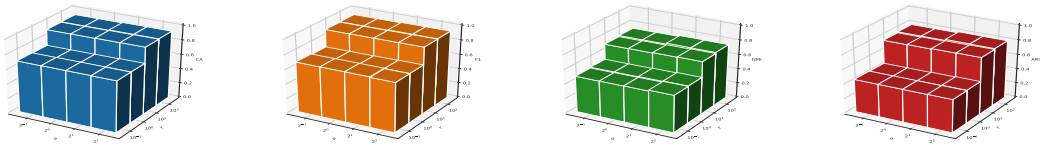
5.3. Experimental Settings

We use the clustering results reported in the original works where possible. When performing our own tests we tried to follow the setups prescribed by the authors of the different models as faithfully as possible. For our model, we set the maximum number of iterations to 30, the tolerance to 0 and $f = k + 1$ in all experiments. We performed experiments with different hyper-parameter values for β and τ. We did a tf-idf normalization of the inputs to our model, and we also centered the data after neighborhood propagation. For the values of β and τ, we try $\beta \in \{.2, 1, 2\}$ and $\tau \in \{1, 10, 100\}$ and we report the best results. The different experiments were run on the same machine with two Intel(R) Xeon(R) CPU @ 2.20GHz and 13GB RAM. Most of the source codes in the official repositories of the baselines were not optimized for GPU. Note that the results reported for our method are the averages of five runs.

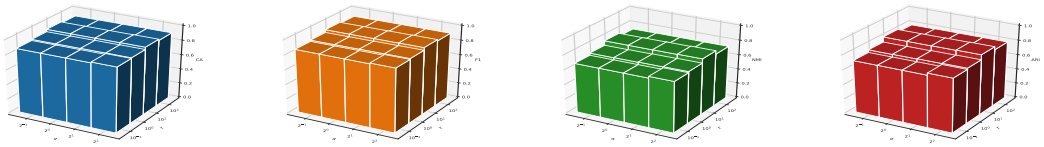
5.4. Experimental Results

Below we study in detail the results from our comparisons and highlight the interest of **LMGEC**.

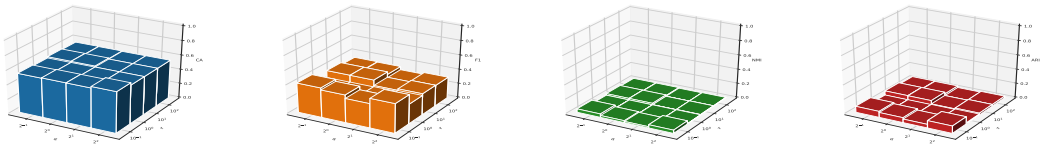
Clustering Results. Tables 8.2 and 8.3 show the results of our experiments for the clustering task. Some of the results for ACM, DBLP, IMDB and Amazon Photos are taken from [Fan, 2020; Lin, 2021c; Lin, 2021b], while results for Wiki are those that we obtained in our experiments. The pattern that emerges is that methods combining multi-view information tend to outperform those using single-view information. Our model LMGEC consistently achieves competitive performances, outperforming other models on ACM, DBLP, IMDB, Amazon Photos and Wiki on most metrics, and being competitive on IMDB, where it has



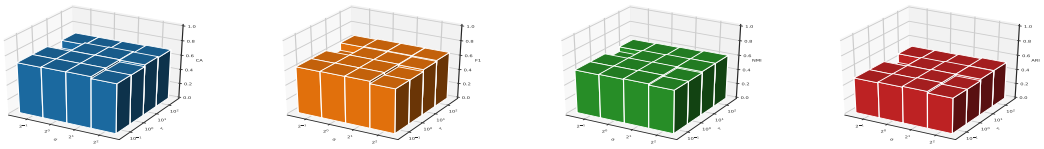
(a) ACM



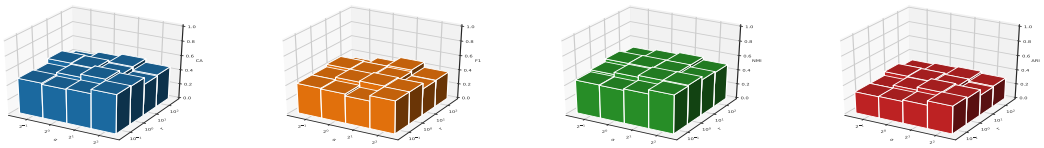
(b) DBLP



(c) IMDB



(d) Amazon Photos



(e) Wiki

Figure 8.5: Sensitivity analysis of the parameters of `LMGEC` on the graph topology heterogeneous datasets.

the second best results on two out of four performance metrics. Overall, our model offers the best results in 15 out of 20 cases and has the best or second best results in 18 out of 20 cases, showing that it is competitive despite its simple nature. Note that for datasets with multiple features, in our experiments we used only the baselines that were the best performing (on average).

We can see the benefits of our model from tables 8.2 and 8.3 and from figure 8.4, where the performance of LMGEC on individual views is shown against the consensus performance; the consensus performance is consistently better than for the individual views. In some instances LMGEC applied to the individual views even outperforms state-of-the-art models, for example, on ACM, Amazon Photos and Wiki.

Embedding Results. Figure 8.3 depicts the embeddings produced by LMGEC on the different datasets by projecting them onto a 2d-plane using t-SNE. A clustering structure is visible on all datasets apart from IMDB, where the embeddings are not very well separated.

Efficiency Results. We report the training times of our method in table 8.4 as well as those of the best performing (on average) baselines in our experiments. Our model is consistently much faster than other models, improving on the training time of the second fastest model (MvAGC) by 75%, 89%, 85%, 48% and 86% on ACM, DBLP, IMDB, Wiki and Amazon Photos respectively.

Sensitivity Analysis. In our experiments we tried various values for the β and τ hyper-parameters. Figure 8.5 illustrates the performance of LMGEC for different pairs of these parameter values on the different datasets and for the different clustering metrics. We see that the on most datasets the performance remains fairly constant, which is an indication of LMGEC's robustness. The exception is ACM, where LMGEC is sensitive to the temperature parameter τ because of the presence of uninformative views. We discuss this in greater detail in the following paragraph. As a rule of thumb, we suggest taking $\tau = 10$ and $\beta = 1$.

Robustness in the face of noisy views. In real applications noisy data sources are not uncommon and can impair the performance of multi-view models. Since LMGEC takes into account the importance of the different views via a weighting scheme based each view's inertia, it is able to filter out noisy views by increasing the temperature τ of the softmax function used in computing $\{\alpha_v\}_{v=1}^{v=V}$.

Tables 8.2 and 8.3 and figure 8.4 report the clustering performance of LMGEC on each individual view for the different datasets, as well as the consensus performance. In the case of DBLP we may consider that the first view is noisy, since LMGEC performs considerably less well on this view than on the second and third views. We remark, however, that the consensus performance is not influenced by the presence of this noisy view, which shows the robustness of LMGEC in the face of noise. The same is true of Wiki as regards the first view, albeit less significantly than for DBLP, since the performance gap is not as flagrant w.r.t the other views.

6. Conclusion

In this work we proposed a simple linear additive model that addresses the dual tasks of multi-view attributed graph representation learning and multi-view attributed clustering in a unified framework. This model is more generic than most state-of-the-art approaches in the sense that it can deal with any number of graph structures and/or any number of feature sets. Experiments showed that our model is competitive with more complex state-of-the-art models, outperforming these models on most benchmarks in terms of both performance and computation time.

The next chapter will deal with the same data as this one, but it will exclusively focus on the clustering task, specifically, we will use the summation property of kernels to propose an efficient attributed multi-view subspace clustering algorithm.

Chapter 9

Multi-view Attributed Graph Subspace Clustering

Objective

In this chapter we aimed at creating an approach that extends [Fettal, 2023b] to the multi-view setting.

Contents

1. Introduction

The explosive growth of data from diverse sources, such as social media, sensor networks, and online platforms, has led to the emergence of complex high-dimensional datasets. These datasets often contain multiple views of the same underlying data, each capturing different aspects or perspectives. Traditional clustering algorithms, designed for single-view data, face significant challenges in effectively capturing the intricate relationships and structures present across multiple views.

Multi-view subspace clustering [Gao, 2015; Sun, 2021; Chen, 2022] has emerged as a powerful paradigm to tackle this challenge by integrating information from multiple views to enhance the clustering performance. By exploring the latent subspaces within each view and leveraging the complementary nature of different views, multi-view subspace clustering algorithms can reveal hidden structures that may not be apparent when considering individual views in isolation. This not only leads to more accurate clustering results but also enables a deeper understanding of the underlying data. Specifically, multi-view subspace clustering for attributed-networks data, which consist of interconnected entities with associated attributes, has gained significant attention in recent literature [Fan, 2020; Lin, 2021b; Lin, 2021c; Pan, 2021], offering a powerful framework to analyze complex datasets. For example, one prominent application of multi-view clustering for attributed-networks data is in social network analysis as social networks often exhibit rich attributes associated with individuals or groups, such as demographic information, interests, and affiliations [Fan, 2020; Hu, 2020]. Another one is in biological research, where multi-omics data can be integrated using multiple graph convolutional networks [Wang, 2021a]. Finally, recommender systems often rely on multiple sources of information, such as user preferences, item attributes, and social connections [Kipf, 2017; Wu, 2019; Shchur, 2018].

Be that as it may, the computational complexity of multi-view subspace clustering is a significant challenge that hinders its widespread adoption in real-world applications. Despite the proposal of efficient approaches, for example, based on the popular framework which consists in using anchor based techniques which are used to decrease the sizes of the matrices handled during optimization [Kang, 2020; Sun, 2021]. We found through experimentation that the existing methods still often fall short of achieving the desired level of efficiency and clustering performance for large-scale multi-view datasets. As a result, there is a pressing need for further advancements in this area to overcome the computational challenges and unlock the full potential of multi-view subspace clustering.

In this work, we tackle the scalable multi-view subspace clustering problem differently from anchor-based methods and propose a novel algorithm that addresses the limitations of existing approaches in terms of both clustering accuracy and computational efficiency. Our algorithm leverages low-rank subspace clustering and properties of kernel features maps, to capture the complex relationships between views and enhance the robustness of clustering results in an efficient manner. Similarly to some state-of-the-art models, our approach scales linearly with the number of inputs but it scales more favorably with other factors such as

the number of dimensions and clusters. Additionally, we can exploit the inherent parallelism present in the multi-view setting, allowing for efficient processing of large-scale datasets. In summary, the key contributions of this work are threefold:

1. First, we introduce a novel framework that integrates multiple views of the data subspace structure graph into a unified consensus subspace structure graph, effectively capturing the complementary information across views and allowing for a new scalability framework for multi-view subspace clustering.

2. Second, we propose an efficient optimization strategy that scales well with very large-scale datasets. By taking advantage of the properties of kernel features maps, our algorithm significantly reduces the computational burden while maintaining state-of-the-art clustering performance. This scalability makes our approach suitable for handling real-world applications as it scales to datasets with millions of data points.

3. Third, we conduct extensive experiments, including statistical significance testing, on real-world benchmark networks of different scales and degrees of overlap, to evaluate the performance of our algorithm in comparison to state-of-the-art multi-view subspace clustering methods as well as attributed-network specific multi-view approaches. Our experiments are entirely reproducible, ensuring transparency and enabling others to validate our results.

2. Related Work

Scalable Multi-view Subspace Clustering [Lin, 2021c] introduced Large-Scale Multi-View Subspace Clustering in Linear Time (**LMVSC**) [Kang, 2020], a method where smaller graphs are created for each view, and then integrated to enable spectral clustering on a reduced graph size, inspired by the concept of anchor graph. Scalable Multi-view Subspace Clustering with Unified Anchors (**SMVSC**) [Sun, 2021] approach integrates anchor learning and graph construction into a unified optimization framework, resulting in more accurate representation of latent data distribution by learned anchors and a more discriminative clustering structure. In [Chen, 2022], an efficient orthogonal multi-view subspace clustering (**OMSC**) approach is proposed; it uses joint anchor learning, graph construction and partitioning. Recently, [Zhang, 2023] developed a Flexible and Diverse Anchor Graph Fusion for Scalable Multi-view Clustering (**FDAGF**) that introduces a fusion strategy for multi-size anchor graphs.

Graph Multi-view Clustering [Pan, 2021] proposed a multi-view contrastive graph clustering (**MCGC**) method to learn a consensus graph with contrastive regularization over the learned graph matrix. [Lin, 2021c] introduced a novel multi-view attributed graph clustering (**MAGC**) framework, which incorporates node attributes and graphs. The approach stands out in three key aspects: utilizing graph filtering for smooth node representation instead of

deep neural networks, learning a consensus graph from data to address noise and incompleteness in the original graph, and exploring high-order relations through a flexible regularizer design. The **MvAGC** model [Lin, 2021b] is a simple yet effective approach for Multi-view Attributed Graph Clustering. It applies a graph filter to smooth the features without relying on neural network parameter learning. A reduced computational complexity is achieved through a novel anchor point selection strategy. Furthermore, a new regularizer is introduced to explore high-order neighborhood information. [Fettal, 2023c] introduced (**LMGEC**), a generic linear model for performing multi-view attributed graph representation learning and clustering simultaneously. It uses a simple weighing scheme based on inertia and a single-hop neighborhood propagation.

3. Preliminaries

In what follows, matrices are denoted in boldface uppercase letters and vectors in boldface lowercase letters. $[n]$ corresponds to the integer set $\{1, \ldots, n\}$. `Tr` is the trace operator. $\oplus$ denotes column-wise concatenation.

3.1. Subspace Clustering

Given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and a desired number of clusters k, most subspace clustering problems have the following form:

$$\min_{\mathbf{C}} \quad \|\mathbf{X} - \mathbf{C}\mathbf{X}\|^2 + \Omega(\mathbf{C}) \quad \text{s.t.} \quad \mathbf{C} \in \mathcal{C} \tag{9.1}$$

where $\mathbf{C}$ is called a coefficient matrix, Ω is a regularization function and $\mathcal{C}$ is the set of feasible solutions. Once a solution $\mathbf{C}$ is found, a subspace affinity graph represented using its adjacency matrix $\mathbf{W}$ is constructed, typically as $\mathbf{W} = (|\mathbf{C}| + |\mathbf{C}^\top|)/2$. Finally, a partition of k groups is obtained via spectral clustering on $\mathbf{W}$. By considering different Ω and $\mathcal{C}$, we retrieve different subspace clustering techniques, for example, we can name low-rank approaches [Liu, 2010] and sparse approaches [Elhamifar, 2013] among others.

3.2. Scalable Subspace Clustering

[Fettal, 2023b] proposed a scalable low-rank subspace clustering model. Specifically, the problem they considered is:

$$\min_{\mathbf{U}} \quad \|\mathbf{X} - \mathbf{U}\mathbf{U}^\top\mathbf{X}\|^2 \quad \text{s.t.} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{I}. \tag{9.2}$$

They then use nonnegative kernel maps to create an affinity matrix $\mathbf{W} = \Phi(\mathbf{U})\Phi(\mathbf{U})^\top$. The advantage of computing the affinity matrix in this manner is that spectral partitioning can be obtained via singular values decomposition on $\Phi(\mathbf{U})$ instead of performing eigendecomposition on $\mathbf{W}$, leading to substantial gains in spatial and computational efficiencies.

3.3. Multi-view Subspace Clustering

Given a set of features $\mathbf{X}_1, \ldots, \mathbf{X}_V$ describing the same data points. A rudimentary multi-view subspace clustering model can consider a problem of the following form:

$$\begin{aligned} \min_{\mathbf{F}, \mathbf{C}_1, \ldots, \mathbf{C}_V} \quad & \sum_v \|\mathbf{X}_v - \mathbf{C}_v \mathbf{X}_v\|^2 + \mathtt{Tr}(\mathbf{F}\mathbf{C}_v\mathbf{F}) \\ \text{such that} \quad & \mathbf{F} \in \{0,1\}^{n \times k}, \quad \mathbf{F}\mathbf{1} = \mathbf{1} \\ & \mathbf{C}_1, \ldots, \mathbf{C}_V \in \mathcal{C}, \end{aligned} \tag{9.3}$$

where $\mathbf{C}_v$ is the coefficient matrix of the v-th view and $\mathbf{F}$ is a a consensus partition matrix. However, although the block structures in different view specific coefficient matrices are similar, the magnitudes and the signs of the entries in each C_v can vary greatly. To solve this issue, [Gao, 2015] instead considered using the affinity matrices $\mathbf{W}_v$ instead of the coefficient matrices $\mathbf{C}_v$. We generalize that formulation by dropping the specific constraints they used. This results in the following generic multi-view subspace clustering problem:

$$\begin{aligned} \min_{\mathbf{G}, \mathbf{C}_1, \ldots, \mathbf{C}_V} \quad & \sum_v \|\mathbf{X}_v - \mathbf{C}_v \mathbf{X}_v\|^2 \\ + \quad & \mathtt{Tr}\left(\mathbf{F}^\top \left(\mathbf{I} - \tfrac{|\mathbf{C}_v| + |\mathbf{C}_v^\top|}{2}\right) \mathbf{F}\right) \\ & \text{such that} \quad \mathbf{F} \in \{0,1\}^{n \times k}, \quad \mathbf{F}\mathbf{1} = \mathbf{1}, \mathbf{C}_1, \ldots, \mathbf{C}_V \in \mathcal{C}. \end{aligned} \tag{9.4}$$

Once solutions $\mathbf{C}_1, \ldots, \mathbf{C}_V$ are available, depending on the regularization used, spectral clustering can be performed on $\mathbf{W} = \sum_v \mathbf{C}_v$ to obtain the partition matrix $\mathbf{F}$.

4. Methodology

We consider the multi-view framework presented in problem 9.4 and introduce low-rank constraints into it as in 9.2 and also use their affinity matrix. However, here we consider a symmetrically normalized affinity matrix instead of the unnormalized one. We obtain the following problem:

$$\begin{aligned} \min_{\mathbf{G}, \mathbf{U}_1, \ldots, \mathbf{U}_V} \quad & \sum_v \|\mathbf{X}_v - \mathbf{U}_v \mathbf{U}_v^\top \mathbf{X}_v\|^2 + \ \mathtt{Tr}(\mathbf{F}^\top (\mathbf{I} - \mathbf{W}_v)\mathbf{F}) \\ \text{such that} \quad & \mathbf{W}_v = \mathbf{D}_v^{-\frac{1}{2}} \Phi(\mathbf{U}_v) \Phi(\mathbf{U}_v)^\top \mathbf{D}_v^{-\frac{1}{2}} \\ & \mathbf{F} \in \{0,1\}^{n \times k}, \quad \mathbf{F}\mathbf{1} = \mathbf{1} \end{aligned} \tag{9.5}$$

where $\mathbf{U} \in \mathbb{R}^{n \times f}$ is a low rank matrix, $\mathbf{W}_v$ is the normalized affinity matrix of view v. If we were to solve this problem directly, the consensus partition $\mathbf{F}$ would be obtained via spectral clustering on

$$\mathbf{W} = \sum_v \mathbf{W}_v = \sum_v \mathbf{D}_v^{-\frac{1}{2}} \Phi(\mathbf{U}_v) \Phi(\mathbf{U}_v)^\top \mathbf{D}_v^{-\frac{1}{2}}. \tag{9.6}$$

The problem of applying spectral clustering via the proper decomposition of this matrix would result in a computational complexity of $\mathcal{O}(n^2 k)$ and a spatial complexity of $\mathcal{O}(n^2)$.

This means that we would lose the efficiency argued by [Fettal, 2023b] and thus lose the advantage of using such an approach in the first place. Therefore, we propose to solve this problem via the summation property of kernels.

4.1. Weighing Views relative to their Clusterability

We additionally introduce a weighting scheme on the view-specific terms in problem 9.5 as the different views should not always have the same contribution level towards the consensus affinity matrix. Here, we evaluate the importance of each view using the affinity matrix associated with said view. Specifically, we consider the following problem:

$$\begin{aligned} \min_{\mathbf{G},\mathbf{U}_1,\dots,\mathbf{U}_V} & \sum_v \|\mathbf{X}_v - \mathbf{U}_v\mathbf{U}_v^\top\mathbf{X}_v\|^2 \;+\; \lambda_v \;\; \mathtt{Tr}(\mathbf{F}^\top(\mathbf{I}-\mathbf{W}_v)\mathbf{F}) \\ \text{such that} \quad & \mathbf{W}_v = \mathbf{D}_v^{-\frac{1}{2}}\Phi(\mathbf{U}_v)\Phi(\mathbf{U}_v)^\top\mathbf{D}_v^{-\frac{1}{2}} \\ & \mathbf{F}\in\{0,1\}^{n\times k}, \quad \mathbf{F}\mathbf{1}=\mathbf{1} \end{aligned} \tag{9.7}$$

where we define the regularization vector $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_V]$ with respect to the cluster-ability of the v-th view:

$$\begin{aligned} \lambda_v &= \frac{\mathtt{exp}(\ \mathtt{Tr}(\ \mathbf{G}_v(\mathbf{I}-\mathbf{W}_v)\mathbf{G}_v)\ /\ T\ )}{\sum_w \mathtt{exp}(\ \mathtt{Tr}(\ \mathbf{G}_w(\mathbf{I}-\mathbf{W}_w)\mathbf{G}_w)\ /\ T\ )} \\ \text{such that} \quad \forall w\in[V] \qquad & \mathbf{W}_v = \Phi(\mathbf{U}_w)\Phi(\mathbf{U}_w)^\top \\ & \mathbf{G}_w\in\{0,1\}^{n\times k}, \quad \mathbf{G}_w\mathbf{1}=\mathbf{1} \end{aligned} \tag{9.8}$$

where T is a temperature parameter and $\mathbf{G}_1, \dots, \mathbf{G}_V$ are different view-specific partition matrices we solve for using spectral clustering by applying singular value decomposition on $\mathbf{D}_w^{-\frac{1}{2}}\Phi(\mathbf{U}_w)$ and then applying a clustering algorithm on the resulting left singular vectors and repeating the process for each $v \in [V]$.

4.2. Multi-view Scalability via Kernel Summation

Since we are using kernels to compute the view-specific affinity matrices $\mathbf{C}_1, \dots, \mathbf{C}_V$, we propose to use another property of kernels in order to obtain a factorized matrix consensus affinity matrix, which will then allow for efficient spectral clustering. Consider the following:

Property 1. *Given two kernels k_a and k_b then their summation $k(x,y) = k_a(x,y) + k_b(x,y)$ is also kernel.*

This means that matrix 9.6 is also a kernel and, consequently, has a decomposition of the form $\mathbf{B}\mathbf{B}^\top$ for some matrix $\mathbf{B}$ (this is property of kernel matrices). So here to solve the efficiency issue of combining the different view-specific affinity matrix, we have to find a way to find a vector realization $\mathbf{B}$ for the decomposition of 9.6. Once we have $\mathbf{B}$, we can obtain a spectral clustering via singular value decomposition of $\mathbf{B}$ instead of eigendecomposition of $\mathbf{B}\mathbf{B}^\top$. We can find this realization by noticing that:

Property 2. *Given kernels k_a and k_b and their associated feature maps Φ_a and Φ_b, the feature map associated with their summation kernel k is $\Phi(x) = \Phi_a(x) \oplus \Phi_b(x)$.*

Using this fact, we can infer that a vector realization is $\mathbf{B} = \left[\mathbf{D}_1^{-\frac{1}{2}}\Phi(\mathbf{U}_1), \ldots, \mathbf{D}_V^{-\frac{1}{2}}\Phi(\mathbf{U}_V)\right]$. This decomposition means that the complexity of the approach is proportional to dimension m of the feature map Φ. This leads to large speedup and memory saving when $m << n$.

4.3. Optimization

In addition to the use of properties associated with kernel feature maps, the efficiency of our approach also stems from the fact that it does not need an iterative algorithm, contrary to many other state-of-the-art models, e.g, [Pan, 2021]. The algorithm consists of the following steps:

Solving for $\mathbf{C}_1, \ldots, \mathbf{C}_V$ The first step is fixing $\mathbf{C}_1, \ldots, \mathbf{C}_{v-1}, \mathbf{C}_{v+1}, \ldots, \mathbf{C}_V$ and F in problem 9.5 and reiterating for each $v \in [V]$. The resulting problem is 9.2, and the solution is to set $\mathbf{U}$ as the f left singular vectors of $\mathbf{X}$ associated with the largest singular values. Note that to solve our approach in one step, as opposed to [Gao, 2015], we consider $\mathbf{W}_v$ to not be a function of $\mathbf{C}_v$.

Computing $\lambda_1, \ldots, \lambda_V$ Once we obtained the singular vectors associated with each coefficient matrix $\mathbf{C}_v$, we compute the corresponding λ_v using formula 9.8.

Solving for $\mathbf{F}$ When fixing every $\mathbf{C}_v$, we obtain this problem in with respect to $\mathbf{F}$, we obtain:

$$\begin{aligned} \min_{\mathbf{F}} \quad & \sum_v \mathtt{Tr}(\mathbf{F}(\mathbf{I} - \mathbf{W}_v)\mathbf{F}) \\ \text{such that} \quad & \mathbf{F} \in \{0,1\}^{n \times k}, \quad \mathbf{F1} = \mathbf{1}. \end{aligned} \tag{9.9}$$

The solution, then, consists in using property 2, and noticing that:

$$\begin{aligned} \mathbf{W} &= \sum_v \mathbf{W}_v = \sum_v \lambda_v \mathbf{D}_v^{-\frac{1}{2}} \Phi(\mathbf{U}_v)\Phi(\mathbf{U}_v)^\top \mathbf{D}_v^{-\frac{1}{2}} \\ &= \left(\oplus_v \sqrt{\lambda_v}\mathbf{D}_v^{-\frac{1}{2}}\Phi(\mathbf{U}_v)\right)\left(\oplus_v \sqrt{\lambda_v}\mathbf{D}_v^{-\frac{1}{2}}\Phi(\mathbf{U}_v)\right)^\top. \end{aligned} \tag{9.10}$$

The solution $\mathbf{F}$ of this problem is obtained by applying k-means clustering on the left singular values corresponding to the largest k singular values of matrix $\oplus_v \sqrt{\lambda_v}\mathbf{D}_v^{-\frac{1}{2}}\Phi(\mathbf{U}_v)$.

A pseudo-code for our approach, is given in algorithm 10.

4.4. Complexity Analysis

To complement our intuitive justification for the efficiency of our approach, we theoretically analyze its complexity:

Solving for $\mathbf{C}_1, \ldots, \mathbf{C}_V$ This process takes $O(vnd\log(f))$ where f is the chosen rank due to the randomized singular values decomposition.

Computing $\lambda_1, \ldots, \lambda_V$ This has a complexity of computing the projections using feature map Φ, we consider the case of using a quadratic polynomial feature map whose computational complexity is $O(nf^2)$. The computational complexity of this step is then $O(Vnf^2log(k))$.

Solving for $\mathbf{F}$ Finally the process of computing the concatenated matrix $\mathbf{B}$ and then applying singular value decomposition is $O(Vnf^2\log(k))$. Applying k-means after results in $O(tVnf^2k)$ operations.

Using other Kernels It is possible to use other nonnegative kernels than the quadratic polynomial. However, in the case of infinite dimensional kernels such as RBF and sigmoid kernels or in the case of polynomials kernels with large even degrees, it is necessary to use kernel feature maps approximations techniques such Nystroem [Williams, 2000] or polynomial sketching [Pham, 2013]. In that case the factor f^2 will be replaced with the dimension of the feature map approximation and the complexity of computing the approximation of the kernel has to be added to the total complexity of our approach.

In table 9.1, we report the complexity of several state-of-the-art algorithms. We see that compared to ours, a lot more factors have to be taken into consideration. Note that we added the complexity of learning graph representations (using 9.11) to non-graph methods.

Algorithm 10: MvSCK

Require: Data matrices $\mathbf{X}_1, \ldots, \mathbf{X}_V$, number of clusters k, number of components f, kernel feature map Φ, temperature T.
Ensure: Partition matrix $\mathbf{F}$
  for $v \in [V]$ **do**
    Subtract column means from $\mathbf{X}_v$
    $\mathbf{U}_v, \_, \_ = \texttt{randomized\_svd}(\mathbf{X}_v, f)$
    $\mathbf{B}_v = \Phi(\mathbf{U}_v)$
    $\mathbf{d}_v = (\mathbf{B}_v \ @ \ \mathbf{B}_v.\texttt{sum}(0))[:, \texttt{None}]$
    $\mathbf{B}_v = \mathbf{B}_v * \mathbf{d}_v^{-0.5}$
    $\mathbf{Q}_v, \_, \_ = \texttt{randomized\_svd}(\mathbf{B}_v, k+1)$
    $\mathbf{G}_v = \texttt{KMeans}(k).\texttt{fit\_predict}(\mathbf{Q}_v[:, 1:])$
  end for
  Compute $\lambda_1, ..., \lambda_V$ via formula 9.8 with temperature T using $\{\mathbf{G}_v\}_{v\in[V]}$ and $\{\mathbf{W}_v\}_{v\in[V]}$
  $\mathbf{B} = \oplus_v \lambda_v \mathbf{B}_v$
  $\mathbf{d} = (\mathbf{B} \ @ \ \mathbf{B}.\texttt{sum}(0))[:, \texttt{None}]$
  $\mathbf{B} = \mathbf{B} * \mathbf{d}^{-0.5}$
  $\mathbf{Q}, \_, \_ = \texttt{randomized\_svd}(\mathbf{B}, k+1)$
  $\mathbf{F} = \texttt{KMeans}(k).\texttt{fit\_predict}(\mathbf{Q}[:, 1:])$

5. Experiments

In what follows, we refer to our approach as MvSCK.

Table 9.1: Complexity of some of the considered approaches on network data. m represents the number of anchors. $|E|$ is the number of edges in the graphs used in formula 9.11. For the sake of simplicity, we consider that the dimension of the features and the number of graph edges in the different views are the same.

Method	**Time Complexity**
LMVSC	$\mathcal{O}(Vp\|E\|d + nm^3V + m^3V^3 + mVn + nk^2t)$
FDAGF	$\mathcal{O}(Vp\|E\|d + rVdm^2 + rm^2n + rVdmn)$
SMVSC	$\mathcal{O}(Vp\|E\|d + Vd^3 + Vd^2k^2 + md^2 + dmk^2 + nm^3)$
MAGC	$\mathcal{O}(Vn^2d)$
MvAGC	$\mathcal{O}(tm^3V + tVm^2nd)$
LMGEC	$\mathcal{O}(V\|E\|d + Vndk)$
MvSCK	$O(Vp\|E\|d + Vnf^2\log(k))$

5.1. Datasets

We evaluate our method on six benchmark datasets with different degrees of overlap, two small scale ones, ACM and DBLP [Fan, 2020], two medium, Amazon Photos and Amazon Computers [Shchur, 2018], and two large, OGBN-ArXiv and OGBN-Products [Hu, 2020]. Their statistical summary is described in Table 9.2.

5.2. Experimental Setup

For comparison, we validate our approach against the state-of-the-art approaches discussed in the related work using clustering accuracy (CA), clustering F1-score (CF1), adjusted rand index (ARI) and normalized mutual information (NMI). Implementations of the comparative approaches used are the official ones. Parameters are set based on the recommendations of the authors. When these recommendations are not available, we set parameters similarly to how they were set on datasets of similar dimensions. Furthermore, in order to perform fair comparisons, we remove any supervised signal when running experiments in the original codes, this includes behaviors such as validating runs based on supervised metrics such as accuracy or setting random seeds. For our approach, the kernel we used is quadratic kernel, that was discussed in [Fettal, 2023b]. The number of components parameter f was set equal to the number of clusters on all datasets with the exceptions of OGBN-Products where it was set to ten due to its large size. Note that all metrics reported are averaged over five different runs. All experiments were performed on a 64gb RAM laptop with a 12th Gen Intel(R) Core(TM) i9-12950HX (24 CPUs) processor with a frequency of 2.3GH as well as a 12GB GeForce RTX 3080 Ti GPU.

5.3. Learning Node Representations

To perform clustering on network datasets using generic approaches such as ours as well as LMVSC, SMVSC, FDAGF and OMSC. We first have to learn node embeddings. So given a multi-view attributed-networks consisting of adjacency matrices $\mathbf{A}_1, \ldots, \mathbf{A}_V$ and features matrices $\mathbf{X}_1, \ldots, \mathbf{X}_V$, we learn embeddings via Laplacian smoothing, this techniques is well-known and was used to deal with graph data in works such as [Pan, 2021; Fettal, 2023c]. It consists in consecutive neighborhoods averaging steps:

$$\forall v \in [V] \qquad \mathbf{X}_v \leftarrow \mathbf{A}_v^p \mathbf{X}_v \tag{9.11}$$

where p is the propagation order. Specifically, in our experiments, we set the propagation order to 2, 2, 20, 40, 60 and 100 to ACM, DBLP, Amazon Photos, Amazon Computers, OGBN-ArXiv and OGBN-Products, respectively. This step can be seen as a preprocessing step that is independent from the aforementioned approaches. This methodology is built upon the work done on simplifying graph convolutional networks [Wu, 2019].

5.4. Clustering Results

The clustering results on the small, medium and large graph datasets are reported in tables 9.3, 9.4 and 9.5 , respectively. As shown in these tables, our approach outperforms other approaches over most datasets and metrics even when compared to methods that are specifically tailored to graphs datasets. LMVSC is competitive with our approach on Amazon Computers, it outperforms it in terms of F1 and ARI but ours has better accuracy and NMI.

Some approaches have shown large discrepancies between the results reported in the original works and the one we obtained by running their codes on the same datasets. Due to space considerations, we do not report those original results here. Most of the approaches fail to scale to OGBN-ArXiv and OGBN-Products. We also have that OMSC fails to retrieve the correct number of clusters in Amazon Photos and Computers, which leads to the impossibility of computing its CF1 scores for it.

5.5. Ablation on $\boldsymbol{\lambda}$

In figure 9.1, we study the effect of the regularization vector $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_V]$ on five datasets. We see that it results in a boost in performance on four out of the five datasets we considered for ablation and retains the same performance on the remaining one, ACM.

5.6. Running Times

The running times of the different models on all datasets are shown in table 9.6. Our approach is the fastest one, we have the fastest approach on small, medium and large graph datasets by a wide margin. Several approaches exceed the three hours threshold on OGBN-

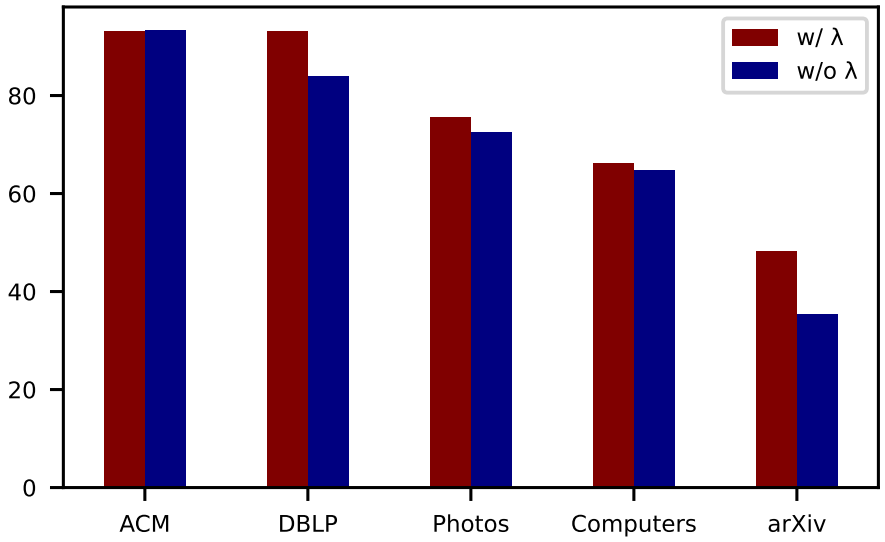


Figure 9.1: Clustering accuracy with and without the regularization vector $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_V]$.

ArXiv and OGBN-Products.

5.7. Statistical Significance Testing

Figure 9.2 shows the results of the Holm mean-rank post-hoc test [Holm, 1979] over the clustering results. We compute the mean rank over rankings generated based on each (dataset, metric, run) triplet. We see that our approach is ranked first by a large margin followed by LMGEC and LMVSC as second and third. After these three come multiple overlapping groups of approaches.

In figure 9.3, we report the results of the Holm mean-rank post-hoc test over the execution times of the different algorithms. Ours is the most efficient followed by LMGEC. LMVSC and MvAGC are ranked third, followed by SMVSC, OMSC, and then a group composed of MAGC and FDAGF. In the last position we find MCGC.

5.8. Experimenting with other Kernels

In table 9.7 we report our approach's results with different kernels, specifically, we use an exact quadratic kernel, and Nystroem approximations of the radial basis function (RBF) and sigmoid kernels. For the approximations, we use $10 * k$ components except on the Products dataset which is too large and for which we use 100 components only. The results show that using different kernels leads to similar performances except on Products where the dimension of the approximations seems to not be sufficient and leads to a poor approximation of the feature maps; the same is true for the sigmoid kernel on ArXiv. Note that in the context of RBF and sigmoid kernels, we use approximations due to the fact that their exact feature maps are infinite dimensional.

Table 9.2: Characteristics of the Datasets. The imbalance is given as the ratio between the cardinalities of the most frequent and least frequent classes.

Dataset	Scale	Nodes	Features	Graph and Edges	Clusters	Imbalance
ACM	Small	3,025	1,830	Co-Subject (29,281) Co-Author (2,210,761)	3	1.1
DBLP	Small	4,057	334	Co-Author (11,113) Co-Conference (5,000,495) Co-Term (6,776,335)	4	1.6
Amazon Photos	Medium	7,487	745 7,487	Co-Purchase (119,043)	8	5.7
Amazon Computers	Medium	13,381	767 13,381	Co-Purchase (504,937)	10	17.5
OGBN-ArXiv	Large	169,343	128	k-NN (1,862,773) Co-Citation (2,484,941)	40	942.1
OGBN-Products	Large	2,449,029	100	k-NN (26,939,319) Co-Purchase (126,167,053)	47	668,950

Table 9.3: Clustering results on the small scale datasets.

	ACM				**DBLP**			
	CA	CF1	NMI	ARI	CA	CF1	NMI	ARI
LMVSC	90.33 ±0.0	90.59 ±0.1	68.33 ±0.2	73.64 ±0.1	52.72 ±0.0	58.30 ±0.0	20.15 ±0.0	14.97 ±0.0
OMSC	70.21 ±0.0	71.11 ±0.0	45.81 ±0.0	43.55 ±0.0	91.45 ±0.0	90.95 ±0.0	74.59 ±0.0	79.82 ±0.0
FDAGF	70.27 ±0.0	79.39 ±0.0	47.07 ±0.1	45.20 ±0.1	89.61 ±0.0	89.39 ±0.0	71.56 ±0.0	75.78 ±0.0
SMVSC	70.45 ±0.0	71.80 ±0.0	46.35 ±0.0	43.91 ±0.0	91.45 ±0.0	90.87 ±0.0	74.50 ±0.0	79.88 ±0.0
MvAGC	83.35 ±4.9	83.49 ±5.0	55.86 ±7.1	58.78 ±8.6	76.62 ±6.4	67.10 ±3.5	59.99 ±10.5	59.87 ±6.5
MAGC	55.81 ±7.8	47.58 ±4.1	26.09 ±12.6	20.05 ±16.6	91.93 ±0.6	91.43 ±0.6	75.28 ±1.1	80.53 ±1.4
MCGC	89.92 ±0.0	89.93 ±0.0	67.03 ±0.0	72.56 ±0.0	87.63 ±0.0	77.24 ±9.9	69.37 ±2.0	79.24 ±7.9
LMGEC	93.00 ±0.0	93.06 ±0.0	**75.13 ±0.1**	80.27 ±0.1	92.85 ±0.0	92.37 ±0.0	77.40 ±0.0	82.84 ±0.0
MvSCK	**93.21 ±0.1**	**93.22 ±0.1**	74.97 ±0.1	**80.76 ±0.1**	**93.09 ±0.1**	**92.57 ±0.1**	**78.05 ±0.2**	**83.36 ±0.2**

Table 9.4: Clustering results on the medium scale datasets.

	Amazon Photos				**Amazon Computers**			
	CA	CF1	NMI	ARI	CA	CF1	NMI	ARI
LMVSC	61.23 ±1.2	55.87 ±1.4	57.17 ±0.7	36.60 ±0.7	64.97 ±0.9	**62.24 ±3.4**	52.39 ±0.9	**48.76 ±0.1**
OMSC	61.44 ±0.0	N/A	49.28 ±0.0	36.73 ±0.0	63.99 ±0.0	N/A	42.27 ±0.0	42.89 ±0.0
FDAGF	55.48 ±0.1	49.37 ±0.0	52.96 ±0.1	30.01 ±0.1	64.68 ±4.1	56.93 ±0.8	52.82 ±0.4	44.94 ±2.6
SMVSC	61.67 ±0.1	55.74 ±0.3	50.83 ±0.2	35.99 ±0.1	64.72 ±0.1	49.55 ±0.6	49.09 ±0.4	46.01 ±0.1
MvAGC	45.28 ±6.7	43.15 ±4.4	29.96 ±5.9	20.02 ±3.9	45.82 ±4.1	45.12 ±4.8	28.45 ±5.5	18.98 ±3.1
MAGC	52.36 ±2.7	43.76 ±3.5	48.86 ±2.5	21.42 ±2.3	63.48 ±6.1	51.81 ±10.2	52.47 ±10.1	35.48 ±9.4
MCGC	41.86 ±3.5	26.75 ±7.7	15.16 ±10.6	23.61 ±9.4	45.29 ±0.7	34.17 ±8.2	22.24 ±7.4	26.96 ±10.4
LMGEC	70.93 ±0.0	64.83 ±0.0	60.88 ±0.0	51.05 ±0.0	46.10 ±1.5	40.21 ±0.8	46.14 ±0.9	27.61 ±0.4
MvSCK	**75.66 ±2.2**	**66.56 ±5.0**	**70.66 ±1.3**	**58.65 ±2.0**	**66.34 ±1.5**	56.23 ±0.8	**57.15 ±0.8**	46.27 ±2.0

Table 9.5: Clustering results on the large scale datasets. We allow for a maximum runtime of two hours per run otherwise we report a timeout.

	OGBN-ArXiv				**OGBN-Products**			
	CA	CF1	NMI	ARI	CA	CF1	NMI	ARI
LMVSC	30.94 ±0.6	23.51 ±4.6	31.92 ±1.0	20.07 ±0.5	Timeout			
OMSC	Timeout				Timeout			
FDAGF	Timeout				Timeout			
SMVSC	Timeout				Timeout			
MvAGC	Out of Memory				Out of Memory			
MAGC	Out of Memory				Out of Memory			
MCGC	Timeout				Timeout			
LMGEC	29.61 ±0.8	16.44 ±0.7	34.53 ±0.3	19.86 ±0.7	36.09 ±1.1	16.23 ±0.7	43.64 ±0.7	21.60 ±0.5
MvSCK	**48.29** ±1.8	**28.46** ±0.9	**46.99** ±0.5	**39.15** ±1.2	**47.14** ±1.5	**16.89** ±1.3	**50.80** ±0.8	**32.24** ±1.3

Table 9.6: Execution times on the different datasets (in seconds).

	ACM	**DBLP**	**Amazon Photos**	**Amazon Computers**	**OGBN-ArXiv**	**OGBN-Products**
LMVSC	2.36 ±0.1	2.97 ±0.1	37.78 ±3.7	144.26 ±14.2	2952.97 ±210.43	Timeout
OMSC	24.66 ±0.2	27.51 ±0.1	288.32 ±43.1	414.94 ±3.7	Timeout	Timeout
FDAGF	37.31 ±1.3	68.28 ±3.2	382.76 ±4.2	1555.28 ±50.6	Timeout	Timeout
SMVSC	16.94 ±1.6	29.68 ±0.6	169.55 ±14.5	352.40 ±95.5	Timeout	Timeout
MvAGC	5.05 ±0.1	5.65 ±0.0	36.59 ±1.0	36.76 ±0.7	Out of Memory	Out of Memory
MAGC	21.83 ±1.6	38.12 ±3.8	304.23 ±22.6	8699.82 ±16343.2	Out of Memory	Out of Memory
MCGC	738.54 ±4.5	358.56 ±13.5	3803.44 ±434.1	11393.04 ±1690.1	Timeout	Timeout
LMGEC	0.63 ±0.4	0.51 ±0.5	2.95 ±0.3	7.80 ±1.0	44.87 ±1.4	601.32 ±29.2
MvSCK	**0.18** ±0.1	**0.19** ±0.1	**0.86** ±0.1	**5.96** ±0.2	**20.43** ±3.8	**239.32** ±14.2

Table 9.7: MvSCK results and running times (in seconds) using different kernels.

	ACM		**DBLP**		**Photos**		**Computers**		**ArXiv**		**Products**	
Kernel	CA	Time	CA	Time	CA	Time	CA	Time	CA	Time	CA	Time
Quadratic	**93.2**	**0.2**	**93.1**	**0.2**	75.7	**0.9**	**66.3**	**6.0**	48.3	20.4	**47.1**	239.3
RBF	**93.2**	0.8	**93.1**	2.0	**76.2**	2.4	65.0	6.5	**48.7**	**15.8**	18.6	325.2
Sigmoid	**93.2**	0.9	93.0	1.9	74.7	2.3	65.6	6.4	34.0	16.6	33.2	**231.3**

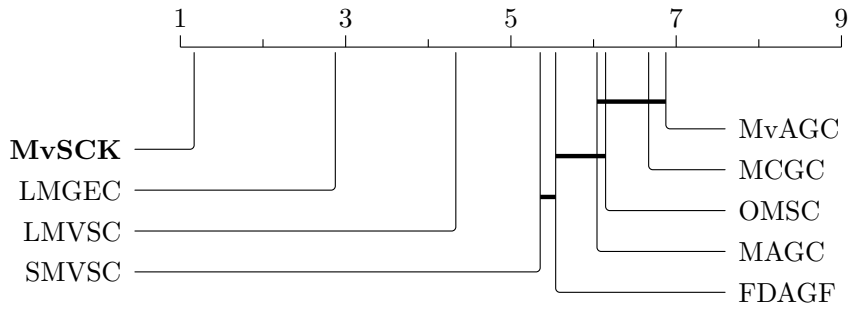


Figure 9.2: Holm post-hoc mean rank test ($\alpha = 0.01$) with respect to clustering performance.

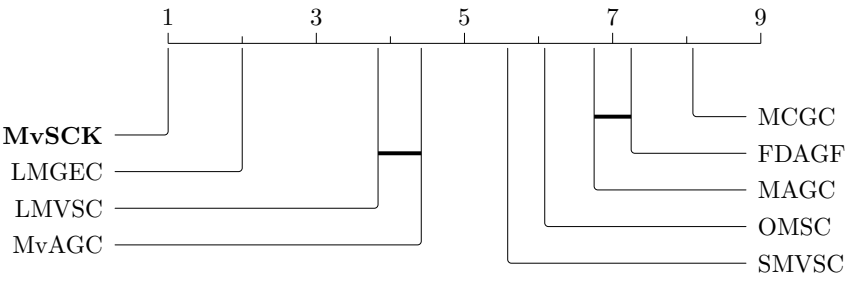


Figure 9.3: Holm post-hoc mean rank test ($\alpha = 0.01$) with respect to running times.

6. Conclusion

In conclusion, this work introduced a novel scalable framework for multi-view subspace clustering through leveraging kernel feature maps for the efficient computation of the consensus subspace affinity graph. The proposed approach results in performance gains and significant speedup thanks to the properties inherent to kernel summation. Extensive experiments on real-world benchmark networks, whatever the degree of overlapping, demonstrate the algorithm's superiority over state-of-the-art methods especially in the context of very large scale network datasets where most other models failed to scale despite using the same computational resources.

With the next chapter we will explore the case of text clustering through the use of large language models, specifically, sentence embedding models. We will show how it is possible to obtain better categorization performance by augmenting those embeddings with semantic graphs.

Chapter 10

Unsupervised Semantic Graph Smoothing for Text Categorization

Objective

This chapter is based on [Fettal, 2024b] where we aimed to create semantically smooth representations of text embeddings in an unsupervised manner. The goal is improving results in text categorization tasks.

Contents

1. Introduction

Text categorization, also known as document categorization, is a natural language processing (NLP) task that involves arranging texts into coherent groups based on their content. It has many applications such as spam detection [Jindal, 2007], sentiment analysis [Melville, 2009], content recommendation [Pazzani, 2007], etc. There are two main approaches to text categorization: classification (supervised learning) and clustering (unsupervised learning). In text classification, the process involves training a model using a labeled dataset, where each document is associated with a specific category. The model learns patterns and relationships between the text features and the corresponding categories during the training phase. Text clustering, however, aims to group similar documents together without prior knowledge of their categories. Unlike text classification, clustering does not require labeled data. Instead, it focuses on finding inherent patterns and similarities in the text data to create clusters.

In the field of NLP, pretrained models have attained state-of-the-art performances in a variety of tasks [Devlin, 2019; Liu, 2019; Reimers, 2019], one of which is text classification. In spite of that, text clustering using such models did not garner significant attention. To this day most text clustering techniques use the representations of texts generated by some pretrained model such as Sentence-BERT [Reimers, 2019] and often use classical clustering approaches such as k-means to obtain a partition of the texts. This is done without any fine-tuning due to the unsupervised nature of the clustering problem.

Recently, graph filtering has appeared as an efficient and effective technique for learning representations for attributed network nodes. The effectiveness of this technique has made it a backbone for popular deep learning architectures for graphs such as the graph convolutional network (GCN) [Kipf, 2017]. Simplified versions of this deep architecture have been proposed wherein the learning of large sets of weights has been deemed unnecessary. Their representation learning scheme works similar to Laplacian smoothing and, by extension, graph filtering. We can give as examples of these simplified techniques the simple graph convolution (SGC) [Wu, 2019], and the simple spectral graph convolution (S^2GC) [Zhu, 2021]. Some researchers used GCNs for the task of text classification. Yao et al. [Yao, 2019] proposed TextGCN which is GCN with a custom adjacency matrix built from word PMI and the TF-IDF of the documents with the attributes being word count vectors. Lin et al. [Lin, 2021a] proposed BertGCN which is similar to TextGCN with the difference being that they use BERT representation for the GCN and combine their training losses. The issue is that these approaches are not suitable for learning unsupervised representations since labels are needed, this is a significant limitation towards their use in unsupervised tasks.

In this work, we propose to use the concept of graph smoothing/filtering, which is the main component accredited with the success of GCNs [Defferrard, 2016; Kipf, 2017; Li, 2018a], to semantically "fine-tune" the representations obtained via sentence embedding models to help traditional clustering (and classification) algorithms better distinguish between semantically different texts and group together texts which have similar meanings, all in an unsupervised manner. To do this, we build a graph with respect to the text which describes the

semantic similarity between the different documents based on the popular cosine similarity measure. Our approach yields almost systematic improvement when using filtering on the textual representations as opposed to using them without filtering in both facets of document categorization: classification and clustering. Experiments on eight popular benchmark datasets support these observations.

2. Graph Smoothing & Filtering

Graph Signal Processing [Shuman, 2013; Ortega, 2018] provides a framework to analyze and process signals defined on graphs, by extending traditional signal processing concepts and tools to the graph domain. This allows for the representation and manipulation of signals in a way that is tailored to the specific structure of the graph. In what follows we refer to matrices in boldface uppercase and vectors in boldface lowercase.

Graph Signals Graph signals are mappings from the set of vertices to the real numbers. A graph signal for a given graph $\mathcal{G}$ can be represented using vector $\mathbf{f} = [f(v_1), \ldots, f(v_n)]$ such that $f : \mathcal{V} \to \mathbb{R}$ is a real-valued function on the vertex set. The smoothness of a signal $\mathbf{f}$ over graph $\mathcal{G}$ can be characterized using the Laplacian quadratic form associated with Laplacian $\mathbf{L}$:

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j} a_{ij}(\mathbf{f}_i - \mathbf{f}_j)^2. \tag{10.1}$$

These signals can be high dimensional and can represent many kinds of data. In our case, signals will represent text embeddings.

Graph Filters Smoother graph signals can be obtained by minimizing the quantity described in formula (10.1). That is the goal of graph filters and the filtering is generally done from a spectral perspective. A specific class of filters that additionally has an intuitive interpretation from a vertex perspective is that of polynomial filters. When the filter is a P-th order polynomial of the form $\hat{h}(\mathbf{L}) = \sum_{m=0}^{p} \theta_m \mathbf{L}^m$, the filtered signal at vertex i, is a linear combination of the components of the input signal at vertices within a P-hop local neighborhood of vertex i:

$$\mathbf{f}_{out}[i] = \alpha_{ii}\mathbf{f}_{in}[i] + \sum_{j \in N(i,p)} \alpha_{ij}\mathbf{f}_{in}[j] \tag{10.2}$$

where $N(i, p)$ is the P-th order neighborhood of vertex i. It is possible to then make the correspondence with a polynomial filter (from a spectral perspective) as follows:

$$\alpha_{ij} = \sum_{m=d_{\mathcal{G}}(i,j)}^{p} \theta_m (\mathbf{L}^m)_{ij} \tag{10.3}$$

where $d_{\mathcal{G}}$ is the shortest distance between node i and j. Several polynomial filters have been proposed in the literature such as the ones associated with SGC [Wu, 2019], S^{2}GC [Zhu,

Table 10.1: The propagation rules associated with the different polynomial filters. $\mathbf{H}^{(0)}$ is the $\mathbf{X}$. P is the propagation order. α and T are filter-specific hyperparameters.

Filter	Propagation Rule
F_{SGC}	$\mathbf{H}^{(p+1)} \leftarrow \mathbf{S}\mathbf{H}^{(p)}$
$F_{\text{S}^2\text{GC}}$	$\mathbf{H}^{(p+1)} \leftarrow \mathbf{H}^{(p)} + \mathbf{S}\mathbf{H}^{(p)}$
F_{APPNP}	$\mathbf{H}^{(p+1)} \leftarrow (1-\alpha)\mathbf{S}\mathbf{H}^{(p)} + \alpha\mathbf{H}^{(0)}$
F_{DGC}	$\mathbf{H}^{(p+1)} \leftarrow (1-\frac{T}{P})\mathbf{H}^{(p)} + \frac{T}{P}\mathbf{S}\mathbf{H}^{(p)}$

2021], APPNP [Gasteiger, 2018] and DGC [Wang, 2021b].

3. Smoothing Sentence Embeddings

In this work, we theorize that smoothing sentence embeddings with a semantic similarity graph can help supervised and unsupervised categorization models better differentiate between the similar and dissimilar documents, leading to performance gains. A common choice for quantifying semantic similarity in the context of text is the cosine similarity. Given two sentence embedding vectors $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$, it is computed as:

$$\cos(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^\top . \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}. \tag{10.4}$$

We build a k-nearest neighbors connectivity graph which we denote $\mathcal{G}$ based on this similarity measure i.e. a graph for which each node has exactly k neighbors and whose edge weights are all equal to one. We characterize the graph $\mathcal{G}$ using its adjacency matrix $\mathbf{A}$, we denote its Laplacian as $\mathbf{L}$. Given a the adjacency matrix, a standard trick to obtain better node representations consists in adding a self-loop

$$\hat{\mathbf{A}} = \mathbf{A} + \lambda \mathbf{I} \tag{10.5}$$

where λ is a hyperparameter controlling the number of self-loops. As such in what follows we consider the symmetrically normalized version of $\hat{\mathbf{A}}$, that is

$$\mathbf{S} = \hat{\mathbf{D}}^{-1/2} \hat{\mathbf{A}} \hat{\mathbf{D}}^{-1/2}. \tag{10.6}$$

Now given a node embedding matrix $\mathbf{X}$ and the previous semantic similarity graph. We consider four polynomial graph filters whose propagation rules we describe in the table 10.1.

4. Experiments

In this section we evaluate our semantically smoothed representations obtained through four filters on two tasks, clustering and classification, with respect to the original representations obtained from SentenceBERT, as well as BERT and RoBERTa based baselines.

Table 10.2: Summary statistics of the datasets. Balance refers to the ratio of the most frequent class over the least frequent class.

Dataset	Documents	Classes	Balance
20 Newsgroup	18,846	20	1.6
DBpedia	12,000	14	1.1
AG News	8,000	4	1.1
BBC News	2,225	5	1.3
Classic3	3,891	3	1.4
Classic4	7,095	4	3.9
R8	7,674	8	76.9
Ohsumed	7,400	23	61.8

Table 10.3: Clustering results in terms of AMI and ARI on the eight datasets.

	AMI	ARI	AMI	ARI	AMI	ARI	AMI	ARI
	20 Newsgroups		AG News		BBC News		Classic3	
$\text{ENS}_{\text{BERT-base}}$	37.5 ±2.5	15.3 ±1.7	54.1 ±3.6	51.4 ±5.8	81.0 ±5.5	80.0 ±8.5	98.6 ±0.1	99.4 ±0.0
$\text{ENS}_{\text{BERT-large}}$	46.1 ±0.7	21.4 ±0.6	58.5 ±2.8	58.2 ±5.9	86.0 ±3.5	86.5 ±6.3	98.4 ±0.2	99.3 ±0.1
$\text{ENS}_{\text{RoBERTa-base}}$	37.5 ±1.4	15.9 ±1.8	55.9 ±4.1	52.1 ±4.1	80.0 ±5.3	77.2 ±9.4	98.4 ±0.1	99.3 ±0.1
$\text{ENS}_{\text{RoBERTa-large}}$	48.0 ±0.8	23.2 ±1.2	56.7 ±4.6	52.8 ±5.1	85.8 ±3.8	85.1 ±7.2	98.7 ±0.1	99.4 ±0.1
SBERT+kM	62.9 ±0.3	47.4 ±1.0	57.9 ±0.1	60.5 ±0.1	90.8 ±0.2	93.0 ±0.1	96.0 ±0.1	97.6 ±0.1
SB+F_{SGC}+kM	65.4 ±0.4	49.1 ±1.1	**60.6 ±0.1**	62.4 ±0.3	90.6 ±0.1	92.9 ±0.1	98.8 ±0.0	99.5 ±0.0
SB+$F_{\text{S}^2\text{GC}}$+kM	64.9 ±0.4	49.0 ±1.1	60.1 ±0.2	62.2 ±0.2	**90.9 ±0.1**	**93.1 ±0.1**	98.3 ±0.0	99.2 ±0.0
SB+F_{APPNP}+kM	65.4 ±0.4	**49.8 ±1.2**	60.6 ±0.0	**62.5 ±0.0**	90.6 ±0.1	92.9 ±0.1	98.5 ±0.0	99.3 ±0.0
SB+F_{DGC}+kM	**65.6 ±0.7**	48.8 ±1.0	60.5 ±1.5	60.5 ±2.2	90.2 ±0.1	92.5 ±0.1	**99.1 ±0.0**	**99.6 ±0.0**
	Classic4		DBpedia		Ohsumed		R8	
$\text{ENS}_{\text{BERT-base}}$	71.4 ±3.5	49.0 ±4.0	73.4 ±2.5	51.0 ±4.0	15.2 ±1.0	9.1 ±1.2	35.3 ±2.0	22.7 ±2.4
$\text{ENS}_{\text{BERT-large}}$	73.0 ±1.8	51.1 ±3.2	72.4 ±2.1	47.2 ±4.2	16.1 ±0.9	9.3 ±0.7	35.7 ±3.5	22.8 ±3.1
$\text{ENS}_{\text{RoBERTa-base}}$	72.1 ±4.7	51.0 ±4.1	74.2 ±2.6	52.5 ±4.7	17.5 ±0.7	11.4 ±0.8	25.6 ±1.0	13.6 ±1.2
$\text{ENS}_{\text{RoBERTa-large}}$	74.1 ±3.5	52.5 ±3.9	72.5 ±2.5	49.0 ±4.4	19.4 ±0.7	12.7 ±0.7	42.4 ±5.6	32.9 ±9.2
SBERT+kM	84.5 ±0.1	86.2 ±0.1	86.0 ±1.4	80.0 ±3.1	39.3 ±0.7	23.5 ±1.2	63.1 ±1.8	45.5 ±3.7
SB+F_{SGC}+kM	85.8 ±2.8	85.6 ±7.4	85.6 ±1.0	78.5 ±2.7	**41.8 ±0.5**	**25.2 ±1.0**	**65.6 ±0.5**	49.0 ±0.6
SB+$F_{\text{S}^2\text{GC}}$+kM	86.0 ±0.0	86.9 ±0.0	**86.6 ±1.2**	**80.4 ±2.8**	41.0 ±0.8	24.5 ±1.5	64.8 ±1.1	47.8 ±0.7
SB+F_{APPNP}+kM	86.2 ±0.0	87.0 ±0.0	85.8 ±1.0	78.9 ±2.7	41.6 ±0.7	24.9 ±1.5	65.1 ±1.6	48.5 ±1.0
SB+F_{DGC}+kM	**86.9 ±0.0**	**87.7 ±0.0**	85.4±1.0	78.4 ±2.2	**41.8 ±0.7**	24.8±1.7	**65.6 ±0.5**	**49.3 ±0.4**

4.1. Datasets and Metrics

We use eight benchmark datasets of varying sizes and number of clusters, and we report their summary statistics in table 10.2. For the metrics, in the supervised context, we use the F1 score as the quality metric while in the unsupervised context we use the adjusted rand index (ARI) [Hubert, 1985] and the adjusted mutual information (AMI) [Vinh, 2009].

4.2. Experimental Settings

For the classification task, we use a random stratified 64%-16%-20% train-val-test split. We also tune the hyperparameters k of the k-nn graph, order of propagation P, the parameter λ and the filter specfic parameters α and T. For the clustering task, we use $k = 10$ for the

Table 10.4: Classification results in terms of F1 score on the eight data sets.

	20News	R8	AGNews	BBCNews	Classic3	Classic4	DBpedia	Ohsumed
$\text{BERT}_{\text{base}}$	80.7	89.94	**89.78**	95.51	**100.0**	**98.58**	97.84	56.48
$\text{RoBERTa}_{\text{base}}$	85.48	89.42	88.06	96.73	99.16	96.47	**98.22**	58.11
SBERT+LR	83.35	90.22	86.25	98.62	99.61	98.19	97.33	62.87
SB+F_{APPNP}+LR	**87.54**	90.9	87.9	**99.06**	99.75	98.36	97.14	**67.6**
SB+F_{DGC}+LR	87.11	90.08	87.59	98.19	99.61	98.52	97.38	67.09
SB+$F_{\text{S}^2\text{GC}}$+LR	87.36	**91.19**	88.33	98.62	99.75	98.19	97.26	67.42
SB+F_{SGC}+LR	87.26	89.22	88.05	**99.06**	99.61	98.32	97.01	67.05

k-nn graph, set $P = 2$ as the propagation order, $\lambda = 1$, $\alpha = .1$ and $T = 5$. We report the averages of the metrics as well as their standard deviations over 10 runs (for the classification task, we omit standard deviation due to them being insignificant).

4.3. Experimental Results

Clustering Results We compare the results of the k-means algorithm (kM) applied on Sentence-BERT (we refer to it as SBERT or SB) embeddings with and without the different filtering operations. In addition to this, we add a baseline which uses an ensemble technique [Ait-Saada, 2021] on the layer outputs of the word embedding of BERT and RoBERTa, this method improves over considering a single layer or taking the mean. We report the clustering results in table 10.3. The filtering operation systematically leads to better results on the benchmark with respect to the filterless clustering scheme on all datasets we have used. These increases are significant in most cases. It also significantly beats the ensemble approach on most datasets.

Classification Results Similar to the clustering setting, we compare results from a Logistic Regression (LR) applied on the original sentence embeddings with and without the filtering operation we introduced. We also use fine-tuned BERT and RoBERTa as baselines (we use the base versions, we did not use the large versions due to computational restrictions). We report the results in table 10.4. We see that this operation leads to better performances on the classification task on the majority of the datasets with respect to the filterless Sentence-BERT but this performance increase is not as pronounced as for the clustering task. We also see that the representations we learn lead to competitive results with respect to BERT and RoBERTa.

Statistical Significance Testing Using the Bonferroni-Dunn post-hoc mean rank test, we analyze the average ranks of the clustering and classification over the Sentence-BERT representations with and without filtering in terms of AMI and ARI, for the clustering task, as well as the F1 score for the classification task on the eight datasets. Figure 10.1 shows that the clustering and classification results when using the proposed semantically smoothed representations are statistically similar and that they all outperform the Sentence-BERT

variant with no filtering in a statistically significant manner with a confidence level of 95%.

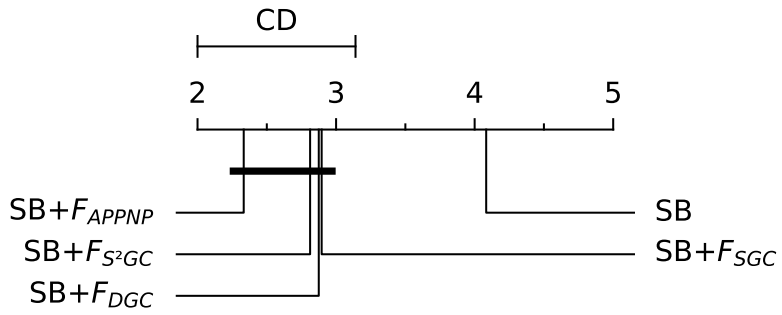


Figure 10.1: Bonferroni-Dunn mean rank test ($\alpha = 0.05$).

5. Conclusion

We proposed a simple empirical approach that consists in using similarity graphs in an unsupervised manner to smooth sentence embeddings obtained from pretrained models in a semantically aware manner. The systematic improvements in performance on both clustering and classification tasks on several benchmark datasets of different scales and balance underscore the effectiveness of using semantic graph smoothing to improve sentence representations.

In the next chapter, we discuss one of the many industrial use cases undertaken within Informatique CDC. We will demonstrate how we used some of the concepts developed throughout this thesis to augment the tabular classification models in order to detect frauds in the MCF system.

Chapter 11

Graph Filtering for Fraud Detection in *Mon Compte Formation*

Objective

In this chapter, we try to improve classification performance on the "Mon Compte Formation" data using different graph filtering techniques from the original non-graph data based on the approaches presented in the previous chapters.

Contents

1. Introduction

In this chapter we will show how we leveraged attributed-graphs for the industrial use case of fraud detection in the *Mon Compte Formation*. We first start by introducing the MCF and the type of frauds that plagues it. We then present the project that *Caisse des Depôts* implemented in order to address this problem and how we augmented the available data using graphs in order to improve the performance of classifiers.

1.1. *Mon Compte Formation*

Mon Compte Formation[1] (MCF) is a public scheme that allows each active person to help fund their professional project. Throughout their working life, they accumulate training rights, recorded in euros or in hours in their MCF, which can be consulted on the website and the mobile application *Mon Compte Formation*. On this portal, they have access to the catalog of eligible professional training courses for which they can register. *Caisse des Dépôts* manages the scheme on behalf of the Ministry of Labour, Employment and Integration. The purchase of a training offer should enable people to acquire new skills useful for their jobs or give life to a professional reconversion project. This public service is therefore more valuable than ever to help people progress in their career and adapt to changes in the labour market [2].

1.2. Fraud in *Mon Compte Formation*

Fraud from training organisms against the MCF is a lingering problem that has cost millions of euros to the french government. Two types of fraud can be identified:

Fraud against the MCF users Here, training organisms try to directly access funds from the users. Advertisements offering gifts in exchange for enrolment in a training offer (such as a tablet, computer or telephone) or for the recovery of part of the balance in cash, are contrary to the spirit of this law and are illegal. For example, more than 2,600 formal notices have already been issued, more than 150 organisations have been excluded from the platform and nearly 30 criminal complaints have been filed. Organisations or individuals offering this type of deals are exposed to very serious sanctions: exclusion from the catalog for the training organisation, administrative sanctions, restitution of earnings and criminal sanctions.

Calls, SMS, emails, or on social networks, fraudsters use the phishing technique as well as aggressive canvassing. The practice of commercial canvassing is not currently prohibited in France. However, it can be abusive or fraudulent if the person on the other end of the line insists that you buy a course or offers you a benefit other than the educational contribution

[1] https://www.moncompteformation.gouv.fr/

[2] https://travail-emploi.gouv.fr/formation-professionnelle/droit-a-la-formation-et-orientation-professionnelle/compte-personnel-formation

(for example, material goods or money) in exchange for enrolling in a course [3].

Fraud against the MCF service The other type of fraud is when a training organism falsely declares that a trainee has fully or partially (as a percentage) followed a course and thus requests reimbursement for the course fee from the MCF fund. A team inside *Caisse des Depôts* is responsible for checking that the declarations made by the training bodies correspond to reality. However, due to the large number of payment requests that the CDC receives, it is impossible to perform a check on each individual request.

1.3. Detecting Frauds through Machine Learning

Caisse des Dépôts launched a machine learning project where the aim is to detect training organisations that commit the second type of fraud, i.e., fraud against the MCF service. The idea of this project is to find payment requests that look suspect and might be fraudulent, this will allow the team responsible for checking these requests to have a higher rate of true positives, thus avoiding checking valid requests and wasting the agents' time.

2. Data Description

Since the MCF data contains sensitive information, we will only be presenting general descriptive statistics of the dataset.. The dataset contains 42,230 payment requests in total and each payment request is described with more than 90 features. After uninformative features and one-hot encoding categorical features and add some engineered features, we obtain exactly 73 features.

The task at hand is that of binary classification, where the only information we can give about the target is that the corresponding payment request is either considered 'suspect' or 'not suspect', they represent 56% and 44% of the data, respectively. We consider that we are in a supervised setting and we perform an 80%/20% training-test stratified split of the data set.

3. Initial Model Selection

We start by selecting an initial model from a set of commonly used classification models. During the project, several models were considered, but for the sake of succinctness we will only be presenting those that performed significantly better than the remaining ones, namely, those based on Decision Trees, which are known to be the best performing algorithms on tabular data as well as for their interpretability.

[3] https://travail-emploi.gouv.fr/actualites/l-actualite-du-ministere/article/fraude-au-MCF-prenez-garde-aux-faux-bons-plans-379534

3.1. Decision Trees

A decision tree is a structure that resembles a flowchart where each inner node represents a test for an attribute, each branch represents the result of the test, and each leaf node represents a Class label. A path from root to leaf represents a classification rule. In machine learning, decision tree learning is the process of learning a decision tree in a supervised manner. Here, since we are interested in predicting categorical variables, we will be dealing with classification trees. Notable decision tree algorithms include ID3 [Quinlan, 1986], C4.5 [Quinlan, 1993], CART [Breiman, 1984], etc.

For instance, the popular machine learning toolkit Scikit-Learn [Pedregosa, 2011] uses the CART which stands for Classification And Regression Trees. CART constructs binary trees using the feature and threshold that yield that minimizes impurity at each node. A simple pseudo-code is provided in algorithm 11.

Algorithm 11: `Classification And Regression Trees (CART)`

Input : D: dataset of pairs $\{(x_i, y_i)\}_{i=1}^n$,
H: impurity function.
Output: T: binary decision tree.
Procedure `TreeGrowing` (T: tree, m: current node, Q: current dataset)
  if $|Q|$ *is pure* **then**
    Make m a leaf node by setting $T[m] \leftarrow$ `NULL`;
    return;
  end
  foreach *candidate split* $\theta \leftarrow (j, t)$ **do**
    Partition Q into left and right sets:

$$Q_\theta^- \leftarrow \{(x, y) \in Q \mid x_j \leq t\} \quad \text{and} \quad Q_\theta^+ \leftarrow Q \setminus Q_\theta^-;$$

    Compute the impurity incurred from the split:

$$G_\theta(Q) \leftarrow \frac{\left|Q_\theta^-\right|}{|Q|} H(Q_\theta^-) \; + \; \frac{\left|Q_\theta^+\right|}{|Q|} H(Q_\theta^+);$$

  end
  Save the split $T[m] \leftarrow \theta^*$ with the lowest incurred impurity where $\theta^* \leftarrow \arg\min_\theta G_\theta(Q)$;
  Grow the left subtree of m, `TreeGrowing` (T, m^-, $Q_{\theta^*}^-$);
  Grow the right subtree of m, `TreeGrowing` (T, m^+, $Q_{\theta^*}^+$);
begin
  Initialize the binary decision tree T;
  Grow the tree starting from the root `TreeGrowing` (T, 0, D);
end

3.2. Boosted Trees

Boosting [Freund, 1996] is a supervised ensemble technique where a set of weak classifiers are combined in order to obtain a strong classifier. The common idea of boosting algorithms is that each time a weak learner is added, the data weights are readjusted. Misclassified inputs gain a higher weight and inputs that are classified correctly lose weight which makes future weak learners focus more on the previous weak learners' misclassified examples. In most applications, decision Trees are generally used as the weak learners. Notable boosting algorithms include AdaBoost [Freund, 1997], XGBoost [Chen, 2016], etc. For instance, XGBoost, from *eXtreme Gradient Boosting*, works as Newton-Raphson in function space and has been the algorithm of choice of many teams in machine learning competitions. A generic pseudo-code for XGBoost is provided in algorithm 12.

Algorithm 12: `eXtreme Gradient Boosting (XGBoost)`

Input : D: dataset of pairs $\{(x_i, y_i)\}_{i=1}^{n}$,
L: loss function,
Φ: set of weak learners,
η: learning rate.

Output: F_M: boosted model.

Initialize model with a constant $F_0 \leftarrow \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$;

for $m \in \{1, \dots, M\}$ **do**

Compute the first-order pseudo-residuals:

$$\forall i \in \{1, \dots, n\}, \quad g_{im} \leftarrow \frac{\partial L(F(x_i), y_i)}{\partial F}|_{F=F_{m-1}};$$

Compute the second-order pseudo-residuals:

$$\forall i \in \{1, \dots, n\}, \quad h_{im} \leftarrow \frac{\partial^2 L(F(x_i), y_i)}{\partial F^2}|_{F=F_{m-1}};$$

Fit a weak learner on the dataset of pairs $\left\{\left(x_i, -\frac{g_{im}}{h_{im}}\right)\right\}_{i=1}^{n}$ by solving:

$$f_m \leftarrow \arg\min_{f \in \Phi} \sum_{i=1}^{n} h_{im} \left[-\frac{g_{im}}{h_{im}} - f(x_i)\right]^2;$$

Perform a functional gradient descent update:

$$F_m(x) \leftarrow F_{m-1}(x) + \eta f_m(x);$$

end

3.3. Bagged Trees

Bootstrap aggregating [Breiman, 1996] (or bagging for short) is a supervised ensemble technique where learners are trained on different random subsets of the data. These subsets are sampled with replacement, hence the bootstrapping part in the name. Algorithm 13 shows a pseudo-code for the training of a bagging classifier.

Once these learners are trained, an aggregation of the different predictions for the same data points is performed. This aggregation generally consists in taking the statistical mode of the different predictions for each single observation which can be interpreted as a voting process between the different learners.

As with boosting, the most common learner used for bagging is the Decision Tree which led to this kind of approach having its own name which is the Random Forest [Breiman, 2001]. Additional randomness can be introduced by considering other tree bagging approaches such as the Extremely Randomized Trees ensemble [Geurts, 2006] where the splits in the trees are randomized unlike in Random Forests where the best split is deterministic.

Algorithm 13: `Bootstrap Aggregating Classifier`

Input : D: dataset of pairs $\{(x_i, y_i)\}_{i=1}^n$,
L: loss function,
$h_1, \dots, h_M$: classifiers.
Output: F: bagged classifier.

for $m \in \{1, \dots, M\}$ **do**
  Bootstrap a sample D_m from D;
  Fit classifier h_m on D_m;
end
Aggregate classifiers via a voting $F(x) \leftarrow \arg\max_y \mathbb{1}_{h_m(x)=y}$

3.4. Initial Results

We consider three initial models: Decision Trees (CART), XGBoost and Random Forests (CART). We perform a grid search over a set of possible parameters with a 3-fold cross validation. The score used to fine-tune the parameters is the F1 score.

Table 11.1: Cross-validation score over the training set (F1%).

Model	**F1**
Decision Tree	62.96
XGBoost	69.86
Random Forest	**71.42**

The initial results are reported in table 11.1, predictably, the ensemble approaches significantly outperform the single decision tree. The bagging approach, Random Forest, in

turn, outperforms the boosting approach, XGBoost. Consequently, in what follows, we will consider the Random Forest model.

4. Data Augmentation via Graphs

We wish to augment the data by introducing neighborhood information via graphs. We consider three types of augmentations: One with a graph on the rows based on [Fettal, 2022c; Fettal, 2023b]. One with multiple graphs on the rows [Fettal, 2023c]. And finally, one with two graphs, one on the rows and one on the columns [Fettal, 2022d].

In each of these scenarios, we assume that we are in a transductive setting, meaning that we have access to the test set in addition to the training set but no access to the test set labels.

4.1. Augmentation with a single Graph

The k-nearest neighbor (k-nn) graph is a graph in which two vertices v and w are connected if the distance between them is among the k-th smallest distances from v to all other data points. Here we create a k-nn graph based on the euclidean metric with a propagation order p. This augmentation can be formulated as

$$\mathbf{H} \leftarrow \mathbf{S}^p\mathbf{X} \tag{11.1}$$

where $\mathbf{X}$ is the concatenation of both the train and test sets. $\mathbf{S}$ is the propagation matrix formed from the k-nn graph and preprocessed using the procedure from [Fettal, 2022c]. The rows of $\mathbf{H}$ corresponding to the training set are then fed into the random forest model for training. Here, $\mathbf{S}$ is computed as

$$\mathbf{S} \leftarrow \tilde{\mathbf{D}}^{-1}\left((\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} + \alpha\mathbf{I}) + \beta\mathbf{I}\right) \tag{11.2}$$

where $\mathbf{A}$ is the adjacency matrix created from the k-nn graph $\mathbf{M}$ via $\mathbf{A} = (\mathbf{M} + \mathbf{M}^\top)/2$. $\mathbf{D}$ is the degree matrix of $\mathbf{A}$ and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} + \alpha\mathbf{I}$. The two scalars α and β are hyperparameters.

4.2. Augmentation with multiple Graphs

Same as for the previous augmentation type, we create three k-nn graphs. The first one is based on the euclidean (or l2) metric, the second one is based on the cosine metric and the last one on the Manhattan (or l1) metric. Here, like in [Fettal, 2023c], we only consider the first order neighborhood during propagation. We formulate it as

$$\mathbf{H} \leftarrow \sum_{i=0}^{2} \alpha_i \mathbf{S}_i\mathbf{X} \tag{11.3}$$

Table 11.2: Results of the original Random Forest and the augmented versions on the test set. Results are averaged over five runs.

Model	ACC	F1	NMI	ARI
Random Forest (RF)	72.61±0.13	77.30±0.13	15.25±0.21	20.36±0.24
Single Graph + RF	72.85±0.18	77.64±0.15	**15.74±0.27**	20.79±0.33
Two-way Graph + RF	72.78±0.21	**77.67±0.17**	15.72±0.32	20.66±0.39
Multi-view Graph + RF	**72.86±0.02**	77.56±0.02	15.67±0.02	**20.81±0.03**

with each propagation matrix $\mathbf{S}_i$ being obtained similarly as the one for the augmentation with a single graph.

4.3. Augmentation with a Row Graph and Column Graph

In this setting, similar to what was done in [Fettal, 2022d], we have two k-nn graphs, one created over the rows and which will be based on the euclidean metric, and one created over the columns and created based on the PPMI score. There will be a propagation order parameter p over the rows, while for columns, we will only perform a one-hop propagation. Formally, we have that

$$\mathbf{H} \leftarrow \mathbf{S}_\mathbf{R}^p \mathbf{X} \mathbf{S}_\mathbf{C} \tag{11.4}$$

with each propagation matrix $\mathbf{S}_\mathbf{R}$ and $\mathbf{S}_\mathbf{R}$ being obtained similarly as for the two previous augmentations.

4.4. Results

After performing the hyperparameter tuning via a grid search with 3-fold cross-validation in order to choose the hyperparameters described when presenting the graph augmented models. We obtain the results shown in table 11.2. Reported are the accuracy (ACC), F1 score [Rijsbergen CJ, 1979], normalized mutual information (NMI) [Strehl, 2002], and adjusted rand index (ARI) [Hubert, 1985]. Results are averaged over five runs.

We see that the models result in a small increase in performance over the four metrics. Augmentation with the multi-view graph yielded the best clustering accuracy and the best ARI, augmentation with the two-way graph resulted in the best F1 score, and the augmentation with single graph yielded the best NMI.

Statistical Significance To see if augmentation improves the base model in a statistically significant manner. We perform Student's paired t-test where the alternative hypothesis is that the augmented model has better (greater) perfomance scores. The compared models are the RF and single graph + RF. For each metric; ACC, F1, NMI, and ARI; the p-value is less than 0.05 which means that we can reject the null hypothesis in each of these cases. Consequently, we can conclude that the improvement of the single graph + RF model over

the RF is statistically significant.

5. Conclusion

In this chapter, we presented an industrial use case inside *Caisse des Dépôts* and *Informatique Caisse des Dépôts* for the detection of suspect payment requests made by training organisms. We have shown how we selected an initial model from a range of commonly used machine learning models for tabular data. We then showed how we can augment the available features via augmentation with graphs according to different schemes in a transductive setting in order to obtain statistically significant better results.

Conclusion & Perspectives

The thesis presented approaches for addressing scalability issues in current graph community detection models (and representation learning to some extent) and proposed new methods for clustering (and embedding in some cases) different types of graphs, including classical, bipartite, attributed, bipartite attributed, and multi-view attributed graphs. These approaches leverage techniques such as linear projections, Laplacian smoothing, optimal transport, and kernel methods. We have three key characteristics: simplicity, cost-effectiveness, and few hyper-parameters. Below we describe the main highlights of the different chapters and then we will present some perspectives to consider for the continuation of the work carried out.

In **Chapter 4**, we presented a novel algorithm for graph clustering with arbitrary size constraints through the use of optimal transport. This approach generalizes the concept of normalized and ratio cuts to any notion of size and size distributions. The proposed algorithm was shown to work well when used as a post-processing step in conjunction with classical graph cut algorithms, as demonstrated by experiments on balanced and imbalanced datasets. The results highlighted the effectiveness of our approach in terms of clustering performance and its ability to recover partitions that closely match the desired distribution.

Chapter 3 [Fettal, 2022b; Fettal, 2023a] presented a novel approach for bipartite graph clustering using optimal transport that addresses the computational challenges in clustering sparse data such as document-term matrices. The problem is formulated as a bilinear program which is solved by an efficient block coordinate descent algorithm to find a sparse vertex solution. The results of experiments on various document-term datasets indicate that the proposed method effectively identifies clusters that align with ground truth document classes and produces semantically meaningful partitions for terms. Additionally, the proposed method performs better than recent OT co-clustering techniques and is more computationally efficient.

In **Chapter 5** [Fettal, 2022c; Fettal, 2022a], we have demonstrated the effectiveness of using the simple formulation of the GCN for efficient node embedding and clustering. We proposed a normalization that makes the GCN encoder act as a low pass filter, a novel approach that leverages information from both the GCN embedding reconstruction loss and the cluster structure of the embeddings, and an algorithm which we rigorously studied for its complexity, and show that it performed better than other graph clustering algorithm in a more efficient manner. We also discussed how GCC is related to other methods and compared

it with other unsupervised methods. The experimental results provided strong evidence for the performance and effectiveness of our approach.

In **Chapter 6** [Fettal, 2023b], we presented an efficient algorithm for attributed graph clustering through the process of subspace clustering. The algorithm begins by learning an initial representation of the graph using a simple yet effective neighborhood propagation step. Next, it learns a factored coefficient matrix using orthogonal constraints, which is then embedded into a new feature space to create a symmetric and nonnegative affinity matrix. This affinity matrix is then used in an implicit spectral clustering algorithm. The experimentation conducted showed that our proposal is effective and efficient compared to current state-of-the-art attributed-graph clustering algorithms.

In **Chapter 7** [Fettal, 2022d; Fettal, 2023d], a new approach for attributed bipartite Graph clustering was proposed through the use of co-clustering and bilateral graph convolution. This approach addresses computational and spatial complexity issues by utilizing factor matrices and nonnegative kernel feature maps. The proposed model was shown to have a grouping effect, and the bilateral convolution improves performance even when no ground truth graph is available. Experiments on both synthetic and real datasets demonstrated that this model is competitive with current state-of-the-art methods for text-attributed graph clustering, while also being efficient and robust in the face of uninformative graph structures.

In **Chapter 8** [Fettal, 2023c] we presented a new model that can jointly perform multi-view attributed graph representation learning and multi-view attributed clustering. The proposed model is generic in that it can handle an arbitrary number of graph structures and feature sets. Experiments show that the proposed model is comparable to more complex state-of-the-art models, with even better performance and faster computation times on most benchmarks.

In **Chapter 9** we presented a scalable subspace clustering framework that leverages kernel feature maps for efficiency purposes, outperforming existing methods in clustering performance and scalability on large datasets.

In **Chapter 10** we introduced an approach to unsupervised text representation learning, focusing on semantic coherence. By utilizing graph smoothing, we enhance sentence embeddings from pretrained models, resulting in improved performance for text clustering and classification tasks. The method's efficacy was demonstrated across eight benchmark datasets.

Finally, among the many industrial projects undertaken within Caisse des Dépôts and Informatique Caisse des Dépôts, we have chosen to focus on a use case presented in **Chapter 11**. Thereby, we discussed an industrial application for identifying potentially fraudulent payment requests. The process for selecting an initial model from various commonly used machine learning models for tabular data was outlined, as well as methods for improving the performance of the model through feature augmentation with graphs in a transductive setting. The result was a statistically significant improvement in performance. We do suggest, however, that it would be worthwhile to improve the storage of the user and training programs information as it was challenging dealing with the data as we had to work with

snapshots instead of having access to all the data at all time. Additionally, there were several cases of data inconsistencies that we have found during the data exploration phase which the industrial partners in charge of CPF were not aware of nor could explain . Note that over the course of this PhD, CDC implemented additional measures to fight fraud. In addition to using machine learning approaches for the detection of frauds that have already happened, they added preventive actions through constraints and conditions for user access to the CPF funds such as authentication through FranceConnect+.

Our work raises new research questions and opens up several interesting academic and industrial research directions such as:

Unsupervised Graph Attention Networks. With the GCN, adapting it to learning unsupervised representations can be easily done via graph filtering process which is akin to Laplacian smoothing. However, with the graph attention network (GAT) [Veličković, 2018], it is necessary to learn an attention adjacency matrix which can be challenging in an unsupervised context. This leads to interesting research directions.

Efficient Contrastive Clustering. In this thesis, we proposed a scalability framework for subspace clustering. Another interesting technique for clustering uses contrastive learning which also suffers from complexity issues and which can be prohibitive for a lot of applications.

Generic propagation order selection rule. In most of our contributions, there is a propagation order parameter. While the rule-of-thumbs we proposed resulted in satisfying results, it would be worthwhile to further investigate this aspect and find a theoretically sound way of selecting that hyperparameter.

Semi-supervised and unsupervised Learning for the CPF. We had tested semi-supervised approaches on the CPF project, however, the data we had received was biased due to being selected based on being flagged as suspicious via selection rules made by the CPF staff. As such, an interesting idea to try would be to select a representative sample of the overall data to label and use it for semi-supervised learning. Of course fraudsters will always come up with new ways to exploit the system, so new representative samples must be regularly added to the training regime. As such, the ideal fraud detection system ought to be unsupervised but this will prove to be very challenging.

Bibliography

[Affeldt, 2020] Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. "Spectral clustering via ensemble deep autoencoder learning (SC-EDAE)". *Pattern Recognition* 108 (2020), p. 107522 (cit. on p. ix).

[Affeldt, 2021] Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. "Regularized bi-directional co-clustering". *Statistics and Computing* 31.3 (2021), pp. 1–17 (cit. on pp. 41, 103).

[Affeldt, 2022] Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. "CAEclust: A consensus of autoencoders representations for clustering". *Image Processing On Line* 12 (2022), pp. 590–603 (cit. on p. ix).

[Aggarwal, 2012] Charu C Aggarwal and ChengXiang Zhai. "A survey of text clustering algorithms". *Mining text data* (2012), pp. 77–128 (cit. on p. 101).

[Agrawal, 1998] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. "Automatic subspace clustering of high dimensional data for data mining applications". *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*. 1998, pp. 94–105 (cit. on p. 52).

[Ahalt, 1990] Stanley C Ahalt, Ashok K Krishnamurthy, Prakoon Chen, and Douglas E Melton. "Competitive learning algorithms for vector quantization". *Neural networks* 3.3 (1990), pp. 277–290 (cit. on p. 53).

[Ahmed, 2013] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. "Distributed large-scale natural graph factorization". *Proceedings of the 22nd international conference on World Wide Web*. 2013, pp. 37–48 (cit. on p. 68).

[Ailem, 2016] Melissa Ailem, François Role, and Mohamed Nadif. "Graph modularity maximization as an effective method for co-clustering text data". *Knowledge-Based Systems* 109 (2016), pp. 160–173 (cit. on p. 25).

[Ailem, 2017] Melissa Ailem, François Role, and Mohamed Nadif. "Model-based co-clustering for the effective handling of sparse data". *Pattern Recognition* 72 (2017), pp. 108–122 (cit. on p. 102).

[Ailon, 2012] Nir Ailon, Noa Avigdor-Elgrabli, Edo Liberty, and Anke Van Zuylen. "Improved approximation algorithms for bipartite correlation clustering". *SIAM Journal on Computing* 41.5 (2012), pp. 1110–1121 (cit. on p. 41).

[Ait-Saada, 2021] Mira Ait-Saada, François Role, and Mohamed Nadif. "How to leverage a multi-layered transformer language model for text clustering: an ensemble approach". *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 2837–2841 (cit. on p. 160).

[Allab, 2016] Kais Allab, Lazhar Labiod, and Mohamed Nadif. "A semi-NMF-PCA unified framework for data clustering". *IEEE Transactions on Knowledge and Data Engineering* 29.1 (2016), pp. 2–16 (cit. on p. 68).

[Allab, 2018] Kais Allab, Lazhar Labiod, and Mohamed Nadif. "Simultaneous spectral data embedding and clustering". *IEEE transactions on neural networks and learning systems* 29.12 (2018), pp. 6396–6401 (cit. on p. 68).

[An, 2015] Jinwon An and Sungzoon Cho. "Variational autoencoder based anomaly detection using reconstruction probability". *Special Lecture on IE* 2.1 (2015), pp. 1–18 (cit. on p. 68).

[Anton Tsitsulin, 2020] Bryan Perozzi Anton Tsitsulin John Palowitch and Emmanuel Müller. "Graph Clustering with Graph Neural Networks". *Proceedings of the 16th International Workshop on Mining and Learning with Graphs (MLG)*. 2020 (cit. on pp. 69, 70, 86).

[Asano, 2020] Yuki M. Asano, Christian Rupprecht, and Andrea Vedaldi. "Self-labelling via simultaneous clustering and representation learning". *International Conference on Learning Representations (ICLR)*. 2020 (cit. on p. 68).

[Barber, 2007] Michael J Barber. "Modularity and community detection in bipartite networks". *Physical Review E* 76.6 (2007), p. 066102 (cit. on pp. 25, 34, 40).

[Belkin, 2003] Mikhail Belkin and Partha Niyogi. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation". *Neural Computation* 15.6 (2003), pp. 1373–1396 (cit. on p. 68).

[Bhunia, 2021] Ayan Kumar Bhunia, Pinaki Nath Chowdhury, Aneeshan Sain, Yongxin Yang, Tao Xiang, and Yi-Zhe Song. "More photos are all you need: Semi-supervised learning for fine-grained sketch based image retrieval". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4247–4256 (cit. on p. 52).

[Bhunia, 2020] Ayan Kumar Bhunia, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. "Sketch less for more: On-the-fly fine-grained sketch-based image retrieval". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9779–9788 (cit. on p. 52).

[Boutalbi, 2021] Rafika Boutalbi, Lazhar Labiod, and Mohamed Nadif. "Implicit consensus clustering from multiple graphs". *Data Mining and Knowledge Discovery* 35 (2021), pp. 2313–2340 (cit. on p. 26).

[Breiman, 1984] L Breiman. "Classification and Regression Trees". *The Wadsworth & Brooks/Cole* (1984) (cit. on p. 166).

[Breiman, 1996] Leo Breiman. "Bagging predictors". *Machine learning* 24.2 (1996), pp. 123–140 (cit. on p. 168).

[Breiman, 2001] Leo Breiman. "Random forests". *Machine learning* 45.1 (2001), pp. 5–32 (cit. on p. 168).

[Brohee, 2006] Sylvain Brohee and Jacques Van Helden. "Evaluation of clustering algorithms for protein-protein interaction networks". *BMC bioinformatics* 7 (2006), pp. 1–19 (cit. on pp. viii, 2).

[Cai, 2008] Deng Cai, Xiaofei He, Xiaoyun Wu, and Jiawei Han. "Non-negative matrix factorization on manifold". *2008 eighth IEEE international conference on data mining*. IEEE. 2008, pp. 63–72 (cit. on pp. 44, 119).

[Cai, 2013] Xiao Cai, Feiping Nie, Weidong Cai, and Heng Huang. "New graph structured sparsity model for multi-label image annotations". *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 801–808 (cit. on p. 52).

[Cai, 2020] Yaoming Cai, Zijia Zhang, Zhihua Cai, Xiaobo Liu, Xinwei Jiang, and Qin Yan. "Graph convolutional subspace clustering: A robust subspace clustering framework for hyperspectral image". *IEEE Transactions on Geoscience and Remote Sensing* 59.5 (2020), pp. 4191–4202 (cit. on p. 86).

[Cao, 2013] Jie Cao, Zhiang Wu, Junjie Wu, and Wenjie Liu. "Towards information-theoretic k-means clustering for image indexing". *Signal Processing* 93.7 (2013), pp. 2026–2037 (cit. on p. 52).

[Cao, 2019] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. "Learning imbalanced datasets with label-distribution-aware margin loss". *Advances in neural information processing systems* 32 (2019) (cit. on p. 63).

[Chang, 2017] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. "Deep adaptive image clustering". *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5879–5887 (cit. on p. 52).

[Chen, 2020] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 2020, pp. 3438–3445 (cit. on p. 97).

[Chen, 2022] Man-Sheng Chen, Chang-Dong Wang, Dong Huang, Jian-Huang Lai, and Philip S Yu. "Efficient orthogonal multi-view subspace clustering". *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 127–135 (cit. on pp. 142, 143).

[Chen, 2016] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794 (cit. on p. 167).

[Chen, 2019a] Xiaojun Chen, Renjie Chen, Qingyao Wu, Yixiang Fang, Feiping Nie, and Joshua Zhexue Huang. "LABIN: balanced min cut for large-scale data". *IEEE transactions on neural networks and learning systems* 31.3 (2019), pp. 725–736 (cit. on p. 52).

[Chen, 2017] Xiaojun Chen, Joshua Zhexue Haung, Feiping Nie, Renjie Chen, and Qingyao Wu. "A self-balanced min-cut algorithm for image clustering". *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2061–2069 (cit. on pp. 52, 53).

[Chen, 2019b] Yong Chen, Hui Zhang, Rui Liu, Zhiwen Ye, and Jianying Lin. "Experimental explorations on short text topic mining between LDA and NMF based Schemes". *Knowledge-Based Systems* 163 (2019), pp. 1–13 (cit. on p. 102).

[Cheng, 2018] Qimin Cheng, Qian Zhang, Peng Fu, Conghuan Tu, and Sen Li. "A survey and analysis on automatic image annotation". *Pattern Recognition* 79 (2018), pp. 242–259 (cit. on p. 52).

[Chizat, 2020] Lenaic Chizat, Pierre Roussillon, Flavien Léger, François-Xavier Vialard, and Gabriel Peyré. "Faster wasserstein distance estimation with the sinkhorn divergence". *Advances in Neural Information Processing Systems* 33 (2020), pp. 2257–2269 (cit. on p. 35).

[Chowdhury, 2021] Samir Chowdhury and Tom Needham. "Generalized spectral clustering via Gromov-Wasserstein learning". *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 712–720 (cit. on pp. 57, 62).

[Clanuwat, 2018] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. "Deep learning for classical japanese literature". *arXiv preprint arXiv:1812.01718* (2018) (cit. on p. 62).

[Crouse, 2016] David F Crouse. "On implementing 2D rectangular assignment algorithms". *IEEE Transactions on Aerospace and Electronic Systems* 52.4 (2016), pp. 1679–1696 (cit. on p. 63).

[Cui, 2020] Ganqu Cui, Jie Zhou, Cheng Yang, and Zhiyuan Liu. "Adaptive graph encoder for attributed graph embedding". *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 976–985 (cit. on p. 80).

[Cuturi, 2013] Marco Cuturi. "Sinkhorn distances: Lightspeed computation of optimal transport". *Advances in neural information processing systems* 26 (2013) (cit. on pp. 35, 65).

[Davies, 1979] David L Davies and Donald W Bouldin. "A cluster separation measure". *IEEE transactions on pattern analysis and machine intelligence* 2 (1979), pp. 224–227 (cit. on pp. 44, 58, 95).

[De Soete, 1994] Geert De Soete and J Douglas Carroll. "K-means clustering in a low-dimensional Euclidean space". *New approaches in classification and data analysis*. Springer, 1994, pp. 212–219 (cit. on p. 68).

[Defferrard, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering". *Advances in neural information processing systems* 29 (2016), pp. 3844–3852 (cit. on pp. 17, 69, 86, 102, 128, 156).

[Dempster, 1977] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the royal statistical society: series B (methodological)* 39.1 (1977), pp. 1–22 (cit. on pp. ix, 115).

[Demšar, 2006] Janez Demšar. "Statistical comparisons of classifiers over multiple data sets". *The Journal of Machine Learning Research* 7 (2006), pp. 1–30 (cit. on p. 48).

[Deng, 2012] Li Deng. "The mnist database of handwritten digit images for machine learning research". *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142 (cit. on p. 62).

[DeSieno, 1988] Duane DeSieno. "Adding a conscience to competitive learning." *ICNN*. Vol. 1. 1988 (cit. on p. 53).

[Devlin, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186 (cit. on p. 156).

[Dhillon, 2001] Inderjit S Dhillon. "Co-clustering documents and words using bipartite spectral graph partitioning". *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 2001, pp. 269–274 (cit. on pp. 25, 34, 101–103, 118).

[Dhillon, 2004] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. "Kernel k-means: spectral clustering and normalized cuts". *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, pp. 551–556 (cit. on p. 62).

[Dhillon, 2003] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. "Information-theoretic co-clustering". *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003, pp. 89–98 (cit. on p. 34).

[Dhillon, 2007] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. "Weighted Graph Cuts without Eigenvectors A Multilevel Approach". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.11 (2007), pp. 1944–1957 (cit. on p. 70).

[Dittrich, 2008] Marcus T Dittrich, Gunnar W Klau, Andreas Rosenwald, Thomas Dandekar, and Tobias Müller. "Identifying functional modules in protein–protein interaction networks: an integrated exact approach". *Bioinformatics* 24.13 (2008), pp. i223–i231 (cit. on pp. viii, 2).

[Dolnicar, 2012] Sara Dolnicar, Sebastian Kaiser, Katie Lazarevski, and Friedrich Leisch. "Biclustering: Overcoming data dimensionality problems in market segmentation". *Journal of Travel Research* 51.1 (2012), pp. 41–49 (cit. on p. 34).

[Drineas, 2005] Petros Drineas, Michael W Mahoney, and Nello Cristianini. "On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning." *journal of machine learning research* 6.12 (2005) (cit. on p. 114).

[Eisen, 1998] Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. "Cluster analysis and display of genome-wide expression patterns". *Proceedings of the National Academy of Sciences* 95.25 (1998), pp. 14863–14868 (cit. on p. 34).

[Elhamifar, 2013] Ehsan Elhamifar and René Vidal. "Sparse subspace clustering: Algorithm, theory, and applications". *IEEE transactions on pattern analysis and machine intelligence* 35.11 (2013), pp. 2765–2781 (cit. on pp. 52, 144).

[Fan, 2021] Jicong Fan. "Large-Scale Subspace Clustering via k-Factorization". *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 342–352 (cit. on pp. 87, 92, 103).

[Fan, 2020] Shaohua Fan, Xiao Wang, Chuan Shi, Emiao Lu, Ken Lin, and Bai Wang. "One2multi graph autoencoder for multi-view graph clustering". *Proceedings of The Web Conference 2020*. 2020, pp. 3070–3076 (cit. on pp. 44, 126, 137, 142, 149).

[Fan, 2019] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, et al. "Graph Neural Networks for Social Recommendation". *The World Wide Web Conference*. WWW '19. San Francisco, CA, USA: Association for Computing Machinery, 2019, pp. 417–426 (cit. on pp. 68, 86, 102, 126).

[Fard, 2020] Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier. "Deep k-means: Jointly clustering with k-means and learning representations". *Pattern Recognition Letters* 138 (2020), pp. 185–192 (cit. on pp. 68, 70).

[Feltes, 2019] Bruno César Feltes, Eduardo Bassani Chandelier, Bruno Iochins Grisci, and Márcio Dorn. "CuMiDa: An Extensively Curated Microarray Database for Benchmarking and Testing of Machine Learning Approaches in Cancer Research". *Journal of Computational Biology* 26.4 (2019), pp. 376–386 (cit. on p. 47).

[Fettal, 2022a] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. "Apprentissage Joint de la Représentation et du Clustering avec un Réseau Convolutif sur Graphe". *Extraction et Gestion des Connaissances: EGC'2022* (2022) (cit. on pp. x, 4, 173).

[Fettal, 2022b] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. "Efficient and Effective Optimal Transport-Based Biclustering". *Advances in Neural Information Processing Systems*. 2022 (cit. on pp. x, 3, 33, 102, 173).

[Fettal, 2022c] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. "Efficient Graph Convolution for Joint Node Representation Learning and Clustering". *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 2022, pp. 289–297 (cit. on pp. x, 4, 45, 54, 58, 61, 67, 88, 91, 97, 104, 117, 132, 169, 173).

[Fettal, 2022d] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. "Subspace Co-clustering with Two-Way Graph Convolution". *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022, pp. 3938–3942 (cit. on pp. xi, 4, 99, 102, 169, 170, 174).

[Fettal, 2023a] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. "Biclustering Basé sur le Transport Optimal". *Extraction et Gestion des Connaissances: EGC'2023* (2023) (cit. on pp. x, 3, 173).

[Fettal, 2023b] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. "Scalable Attributed-Graph Subspace Clustering". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 2023 (cit. on pp. x, 4, 85, 102, 141, 144, 146, 149, 169, 174).

[Fettal, 2023c] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. "Simultaneous Linear Multi-view Attributed Graph Representation Learning and Clustering". *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 2023, pp. 303–311 (cit. on pp. xi, 4, 125, 144, 150, 169, 174).

[Fettal, 2023d] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. "Subspace Co-clustering avec Convolution Bilatérale sur Graphe". *Extraction et Gestion des Connaissances: EGC'2023* (2023) (cit. on pp. xi, 4, 174).

[Fettal, 2024a] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. "Boosting Subspace Co-Clustering via Bilateral Graph Convolution". *IEEE Transactions on Knowledge and Data Engineering* 36.3 (2024), pp. 960–971 (cit. on pp. xi, 4, 99).

[Fettal, 2024b] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. "More Discriminative Sentence Embeddings via Semantic Graph Smoothing". *18th Conference of the European Chapter of the Association for Computational Linguistics*. 2024 (cit. on pp. xi, 4, 155).

[Flamary, 2021] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, et al. "Pot: Python optimal transport". *Journal of Machine Learning Research* 22.78 (2021), pp. 1–8 (cit. on pp. 44, 63).

[Forrow, 2019] Aden Forrow, Jan-Christian Hütter, Mor Nitzan, Philippe Rigollet, Geoffrey Schiebinger, and Jonathan Weed. "Statistical optimal transport via factored couplings". *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 2454–2465 (cit. on p. 39).

[Freund, 1996] Yoav Freund, Robert E Schapire, et al. "Experiments with a new boosting algorithm". *icml*. Vol. 96. Citeseer. 1996, pp. 148–156 (cit. on p. 167).

[Freund, 1997] Yoav Freund and Robert E Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". *Journal of computer and system sciences* 55.1 (1997), pp. 119–139 (cit. on p. 167).

[Ganganath, 2014] Nuwan Ganganath, Chi-Tsun Cheng, and K Tse Chi. "Data clustering with cluster size constraints using a modified k-means algorithm". *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. IEEE. 2014, pp. 158–161 (cit. on p. 53).

[Gao, 2015] Hongchang Gao, Feiping Nie, Xuelong Li, and Heng Huang. "Multi-view subspace clustering". *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4238–4246 (cit. on pp. 142, 145, 147).

[Gasteiger, 2018] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. "Predict then Propagate: Graph Neural Networks meet Personalized PageRank". *International Conference on Learning Representations*. 2018 (cit. on p. 158).

[Genevay, 2019] Aude Genevay, Gabriel Dulac-Arnold, and Jean-Philippe Vert. "Differentiable deep clustering with cluster size constraints". *arXiv preprint arXiv:1910.09036* (2019) (cit. on pp. 52, 54, 62).

[Geurts, 2006] Pierre Geurts, Damien Ernst, and Louis Wehenkel. "Extremely randomized trees". *Machine learning* 63.1 (2006), pp. 3–42 (cit. on p. 168).

[Ghasedi Dizaji, 2017] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization". *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5736–5745 (cit. on p. 68).

[Govaert, 2003] Gérard Govaert and Mohamed Nadif. "Clustering with block mixture models". *Pattern Recognition* 36.2 (2003), pp. 463–473 (cit. on p. 25).

[Govaert, 2008] Gérard Govaert and Mohamed Nadif. "Block clustering with Bernoulli mixture models: Comparison of different approaches". *Computational Statistics & Data Analysis* 52.6 (2008), pp. 3233–3245 (cit. on pp. 47, 101, 118).

[Govaert, 2013] Gérard Govaert and Mohamed Nadif. *Co-clustering: models, algorithms and applications*. John Wiley & Sons, 2013 (cit. on pp. 34, 101–103).

[Govaert, 2018] Gérard Govaert and Mohamed Nadif. "Mutual information, phi-squared and model-based co-clustering for contingency tables". *Advances in data analysis and classification* 12.3 (2018), pp. 455–488 (cit. on p. 34).

[Grover, 2016] Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks". *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864 (cit. on p. 69).

[Gu, 2008] Jiajun Gu and Jun S Liu. "Bayesian biclustering of gene expression data". *BMC genomics* 9.1 (2008), pp. 1–10 (cit. on p. 34).

[Guo, 2018] Ting Guo, Shirui Pan, Xingquan Zhu, and Chengqi Zhang. "CFOND: consensus factorization for co-clustering networked data". *IEEE Transactions on Knowledge and Data Engineering* 31.4 (2018), pp. 706–719 (cit. on p. 103).

[Haeffele, 2021] Benjamin David Haeffele, Chong You, and Rene Vidal. "A Critique of Self-Expressive Deep Subspace Clustering". *International Conference on Learning Representations*. 2021 (cit. on pp. viii, 2, 87, 103).

[Hagen, 1992] Lars Hagen and Andrew B Kahng. "New spectral methods for ratio cut partitioning and clustering". *IEEE transactions on computer-aided design of integrated circuits and systems* 11.9 (1992), pp. 1074–1085 (cit. on p. 52).

[Halko, 2011] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions". *SIAM review* 53.2 (2011), pp. 217–288 (cit. on pp. 77, 92, 114).

[Hamilton, 2017] William L. Hamilton, Rex Ying, and Jure Leskovec. "Representation Learning on Graphs: Methods and Applications". *IEEE Data Eng. Bull.* 40.3 (2017), pp. 52–74 (cit. on p. 68).

[Handcock, 2007] Mark S Handcock, Adrian E Raftery, and Jeremy M Tantrum. "Model-based clustering for social networks". *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 170.2 (2007), pp. 301–354 (cit. on pp. viii, 2).

[Harpaz, 2011] Rave Harpaz, Hector Perez, Herbert S Chase, Raul Rabadan, George Hripcsak, and Carol Friedman. "Biclustering of adverse drug events in the FDA's spontaneous reporting system". *Clinical Pharmacology & Therapeutics* 89.2 (2011), pp. 243–250 (cit. on p. 34).

[Hartigan, 1972] John A Hartigan. "Direct clustering of a data matrix". *Journal of the american statistical association* 67.337 (1972), pp. 123–129 (cit. on p. 34).

[Hassani, 2020] Kaveh Hassani and Amir Hosein Khasahmadi. "Contrastive multi-view representation learning on graphs". *International Conference on Machine Learning*. PMLR. 2020, pp. 4116–4126 (cit. on p. 27).

[He, 2010] Jianming He and Wesley W Chu. "A social network-based recommender system (SNRS)". *Data mining for social network data*. Springer, 2010, pp. 47–74 (cit. on pp. viii, 2).

[Hersh, 1994] William Hersh, Chris Buckley, TJ Leone, and David Hickam. "OHSUMED: An interactive retrieval evaluation and new large test collection for research". *SIGIR'94*. Springer. 1994, pp. 192–201 (cit. on p. 44).

[Hitchcock, 1941] Frank L Hitchcock. "The distribution of a product from several sources to numerous localities". *Journal of mathematics and physics* 20.1-4 (1941), pp. 224–230 (cit. on p. 59).

[Hollocou, 2019] Alexandre Hollocou, Thomas Bonald, and Marc Lelarge. "Modularity-based sparse soft graph clustering". *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 323–332 (cit. on p. 40).

[Holm, 1979] Sture Holm. "A simple sequentially rejective multiple test procedure". *Scandinavian journal of statistics* (1979), pp. 65–70 (cit. on p. 151).

[Höppner, 2008] Frank Höppner and Frank Klawonn. "Clustering with size constraints". *Computational Intelligence Paradigms*. Springer, 2008, pp. 167–180 (cit. on p. 52).

[Hornik, 2012] Kurt Hornik, Ingo Feinerer, Martin Kober, and Christian Buchta. "Spherical k-Means Clustering". *Journal of Statistical Software, Articles* 50.10 (2012), pp. 1–22 (cit. on p. 80).

[Hoshen, 2017] Yedid Hoshen. "VAIN: Attentional Multi-agent Predictive Modeling". *Neural Information Processing Systems (NIPS)*. 2017, pp. 2701–2711 (cit. on pp. 68, 86).

[Hu, 2014] Han Hu, Zhouchen Lin, Jianjiang Feng, and Jie Zhou. "Smooth representation clustering". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 3834–3841 (cit. on p. 106).

[Hu, 2020] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, et al. "Open graph benchmark: Datasets for machine learning on graphs". *Advances in neural information processing systems* 33 (2020), pp. 22118–22133 (cit. on pp. 94, 142, 149).

[Hubert, 1985] Lawrence Hubert and Phipps Arabie. "Comparing partitions". *Journal of classification* 2.1 (1985), pp. 193–218 (cit. on pp. 23, 44, 63, 94, 119, 134, 159, 170).

[Jelodar, 2019] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, et al. "Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey". *Multimedia Tools and Applications* 78 (2019), pp. 15169–15211 (cit. on p. 102).

[Ji, 2017] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. "Deep subspace clustering networks". *Advances in neural information processing systems* 30 (2017) (cit. on pp. 52, 87).

[Ji, 2019] Xu Ji, Joao F Henriques, and Andrea Vedaldi. "Invariant information clustering for unsupervised image classification and segmentation". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9865–9874 (cit. on p. 52).

[Jindal, 2007] Nitin Jindal and Bing Liu. "Review spam detection". *Proceedings of the 16th international conference on World Wide Web*. 2007, pp. 1189–1190 (cit. on p. 156).

[Kang, 2020] Zhao Kang, Wangtao Zhou, Zhitong Zhao, Junming Shao, Meng Han, and Zenglin Xu. "Large-scale multi-view subspace clustering in linear time". *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 4412–4419 (cit. on pp. 142, 143).

[Karim, 2020a] Md Rezaul Karim, Oya Beyan, Achille Zappa, Ivan G Costa, Dietrich Rebholz-Schuhmann, Michael Cochez, et al. "Deep learning-based clustering approaches for bioinformatics". *Briefings in Bioinformatics* (2020), pp. 1–23 (cit. on p. 68).

[Karim, 2020b] Md Rezaul Karim, Michael Cochez, Achille Zappa, Ratnesh Sahay, Dietrich Rebholz-Schuhmann, Oya Beyan, et al. "Convolutional Embedded Networks for Population Scale Clustering and Bio-ancestry Inferencing". *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2020) (cit. on p. 68).

[Ketchen, 1996] David J Ketchen and Christopher L Shook. "The application of cluster analysis in strategic management research: an analysis and critique". *Strategic management journal* 17.6 (1996), pp. 441–458 (cit. on p. 98).

[Kipf, 2016] Thomas N Kipf and Max Welling. "Variational Graph Auto-Encoders". *NIPS Workshop on Bayesian Deep Learning* (2016) (cit. on pp. 26, 69, 126, 136).

[Kipf, 2017] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". *International Conference on Learning Representations (ICLR)*. 2017 (cit. on pp. 17, 69, 86, 102, 128, 142, 156).

[Kluger, 2003] Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. "Spectral biclustering of microarray data: coclustering genes and conditions". *Genome research* 13.4 (2003), pp. 703–716 (cit. on p. 118).

[Konno, 1976] Hiroshi Konno. "A cutting plane algorithm for solving bilinear programs". *Mathematical Programming* 11.1 (1976), pp. 14–27 (cit. on p. 38).

[Krizhevsky, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images" (2009) (cit. on p. 62).

[Kullback, 1951] Solomon Kullback and Richard A Leibler. "On information and sufficiency". *The annals of mathematical statistics* 22.1 (1951), pp. 79–86 (cit. on p. 63).

[Kumar, 2011] Abhishek Kumar, Piyush Rai, and Hal Daume. "Co-regularized multi-view spectral clustering". *Advances in neural information processing systems* 24 (2011) (cit. on p. 28).

[Labiod, 2011] Lazhar Labiod and Mohamed Nadif. "Co-clustering for binary and categorical data with maximum modularity". *2011 IEEE 11th international conference on data mining*. IEEE. 2011, pp. 1140–1145 (cit. on p. 25).

[Labiod, 2021] Lazhar Labiod and Mohamed Nadif. "Efficient regularized spectral data embedding". *Advances in Data Analysis and Classification* 15.1 (2021), pp. 99–119 (cit. on p. 68).

[Laclau, 2017a] Charlotte Laclau and Mohamed Nadif. "Diagonal latent block model for binary data". *Statistics and Computing* 27.5 (2017), pp. 1145–1163 (cit. on p. 105).

[Laclau, 2017b] Charlotte Laclau, Ievgen Redko, Basarab Matei, Younes Bennani, and Vincent Brault. "Co-clustering through optimal transport". *International Conference on Machine Learning*. PMLR. 2017, pp. 1955–1964 (cit. on pp. 34, 47).

[Lang, 1995] Ken Lang. "Newsweeder: Learning to filter netnews". *Proceedings of the Twelfth International Conference on Machine Learning*. 1995, pp. 331–339 (cit. on p. 44).

[Lee, 2022] Seongwon Lee, Hongje Seong, Suhyeon Lee, and Euntai Kim. "Correlation Verification for Image Retrieval". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5374–5384 (cit. on p. 52).

[Li, 2018a] Qimai Li, Zhichao Han, and Xiao-Ming Wu. "Deeper insights into graph convolutional networks for semi-supervised learning". *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018 (cit. on p. 156).

[Li, 2018b] Zhihui Li, Feiping Nie, Xiaojun Chang, Zhigang Ma, and Yi Yang. "Balanced clustering via exclusive lasso: A pragmatic approach". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018 (cit. on pp. 52, 53).

[Lin, 2019] Weibo Lin, Zhu He, and Mingyu Xiao. "Balanced Clustering: A Uniform Model and Fast Algorithm." *IJCAI*. 2019, pp. 2987–2993 (cit. on p. 53).

[Lin, 2021a] Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, et al. "Bertgcn: Transductive text classification by combining gcn and bert". *arXiv preprint arXiv:2105.05727* (2021) (cit. on p. 156).

[Lin, 2021b] Zhiping Lin and Zhao Kang. "Graph Filter-based Multi-view Attributed Graph Clustering." *IJCAI*. 2021, pp. 2723–2729 (cit. on pp. 126, 137, 142, 144).

[Lin, 2021c] Zhiping Lin, Zhao Kang, Lizong Zhang, and Ling Tian. "Multi-view attributed graph clustering". *IEEE Transactions on Knowledge and Data Engineering* (2021) (cit. on pp. 126, 137, 142, 143).

[Liu, 2010] Guangcan Liu, Zhouchen Lin, and Yong Yu. "Robust subspace segmentation by low-rank representation". *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 663–670 (cit. on p. 144).

[Liu, 2017] Weiyi Liu, Pin-Yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. "Principled multilayer network embedding". *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2017, pp. 134–141 (cit. on p. 136).

[Liu, 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, et al. "Roberta: A robustly optimized bert pretraining approach". *arXiv preprint arXiv:1907.11692* (2019) (cit. on p. 156).

[Lloyd, 1982] S. P. Lloyd. "Least squares quantization in PCM". *IEEE Trans. Inf. Theory* 28 (1982), pp. 129–136 (cit. on p. 71).

[Lu, 2012] Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. "Robust and efficient subspace segmentation via least squares regression". *European conference on computer vision*. Springer. 2012, pp. 347–360 (cit. on pp. 63, 103, 106).

[Lu, 2013] Canyi Lu, Jiashi Feng, Zhouchen Lin, and Shuicheng Yan. "Correlation adaptive subspace segmentation by trace lasso". *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1345–1352 (cit. on p. 106).

[Maaten, 2008] Laurens van der Maaten and Geoffrey E. Hinton. "Visualizing High-Dimensional Data Using t-SNE". *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605 (cit. on p. 76).

[Marcheggiani, 2017] Diego Marcheggiani and Ivan Titov. "Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling". *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 1506–1515 (cit. on pp. 68, 86).

[Marcotorchino, 1987] Jean François Marcotorchino. "Block seriation problems: A unified approach. Reply to the problem of H. Garcia and JM Proth (Applied Stochastic Models and Data Analysis, 1,(1), 25–34 (1985))". *Applied Stochastic Models and Data Analysis* 3.2 (1987), pp. 73–91 (cit. on pp. 36, 105).

[Matias, 2014] Catherine Matias and Stéphane Robin. "Modeling heterogeneity in random graphs through latent space models: a selective review". *ESAIM: Proceedings and Surveys* 47 (2014), pp. 55–74 (cit. on p. 26).

[Mavromatis, 2021] Costas Mavromatis and George Karypis. "Graph InfoClust: Maximizing Coarse-Grain Mutual Information in Graphs". *PAKDD (1)*. 2021, pp. 541–553 (cit. on pp. 70, 80, 88, 104).

[Melville, 2009] Prem Melville, Wojciech Gryc, and Richard D Lawrence. "Sentiment analysis of blogs by combining lexical knowledge with text classification". *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 1275–1284 (cit. on p. 156).

[Mimno, 2011] David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. "Optimizing semantic coherence in topic models". *Proceedings of the 2011 conference on empirical methods in natural language processing*. 2011, pp. 262–272 (cit. on p. 123).

[Mishra, 2007] Nina Mishra, Robert Schreiber, Isabelle Stanton, and Robert E Tarjan. "Clustering social networks". *International Workshop on Algorithms and Models for the Web-Graph*. Springer. 2007, pp. 56–67 (cit. on pp. viii, 2).

[Mrabah, 2022] Nairouz Mrabah, Mohamed Bouguessa, Mohamed Fawzi Touati, and Riadh Ksantini. "Rethinking graph auto-encoder models for attributed graph clustering". *IEEE Transactions on Knowledge and Data Engineering* (2022) (cit. on p. 27).

[Nemenyi, 1963] Peter Bjorn Nemenyi. *Distribution-free multiple comparisons*. Princeton University, 1963 (cit. on pp. 48, 97, 121).

[Newman, 2009] David Newman, Sarvnaz Karimi, and Lawrence Cavedon. "External evaluation of topic models". *in Australasian Doc. Comp. Symp., 2009*. Citeseer. 2009 (cit. on p. 45).

[Ng, 2001] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an Algorithm". *International Conference on Neural Information Processing Systems: Natural and Synthetic*. Vol. 14. 2001, pp. 849–856 (cit. on pp. 52, 56, 70, 92, 114).

[Nie, 2017] Feiping Nie, Jing Li, Xuelong Li, et al. "Self-weighted Multiview Clustering with Multiple Graphs." *IJCAI*. 2017, pp. 2564–2570 (cit. on p. 137).

[Orlin, 1997] James B Orlin. "A polynomial time primal network simplex algorithm for minimum cost flows". *Mathematical Programming* 78.2 (1997), pp. 109–129 (cit. on pp. 42, 60).

[Ortega, 2018] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. "Graph signal processing: Overview, challenges, and applications". *Proceedings of the IEEE* 106.5 (2018), pp. 808–828 (cit. on pp. 129, 157).

[Pan, 2021] Erlin Pan and Zhao Kang. "Multi-view contrastive graph clustering". *Advances in neural information processing systems* 34 (2021), pp. 2148–2159 (cit. on pp. 142, 143, 147, 150).

[Pan, 2018] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. "Adversarially Regularized Graph Autoencoder for Graph Embedding". *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 2609–2615 (cit. on p. 27).

[Park, 2020] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. "Unsupervised attributed multiplex network embedding". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 2020, pp. 5371–5378 (cit. on pp. 126, 137).

[Park, 2019] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. "Symmetric Graph Convolutional Autoencoder for Unsupervised Graph Representation Learning". *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019 (cit. on pp. 26, 27).

[Parsons, 2004] Lance Parsons, Ehtesham Haque, and Huan Liu. "Subspace clustering for high dimensional data: a review". *Acm sigkdd explorations newsletter* 6.1 (2004), pp. 90–105 (cit. on p. 101).

[Pazzani, 2007] Michael J Pazzani and Daniel Billsus. "Content-based recommendation systems". *The adaptive web: methods and strategies of web personalization*. Springer, 2007, pp. 325–341 (cit. on p. 156).

[Pedregosa, 2011] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, et al. "Scikit-learn: Machine learning in Python". *the Journal of machine Learning research* 12 (2011), pp. 2825–2830 (cit. on pp. 63, 166).

[Perozzi, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "DeepWalk: online learning of social representations". *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2014, pp. 701–710 (cit. on p. 68).

[Peyré, 2017] Gabriel Peyré, Marco Cuturi, et al. "Computational optimal transport". *Center for Research in Economics and Statistics Working Papers* 2017-86 (2017) (cit. on pp. 37, 39).

[Peyré, 2019] Gabriel Peyré, Marco Cuturi, et al. "Computational optimal transport: With applications to data science". *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607 (cit. on p. 58).

[Peyré, 2016] Gabriel Peyré, Marco Cuturi, and Justin Solomon. "Gromov-wasserstein averaging of kernel and distance matrices". *International conference on machine learning*. PMLR. 2016, pp. 2664–2672 (cit. on pp. 56, 59).

[Pham, 2013] Ninh Pham and Rasmus Pagh. "Fast and scalable polynomial kernels via explicit feature maps". *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 239–247 (cit. on pp. 93, 114, 148).

[Pizzuti, 2014] Clara Pizzuti and Simona E Rombo. "Algorithms and tools for protein–protein interaction networks clustering, with a special focus on population-based stochastic methods". *Bioinformatics* 30.10 (2014), pp. 1343–1352 (cit. on pp. viii, 2).

[Qi, 2017] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. "3D Graph Neural Networks for RGBD Semantic Segmentation". *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5209–5218 (cit. on pp. 68, 126).

[Quinlan, 1986] J. Ross Quinlan. "Induction of decision trees". *Machine learning* 1.1 (1986), pp. 81–106 (cit. on p. 166).

[Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993 (cit. on p. 166).

[Rahimi, 2007] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". *Advances in neural information processing systems* 20 (2007) (cit. on p. 114).

[Reimers, 2019] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 3982–3992 (cit. on p. 156).

[Rijsbergen CJ, 1979] van Rijsbergen (CJ). *Information retrieval*. Butterworth, 1979 (cit. on pp. 134, 170).

[Riverain, 2022] Paul Riverain, Simon Fossier, and Mohamed Nadif. "Semi-supervised Latent Block Model with pairwise constraints". *Machine Learning* 111.5 (2022), pp. 1739–1764 (cit. on pp. 26, 101).

[Rousseeuw, 1987] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". *Journal of computational and applied mathematics* 20 (1987), pp. 53–65 (cit. on p. 58).

[Rozemberczki, 2019] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. "Gemsec: Graph embedding with self clustering". *Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining*. 2019, pp. 65–72 (cit. on pp. 26, 70).

[Rumelhart, 1985] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985 (cit. on p. 26).

[Salah, 2018] Aghiles Salah, Melissa Ailem, and Mohamed Nadif. "Word co-occurrence regularized non-negative matrix tri-factorization for text data co-clustering". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018 (cit. on p. 102).

[Salah, 2017a] Aghiles Salah and Mohamed Nadif. "Model-based von mises-fisher co-clustering with a conscience". *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM. 2017, pp. 246–254 (cit. on pp. 41, 103).

[Salah, 2017b] Aghiles Salah and Mohamed Nadif. "Social regularized von Mises–Fisher mixture model for item recommendation". *Data Mining and Knowledge Discovery* 31.5 (2017), pp. 1218–1241 (cit. on pp. 68, 126).

[Salha, 2020] Guillaume Salha, Romain Hennequin, and Michalis Vazirgiannis. "Simple and Effective Graph Autoencoders with One-Hop Linear Models". *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*. 2020, pp. 319–334 (cit. on pp. 70, 71, 80, 129).

[Sanchez-Gonzalez, 2018] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, et al. "Graph networks as learnable physics engines for inference and control". *International Conference on Machine Learning*. 2018, pp. 4470–4479 (cit. on pp. 68, 126).

[Satorras, 2018] Victor Garcia Satorras and Joan Bruna Estrach. "Few-Shot Learning with Graph Neural Networks". *International Conference on Learning Representations*. 2018 (cit. on pp. 68, 86, 126).

[Scetbon, 2022] Meyer Scetbon and marco cuturi. "Low-rank Optimal Transport: Approximation, Statistics and Debiasing". *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022 (cit. on p. 65).

[Scetbon, 2021] Meyer Scetbon, Marco Cuturi, and Gabriel Peyré. "Low-rank sinkhorn factorization". *International Conference on Machine Learning*. PMLR. 2021, pp. 9344–9354 (cit. on p. 39).

[Sen, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. "Collective classification in network data". *AI magazine* 29.3 (2008), pp. 93–93 (cit. on pp. 44, 79, 94, 102, 119).

[Shchur, 2018] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. "Pitfalls of graph neural network evaluation". *arXiv preprint arXiv:1811.05868* (2018) (cit. on pp. 94, 119, 134, 142, 149).

[Shi, 2000] Jianbo Shi and Jitendra Malik. "Normalized cuts and image segmentation". *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 888–905 (cit. on pp. 52, 55, 90, 104).

[Shuman, 2013] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". *IEEE signal processing magazine* 30.3 (2013), pp. 83–98 (cit. on pp. 16, 129, 157).

[Song, 2018] Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. "A Graph-to-Sequence Model for AMR-to-Text Generation". *the Association for Computational Linguistics, ACL 2018*. 2018, pp. 1616–1626 (cit. on p. 68).

[Strehl, 2002] Alexander Strehl and Joydeep Ghosh. "Cluster ensembles—a knowledge reuse framework for combining multiple partitions". *Journal of machine learning research* 3.Dec (2002), pp. 583–617 (cit. on pp. 94, 134, 170).

[Sun, 2020] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. "InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization". *International Conference on Learning Representations*. 2020 (cit. on p. 69).

[Sun, 2021] Mengjing Sun, Pei Zhang, Siwei Wang, Sihang Zhou, Wenxuan Tu, Xinwang Liu, et al. "Scalable multi-view subspace clustering with unified anchors". *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 3528–3536 (cit. on pp. 142, 143).

[Tang, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. "Line: Large-scale information network embedding". *Proceedings of the 24th international conference on world wide web*. 2015, pp. 1067–1077 (cit. on p. 136).

[Templin, 2010] Jonathan Templin, Robert A Henson, et al. *Diagnostic measurement: Theory, methods, and applications*. Guilford Press, 2010 (cit. on p. 34).

[Titouan, 2020] Vayer Titouan, Ievgen Redko, Rémi Flamary, and Nicolas Courty. "Co-optimal transport". *Advances in Neural Information Processing Systems* 33 (2020), pp. 17559–17570 (cit. on pp. 34, 47).

[Velickovic, 2019] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. "Deep Graph Infomax". *ICLR (Poster)* 2.3 (2019), p. 4 (cit. on pp. 69, 80).

[Veličković, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks". *International Conference on Learning Representations*. 2018 (cit. on p. 175).

[Vinh, 2009] Nguyen Xuan Vinh, Julien Epps, and James Bailey. "Information theoretic measures for clusterings comparison: is a correction for chance necessary?" *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 1073–1080 (cit. on p. 159).

[Von Luxburg, 2007] Ulrike Von Luxburg. "A tutorial on spectral clustering". *Statistics and computing* 17.4 (2007), pp. 395–416 (cit. on pp. ix, 52).

[Wang, 2017] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. "Mgae: Marginalized graph autoencoder for graph clustering". *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 889–898 (cit. on p. 27).

[Wang, 2016] Daixin Wang, Peng Cui, and Wenwu Zhu. "Structural deep network embedding". *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 1225–1234 (cit. on p. 69).

[Wang, 2011] Hua Wang, Feiping Nie, Heng Huang, and Fillia Makedon. "Fast nonnegative matrix tri-factorization for large-scale data co-clustering". *Twenty-Second International Joint Conference on Artificial Intelligence*. 2011 (cit. on p. 34).

[Wang, 2018a] Jianfeng Wang, Mei Lu, Francesco Belardo, and Milan Randić. "The anti-adjacency matrix of a graph: Eccentricity matrix". *Discrete Applied Mathematics* 251 (2018), pp. 299–309 (cit. on p. 36).

[Wang, 2021a] Tongxin Wang, Wei Shao, Zhi Huang, Haixu Tang, Jie Zhang, Zhengming Ding, et al. "MOGONET integrates multi-omics data using graph convolutional networks allowing patient classification and biomarker identification". *Nature Communications* 12.1 (2021), p. 3445 (cit. on p. 142).

[Wang, 2019] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, et al. "Heterogeneous graph attention network". *The world wide web conference*. 2019, pp. 2022–2032 (cit. on pp. 94, 119, 134).

[Wang, 2018b] Yang Wang and Lin Wu. "Beyond low-rank representations: Orthogonal clustering basis reconstruction with optimized graph structure for multi-view spectral clustering". *Neural Networks* 103 (2018), pp. 1–8 (cit. on p. 103).

[Wang, 2018c] Yang Wang, Lin Wu, Xuemin Lin, and Junbin Gao. "Multiview Spectral Clustering via Structured Low-Rank Matrix Factorization". *IEEE Transactions on Neural Networks and Learning Systems* 29.10 (2018), pp. 4833–4843 (cit. on p. 103).

[Wang, 2015] Yang Wang, Wenjie Zhang, Lin Wu, Xuemin Lin, and Xiang Zhao. "Unsupervised metric fusion over multiview data by graph random walk-based cross-view diffusion". *IEEE transactions on neural networks and learning systems* 28.1 (2015), pp. 57–70 (cit. on p. 103).

[Wang, 2021b] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. "Dissecting the diffusion process in linear graph convolutional networks". *Advances in Neural Information Processing Systems* 34 (2021), pp. 5758–5769 (cit. on p. 158).

[Williams, 2000] Christopher Williams and Matthias Seeger. "Using the Nyström method to speed up kernel machines". *Advances in neural information processing systems* 13 (2000) (cit. on p. 148).

[Wu, 2019] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. "Simplifying graph convolutional networks". *International conference on machine learning*. PMLR. 2019, pp. 6861–6871 (cit. on pp. 18, 70, 72, 86, 88, 90, 102, 128, 142, 150, 156, 157).

[Xia, 2014] Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. "Robust multi-view spectral clustering via low-rank and sparse decomposition". *Proceedings of the AAAI conference on artificial intelligence*. Vol. 28. 2014 (cit. on p. 136).

[Xiao, 2017] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms". *arXiv preprint arXiv:1708.07747* (2017) (cit. on p. 62).

[Xu, 2013] Chang Xu, Dacheng Tao, and Chao Xu. "A survey on multi-view learning". *arXiv preprint arXiv:1304.5634* (2013) (cit. on p. 126).

[Xu, 2017] Danfei Xu, Yuke Zhu, Christopher Choy, and Li Fei-Fei. "Scene Graph Generation by Iterative Message Passing". *Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on p. 126).

[Xu, 2019] Hongteng Xu, Dixin Luo, and Lawrence Carin. "Scalable gromov-wasserstein learning for graph partitioning and matching". *Advances in neural information processing systems* 32 (2019) (cit. on pp. 56, 62).

[Xu, 2022] Yue Xu, Yong-Lu Li, Jiefeng Li, and Cewu Lu. "Constructing balance from imbalance for long-tailed image recognition". *European Conference on Computer Vision*. Springer. 2022, pp. 38–56 (cit. on p. 52).

[Yamamoto, 2014] Michio Yamamoto and Heungsun Hwang. "A general formulation of cluster analysis with dimension reduction and subspace separation". *Behaviormetrika* 41.1 (2014), pp. 115–129 (cit. on pp. 68, 74, 130, 132).

[Yang, 2017] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. "Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering". *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 3861–3870 (cit. on pp. 70, 74, 80).

[Yang, 2015] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. "Network Representation Learning with Rich Text Information". *IJCAI*. 2015 (cit. on pp. 44, 79, 94, 119, 134).

[Yang, 2018a] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. *Graph R-CNN for Scene Graph Generation*. 2018 (cit. on pp. 68, 86).

[Yang, 2018b] Yingzhen Yang, Jiashi Feng, Nebojsa Jojic, Jianchao Yang, and Thomas S Huang. "Subspace learning by L0-induced sparsity". *International Journal of Computer Vision* 126.10 (2018), pp. 1138–1156 (cit. on pp. 103, 117).

[Yang, 2022] Yingzhen Yang and Ping Li. "Noisy L0-sparse subspace clustering on dimensionality reduced data". *Uncertainty in Artificial Intelligence*. PMLR. 2022, pp. 2235–2245 (cit. on pp. 103, 117).

[Yao, 2019] Liang Yao, Chengsheng Mao, and Yuan Luo. "Graph convolutional networks for text classification". *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 7370–7377 (cit. on p. 156).

[Ying, 2018] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. "Graph convolutional neural networks for web-scale recommender systems". *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 974–983 (cit. on pp. 68, 86, 126).

[You, 2016a] Chong You, Chun-Guang Li, Daniel P Robinson, and René Vidal. "Oracle based active set algorithm for scalable elastic net subspace clustering". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3928–3937 (cit. on pp. 87, 103).

[You, 2016b] Chong You, Daniel Robinson, and René Vidal. "Scalable sparse subspace clustering by orthogonal matching pursuit". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3918–3927 (cit. on pp. 87, 103).

[Yu, 2020] Zhiwen Yu, Zhongfan Zhang, Wenming Cao, Cheng Liu, Junlong Philip Chen, and Hau San Wong. "Gan-based enhanced deep subspace clustering networks". *IEEE Transactions on Knowledge and Data Engineering* (2020) (cit. on p. 103).

[Zhang, 2018] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. "Scalable multiplex network embedding." *IJCAI*. Vol. 18. 2018, pp. 3082–3088 (cit. on p. 136).

[Zhang, 2008] Kai Zhang, Ivor W Tsang, and James T Kwok. "Improved Nyström low-rank approximation and error analysis". *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1232–1239 (cit. on p. 93).

[Zhang, 2023] Pei Zhang, Siwei Wang, Liang Li, Changwang Zhang, Xinwang Liu, En Zhu, et al. "Let the Data Choose: Flexible and Diverse Anchor Graph Fusion for Scalable Multi-View Clustering". *Proceedings of the AAAI Conference on Artificial Intelligence* 37.9 (2023), pp. 11262–11269 (cit. on p. 143).

[Zhang, 2019] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. "Attributed Graph Clustering via Adaptive Graph Convolution". *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. IJCAI'19. Macao, China: International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 4327–4333 (cit. on pp. 26, 27, 69, 70, 77, 97).

[Zhao, 2021] Han Zhao, Xu Yang, Zhenru Wang, Erkun Yang, and Cheng Deng. "Graph Debiased Contrastive Learning with Joint Representation Clustering." *IJCAI*. 2021, pp. 3434–3440 (cit. on p. 27).

[Zhou, 2004] Dengyong Zhou and Bernhard Schölkopf. "A regularization framework for learning from graph data". *ICML 2004 Workshop on Statistical Relational Learning and Its Connections to Other Fields (SRL 2004)*. 2004, pp. 132–137 (cit. on p. 128).

[Zhu, 2021] Hao Zhu and Piotr Koniusz. "Simple Spectral Graph Convolution". *9th International Conference on Learning Representations, ICLR, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021 (cit. on pp. 26, 28, 70, 74, 86, 88, 104, 156, 157).

[Zhu, 2010] Shunzhi Zhu, Dingding Wang, and Tao Li. "Data clustering with size constraints". *Knowledge-Based Systems* 23.8 (2010), pp. 883–889 (cit. on pp. 52, 53).

[Zhu, 2020] Wenwu Zhu, Xin Wang, and Peng Cui. "Deep learning for learning graph representations". *Deep Learning: Concepts and Architectures*. Springer, 2020, pp. 169–210 (cit. on p. 69).

[Zhu, 2014] Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. "Capturing long-tail distributions of object subcategories". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 915–922 (cit. on p. 52).