Deep probabilistic models for recommender systems and network clustering

Dingge Liang

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES FONDAMENTALES ET APPLIQUÉES

THÈSE DE DOCTORAT

Modèles probabilistes profonds pour les systèmes de recommandation et le clustering de réseaux

Dingge LIANG

Centre Inria d'Université Côte d'Azur, Laboratoire Jean Alexandre Dieudonné

Présentée en vue de l'obtention
du grade de docteur en mathématiques appliquées
d'Université Côte d'Azur

Dirigée par : Charles BOUVEYRON, Professeur, Université Côte d'Azur & Inria

Co-dirigée par : Pierre LATOUCHE, Professeur, Université Clermont Auvergne

Co-encadrée par : Marco CORNELI, Chaire de Professeur Junior, Université Côte d'Azur

Soutenue le : 15 décembre 2022

Devant le jury, composé de :

Pierre BARBILLON, Professeur, Université Paris-Saclay

Monica BIANCHINI, Professeur, University of Siena

Hervé DELINGETTE, Directeur de recherche, Inria

Chloé FRIGUET, Maître de conférences, Université de Bretagne Sud

Claire GORMLEY, Professeur, University College Dublin

MODÈLES PROBABILISTES PROFONDS POUR LES SYSTÈMES DE RECOMMANDATION ET LE CLUSTERING DE RÉSEAUX

Deep probabilistic models for recommender systems and network clustering

Dingge LIANG

Jury :

Rapporteurs

Pierre BARBILLON, Professeur, Université Paris-Saclay

Monica BIANCHINI, Professeur, University of Siena

Examinateurs

Hervé DELINGETTE, Directeur de recherche, Inria

Chloé FRIGUET, Maître de conférences, Université de Bretagne Sud

Claire GORMLEY, Professeur, University College Dublin

Directeur de thèse

Charles BOUVEYRON, Professeur, Université Côte d'Azur & Inria

Co-directeur de thèse

Pierre LATOUCHE, Professeur, Université Clermont Auvergne

Co-encadrant de thèse

Marco CORNELI, Chaire de Professeur Junior, Université Côte d'Azur

---

Université Côte d'Azur

Dingge LIANG

Modèles probabilistes profonds pour les systèmes de recommandation et le clustering de réseaux

xiv+157 p.

Ce document a été préparé avec LaTeX2e et la classe `these-ISSS` version v. 2.10.

Impression : sommaire.tex – 21/3/2023 – 2:31

Résumé

Cette thèse ponte sur de nouvelles méthodes d'analyse de trois types de données: ordinales, d'interaction et textuelles. Avec le développement du numérique, les données ordinales, liées à l'évaluation de produits ou services, sont omniprésentes sur des sites web tels qu'Amazon et Yelp. En effet, les clients peuvent obtenir des informations précieuses sur les produits et les services à partir de données de ce type, ce qui les aide à prendre des décisions. D'autre part, les données d'interaction, qu'il s'agisse de médias sociaux, de communications par courrier électronique ou d'interactions protéine-protéine, peuvent souvent être modélisés sous forme de graphes: des structures simples mais capables de modéliser des systèmes complexes. Enfin, avec le développement d'Internet et la croissance des médias sociaux, des quantités massives de données textuelles sont générées sous la forme de blogs, de tweets, de commentaires et d'enquêtes. Les trois types de données illustrés jusqu'ici peuvent être utilisées individuellement, pour diverses tâches, mais elles peuvent également être combinées, ce qui entraîne les problèmes typiques d'analyse des données hétérogènes.

Dans cette thèse, nous analysons ces trois types de données à travers trois modèles génératifs profonds, qui combinent la modélisation probabiliste et les techniques d'apprentissage profond. Premièrement, nous introduisons un système de recommandation latent profond (deepLTRS) afin de fournir aux utilisateurs des recommandations de haute qualité basées sur les évaluations observées des utilisateurs et les textes des critiques de produits. Notre approche adopte une architecture d'auto-encodeur variationnel (VAE) comme modèle latent génératif profond pour une matrice ordinale codant les évaluations et une matrice de termes et documents codant les critiques. Des expériences numériques sur des ensembles de données simulées et réelles démontrent que deepLTRS surpasse l'état de l'art, en particulier dans le contexte d'une extrême rareté des données. Le modèle de positions latentes profond (DeepLPM) est ensuite présenté comme une approche de clustering génératif de bout en bout qui combine le modèle de position latente couramment utilisé pour l'analyse de réseaux avec une stratégie d'encodage de réseau convolutif de graphes. Des expériences numériques sur des scénarios simulés mettent en évidence ses capacités de clustering. DeepLPM est ensuite appliqué à un réseau ecclésiastique de la Gaule mérovingienne et au réseau de citations Cora pour illustrer l'intérêt pratique de l'exploration de grands réseaux complexes du

monde réel. Enfin, nous proposons un encodeur de réseau convolutif de graphes basé sur la similarité des documents (DS-GCN) pour combiner les réseaux convolutifs de graphes et les modèles thématiques intégrés pour une représentation de réseaux riches en texte. En incluant une variable d'appartenance à un groupe, nous construisons ainsi une méthode de regroupement de « bout en bout » appelée GETM. La capacité de GETM à fusionner la structure topologique du graphe et les modèles thématiques intégrés est démontrée par des expériences numériques sur trois réseaux synthétiques, qui soulignent également ses performances en matière de clustering de nœuds.

Mots-clés: Modèles de variables latentes profondes, modélisation de sujets, systèmes de recommandation, réseaux de neurones graphiques, analyse de réseau, regroupement de nœuds.

Abstract

This thesis focuses on new methods for analyzing three types of data: ordinal, interactive and textual. With the advancement of the digital era, the ordinal data, related to the evaluation of products or services, is ubiquitous on websites such as Amazon and Yelp. Indeed, customers can gain valuable information about products and services from rating data, which helps them make decisions. Besides, the interaction data, whether from social media, email communications or protein-protein interactions, can often be modeled as graphs, because they are simple structures yet are capable of modeling complex real-world systems. Moreover, with the development of the Internet and the growth of social media, massive amounts of textual data are generated in the form of blogs, tweets, comments, and surveys. The three types of data illustrated so far can be used individually, for various tasks, but they can also be combined, leading to the typical problems of heterogeneous data analysis.

In this thesis, we analyze these three types of data through three deep generative models, which combine probabilistic modeling and deep learning techniques. First, we introduce a deep latent recommender system (deepLTRS) in order to provide users with high quality recommendations based on observed user ratings and texts of product reviews. Our approach adopts a variational auto-encoder (VAE) architecture as a deep generative latent model for an ordinal matrix encoding ratings and a document-term matrix encoding the reviews. Numerical experiments on simulated and real-world data sets demonstrate that deepLTRS outperforms the state-of-the-art, in particular, in the context of extreme data sparsity. The deep latent position model (DeepLPM) is then introduced as an end-to-end generative clustering approach that combines the widely used latent position model for network analysis with a graph convolutional network encoding strategy. Numerical experiments on simulated scenarios highlight its clustering capabilities. DeepLPM is further applied to an ecclesiastical network in Merovingian Gaul and to the citation network Cora to illustrate the practical interest in exploring large and complex real-world networks. Finally, we propose a document similarity-based graph convolutional network encoder (DS-GCN) to combine graph convolutional networks and embedded topic models for a text-rich network representation. By including a cluster membership variable, we thus build an end-to-end clustering method named graph embedded topic model (GETM). The ability of GETM in fusing the graph topology

structure and the topic embeddings is demonstrated by numerical experiments on three synthetic networks, which also emphasize its performance in node clustering.

Keywords: Deep latent variable models, Topic modeling, Recommender systems, Graph neural networks, Network analysis, Node clustering.

Remerciements

Tout d'abord, je tiens à remercier Charles, Pierre et Marco pour la qualité de votre encadrement durant ma thèse. Cette thèse est à la fois une aventure humaine et scientifique, et vous m'avez beaucoup aidé sur ces deux aspects. Ce manuscrit doit beaucoup à votre disponibilité, vos conseils, vos idées, votre patience et votre confiance durant ces trois années.

Ensuite, merci pour la meilleure équipe Maasai, j'ai senti que nous étions comme une grande famille chaleureuse. Je me souviendrai toujours des activités que nous avons faites ensemble, le kayak, la randonnée... et toutes les conférences auxquelles nous avons participé ensemble sont de très beaux voyages. Merci à Aude, Alessandro, Cedric, Célia, Davide, Giulia, Gabriele, Gianluigi, Hugo, Kevin, Louis, Lucas, Mansour, Stéphane, Xuchun... vous n'êtes pas seulement des collèges mais aussi de bons amis pour moi. J'ai eu beaucoup de chance de vous rencontrer tous. Et merci à Fred, Pierre-Alexandre, Vincent, Damien, Michel... vous êtes de très bons professeurs, chercheurs et j'ai beaucoup appris de vous tous.

Je veux aussi remercier tous mes amis, mon petit copain Mulin, mon petit chat Duff... Nous avons passé tant de merveilleux moments ensemble, et j'ai senti votre soutien et votre attention pendant les temps difficiles.

Pour ma famille en Chine, bien que nous soyons éloignés les uns des autres, je ressens toujours votre soutien, votre attention et votre confiance. Sans vous, je ne pourrais pas arriver à ce stade.

Enfin, j'ai eu beaucoup de chance et de bonheur de terminer cette thèse. Et j'ai beaucoup apprécié ces six années en France, à Nice, à Sophia-Antipolis. Ce sera un voyage inoubliable dans ma vie. Merci pour tout !

Contents

Notations 1

1 Introduction 3

1.1 Recommender systems . . . 5

1.1.1 Rating data and examples . . . 5

1.1.2 What is a recommender system ? . . . 6

1.2 Network analysis . . . 8

1.2.1 Network data and examples . . . 8

1.2.2 Clustering in networks . . . 10

1.3 Text analysis . . . 11

1.3.1 Text data and examples . . . 11

1.3.2 Challenges in text analysis . . . 13

1.4 The need for methods to analyze heterogeneous data . . . 14

1.4.1 Context . . . 14

1.4.2 The rise of deep probabilistic methods . . . 15

1.5 Organization of the thesis . . . 17

2 Statistical and deep learning models for recommender systems and node clustering 19

2.1 Warmup : a few fundamental DLVMs . . . 21

2.1.1 Variational auto-encoders for continuous data . . . 21

2.1.2 Topic modeling for count data . . . 24

2.2 Construction of recommender systems . . . 26

2.2.1 Matrix factorization models . . . 26

2.2.2 Latent factor models for recommender systems . . . 27

2.2.3 Deep learning-based recommender systems . . . 30

2.3 Network data analysis . . . 32

2.3.1 Probabilistic graphical models . . . 32

2.3.2 Graph neural networks . . . 34
2.3.3 Deep probabilistic models for node clustering . . . 37
2.4 Clustering in heterogeneous information networks . . . 40

3 DeepLTRS: a deep latent recommender system based on user ratings and reviews 45
3.1 Introduction . . . 47
3.1.1 Organization of the chapter . . . 48
3.2 A rating-and-review based recommender system . . . 48
3.2.1 Framework and notations . . . 48
3.2.2 Generative model of deepLTRS . . . 49
3.3 Variational auto-encoding inference . . . 52
3.3.1 Variational lower bound (ELBO) . . . 52
3.3.2 Monte Carlo EM algorithm and mini-batching . . . 54
3.4 Numerical experiments on simulated data . . . 57
3.4.1 Architecture and simulation setup . . . 57
3.4.2 DeepLTRS with and without text data . . . 58
3.4.3 Benchmark and effect of data sparsity . . . 61
3.5 Application on real-world data . . . 62
3.6 Conclusion and perspectives . . . 68

4 Clustering by deep latent position model with graph convolutional networks 69
4.1 Introduction . . . 73
4.1.1 Organization of the chapter . . . 75
4.2 Deep latent position model . . . 75
4.2.1 Notations . . . 75
4.2.2 Generative model . . . 75
4.3 Model inference . . . 77
4.3.1 Variational auto-encoding inference . . . 77
4.3.2 Links with related models . . . 78
4.3.3 Optimization . . . 79
4.3.4 Model selection . . . 83
4.4 Numerical experiments . . . 84

4.4.1 Simulation setup . . . 84
4.4.2 Benchmark study . . . 85
4.4.3 Model selection . . . 88
4.5 Analysis of a medieval network . . . 90
4.5.1 Dataset . . . 90
4.5.2 Results without covariates . . . 91
4.5.3 Results with covariates . . . 93
4.6 Cora citation network . . . 96
4.6.1 Dataset . . . 96
4.6.2 Results without covariates . . . 97
4.6.3 Results with covariates . . . 99
4.7 Conclusion and perspectives . . . 104

5 The graph embedded topic model **107**
5.1 Introduction . . . 109
5.1.1 Organization of the chapter . . . 110
5.2 The graph embedded topic model . . . 110
5.2.1 Notations . . . 110
5.2.2 Generative model . . . 111
5.3 Inference model . . . 113
5.3.1 Variational inference . . . 113
5.3.2 Document similarity-based GCN. . . . 114
5.3.3 Optimization . . . 116
5.4 Numerical experiments . . . 119
5.4.1 Simulation setup . . . 120
5.4.2 Benchmark study . . . 121
5.4.3 A more detailed example . . . 122
5.4.4 Model selection . . . 126
5.5 Application on real-world network . . . 128
5.5.1 Model selection . . . 128
5.5.2 Visualisation and analysis . . . 130

5.6 Conclusion 134

6 Conclusion and Perspectives **135**

6.1 Summary of the contributions 135

6.2 Perspectives 136

6.2.1 Graph learning-based recommender systems 136

6.2.2 Generalized graph neural networks 137

6.2.3 Clustering with heterogeneous graph neural networks 138

6.2.4 From topic modeling to intelligent document analysis techniques 139

Appendix

A Appendix for Chapter 4 144

A.1 Implementation details and computation time 144

B Appendix for Chapter 5 144

B.1 Implementation details and computation time 144

Bibliography **147**

Notations

Variables

M, N	number of observations
i, j	observation indexes
P	dimension of variables in a latent space
Z	P-dimensional latent embeddings
	Recommender systems
Y	observed rating data in $\mathbb{R}^{M \times N}$
W	document-term matrix encoding text data
R, C	user and product latent variables
	Graph learning
G	network/graph data
V	set of nodes in a network
E	set of edges in a network
F	number of node features
U	dimension of covariate between two nodes
A	graph adjacency matrix in $\{0, 1\}^{N \times N}$
X	node feature/attribute matrix in $\mathbb{R}^{N \times F}$
$\hat{D}$	graph degree matrix
Y	edge feature matrix in $\mathbb{R}^{\lvert E \rvert \times U}$
K	number of clusters
	Topic modeling
D	number of documents
V	number of words in a corpus
T	number of topics
θ	topic proportions

β	word occurrence matrix whose entry is the probability that vocable v occurs in topic t
L	dimension of word and topic embeddings
ρ	word embedding matrix
α	topic embedding matrix

Distributions

$p(\cdot)$	data prior distribution
$q(\cdot)$	approximate variational distribution
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2
$\mathcal{B}(\cdot)$	Bernoulli distribution
$\mathcal{M}(\cdot)$	multinomial distribution

Operators

$\mathcal{KL}(q(\cdot)\|\|p(\cdot))$	Kullback-Leibler divergence between two distributions
$\sigma(\cdot)$	non-linear activation function
$f_\tau(\cdot)$	decoder parametrized by τ in an auto-encoder
$g_\phi(\cdot)$	encoder parametrized by ϕ in an auto-encoder
$\hat{W}$	weight matrix in neural networks
$\oplus$	concatenation operator
$\odot$	element-wise multiplication operator

CHAPTER 1

Introduction

In this thesis, we will go through three common types of data: ordinal data like ratings in recommender systems, graph data with nodes and edges, and text data in count format. Each data type has its own characteristics and potential range of applications. These data can be used independently for a variety of purposes, but they can also be combined together, which leads to the typical problems associated with heterogeneous data analysis.

Let us look at a few straightforward instances first to gain a better intuitive understanding of these data and their applications. Real-world systems can often be modeled as networks, including

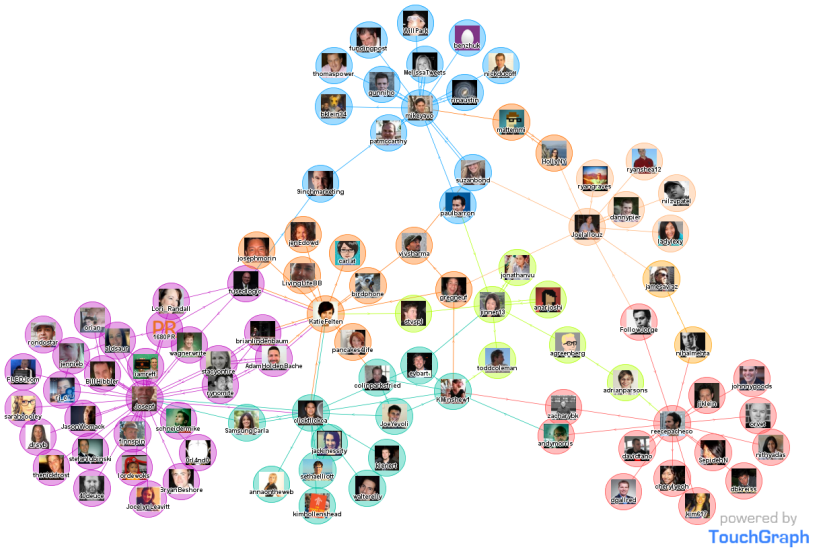


Figure 1.1 – Visualization of Twitter activity data†. Nodes correspond to users in the network, edges represent actions like "follow" or "send to", and colors characterize various social communities.

†. source from https://www.touchgraph.com/news

social networks, biological networks and communication networks. The information from social media, for example, might be seen as a network of friends and followers. They have a significant impact on our lives, from sharing knowledge to influencing others. How we analyze and visualize a network based on these connections and influences is therefore crucial. In Figure 1.1, a simple visualization of a social network is given. By analyzing the connections between people, we may cluster them into various groups, each of which being represented by a distinct color. With this group information, we might be able to predict which person in the network will be followed by a given user in the future, due to the "your friend is my friend" effect; Or for business purposes, if we want to promote a product to a target user in a group, and we know that some members of that group have positive feedbacks, then that user is quite likely to favor the product as well, as friends typically share similar interests.

Moreover, since we regularly purchase goods, watch movies, or book hotels, we are overloaded with e-commerce data in the modern world. This data also provides us a lot of useful information. Figure 1.2 displays an example of the Amazon product review data containing the user ID, reviews on the products, given ratings, item categories, etc. By exploiting the similarity between users and products, recommendation systems are developed to provide recommendations. The system can

	review_title	review_rating	review_date	user_id	brand
1	Best Natural Deodorant!!	5.0	2020-03-06	AGIZFDSHLSBZ7KHQ4FV65W7B3HMQ	Schmidt's Deodorant
2	Schmidt's Charcoal Deodorant is the b	5.0	2020-03-06	AHOZPYWO6WE5MLAZZPDUAWNKKNRQ	Schmidt's Deodorant
3	Beware. Defective or counterfeit.	1.0	2020-03-06	AECC6GYWWLNCKSBG3DZDGFSYDOTA	Schmidt's Deodorant
4	Good for infrequent shavers in cooler	3.0	2020-03-05	AEOO6OOEO4QD7Z4F6C2FMZUCW7PA	Schmidt's Deodorant
5	This isn't the true product and will	1.0	2020-03-02	AEHSDCGRIMVOUG6ST35FJ4FMMW5Q	Schmidt's Deodorant
6	A good choice for deodorant	5.0	2020-03-01	AEBMSOME6YNIFY4XQ7D2YLVWVO2Q	Schmidt's Deodorant
7	Nice Smell	5.0	2020-02-26	AGXV56HW7TKTKEGVIMMUQMBCAFWQ	Schmidt's Deodorant
8	Works OK but stains clothes	2.0	2020-02-26	AEHB3GIOUHLLXJTETQOHBGBGCX6Q	Schmidt's Deodorant
9	My skin hated it	3.0	2020-02-24	AHDNFS55ZAXA3URINNMOCPR35GFA	Schmidt's Deodorant
10	My go to	5.0	2020-02-24	AH5T526ODDURVWQRRAPJY7SDC7HA	Schmidt's Deodorant
11	Yum	5.0	2019-07-07	AGSXTIXZYZ4CNO5NAJXURXFKLAAQ	Sir Kensingtons
12	Yummy!	5.0	2019-02-24	AEAMBNKJXX47K6ZEWAYXTTEAW47Q	Sir Kensingtons
13	one of the best ones	5.0	2019-02-17	AFFLAVXOKYKYQ4XU6RRU44VUJVZA	Sir Kensingtons

Figure 1.2 – Sample of Amazon product review data†. Each row corresponds to details about a user-purchased item, including the user review and rating for the product, the user ID and the product brand, etc.

†. source from https://data.world/datasets/amazon

analyze user preferences based on observed historical records, assisting the user in choosing the appropriate product for future purchases. It also aids the item suppliers in delivering users with proper products. Thus, 35% of Amazon sales are attributable to recommendations. Additionally, it contributes to personalizing the contents, as the fact that most of the movies that people rent on Netflix come from recommendations.

These are just a few of the applications we can imagine that involve these types of data. Since there are many sources of data in the actual world, data analysis and its applications have emerged as crucial research areas. In the following, we primarily concentrate on two tasks : the first is the construction of a recommender system using both rating and textual data, and the second is the clustering of nodes in networks.

1.1 Recommender systems

This section discusses rating data via a simple illustration, and offers a guideline for building recommender systems with this type of data.

1.1.1 Rating data and examples

With the advancement of the Internet era, rating data is ubiquitous on websites such as Amazon, TripAdvisor, or Yelp. Customers can gain valuable information about products and services from rating data, which helps them make decisions. Companies and manufacturers can also benefit from the data by analyzing users satisfaction in order to make product recommendations or to detect functional weaknesses in their products.

In general, rating data involves three types of resources : users, products, and the users ratings for products. Consider a dataset involving M users who are scoring N products. This type of data structure is commonly formalized as an ordinal data matrix Y in $\mathbb{R}^{M\times N}$, such that Y_{ij} corresponds to the rating that the i-th user assigns to the j-th product in practice. This matrix is usually extremely sparse (most of its entries are missing), corresponding to users not scoring nor reviewing some products. Conversely, when a score is assigned, it takes values in $\{1,\dots,H\}$ with $H > 1$ (usually $H = 5$ or 10). Henceforth, we assume that an ordinal scale is consistently defined. For instance, when customers evaluate products, 1 always means "very poor" and H is always associated with "excellent" ratings. The assumption is necessary, otherwise the results obtained

Table 1.1 – An example of rating data for three users and five products. For instance, the user U_1 gave the ratings 5, 4, 4 to products P_1, P_3 and P_4, respectively.

Users \ Products	P_1	P_2	P_3	P_4	P_5
U_1	5		4	4	
U_2		4			5
U_3	3		3		2

when analyzing data would be completely misleading, therefore the analyst should take this point into account when designing the data collection. The number of ordered levels H is assumed to be the same for all (not missing) Y_{ij}. If it is not the case, a scale conversion pre-processing algorithm (see e.g. [Gilula et al., 2019]) can be employed to normalize the number of levels.

An example of rating data is described in Table 1.1. A total of $M = 3$ consumers and $N = 5$ products are present here. Specific user ratings on relevant items are in purple. The blank cells are the missing ratings that we want to predict based on the observed historical information.

1.1.2 What is a recommender system ?

Using the observed rating data, it is natural to develop a recommender system that attempts to forecast the rating that one user would give to an unrated product, allowing one to make relevant product recommendations to the users. At the core of the research on recommendation systems, we point out a widely adopted collaborative filtering (CF) approach [Su and Khoshgoftaar, 2009], which relies on the similarities among the users ratings in the past. It operates by looking through a large group of people and by identifying a smaller group of users who have similar preferences to the target user. It then looks for products that similar users like and combines them to predict a sorted list of recommendations for the target user. An illustration of a CF-based recommender system is shown in Figure 1.3.

Next, by converting the list of users and products into a user-item interaction matrix, such as the previously described ordinal matrix Y, a CF-based recommender system can be considered as completing the missing values in the matrix based on observed entries [Ramlatchan et al.,

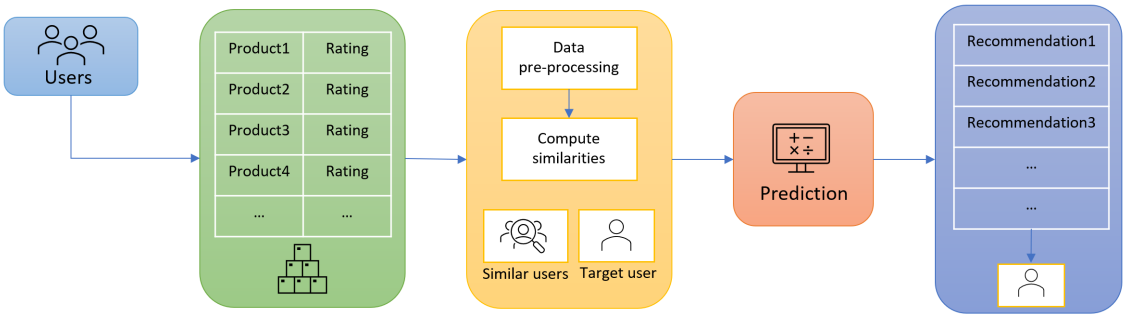


Figure 1.3 – Diagram of pipeline in recommender system. The system collects and examines customer ratings for products before recommending new ones to a target user based on their similarities.

2018]. Matrix factorization [Mnih and Salakhutdinov, 2007; Koren et al., 2009] is a well-known and traditional CF technique for rating predictions, which has shown effectiveness in learning representation of the data. The goal is to accurately profile users and items by breaking down the user-item rating matrix into the user latent (non-observed) factors and item latent factors. For instance, given a rating matrix $Y \in \mathbb{R}^{M \times N}$, it aims at finding two latent factors $R \in \mathbb{R}^{M \times P}$ and $C \in \mathbb{R}^{N \times P}$, where P is the latent space dimension with $P << \min\{M, N\}$. Then, a completed matrix with predictions is obtained as $\hat{Y} = RC^{\top}$. Figure 1.4 shows an intuitive illustration of the matrix factorization paradigm.

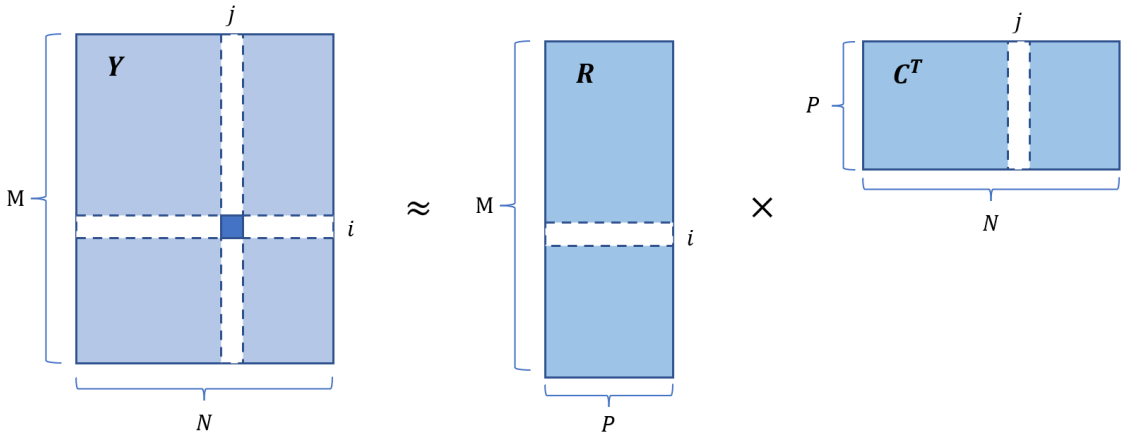


Figure 1.4 – An illustration of matrix factorization.

Finally, by minimizing a loss function between the observed values in Y and their predictions in $\hat{Y}$ (e.g., $||Y - \hat{Y}||^2$), the predictions are forced to be closer to the actual ratings, and the user and product latent factors are optimized as well as updated. Once we have well-learned representations for users and products, values of predictive ratings are directly computed by multiplying the two matrices R and C. In this way, the products with higher ratings are recommended to users, thus achieving the goal of a recommender system.

In this thesis, when developing a recommender system, we make use of the user-item interaction matrix, where each entry corresponds to a user rating of the product. A different approach, on the other hand, considers users and products as nodes in a network or graph, with each link between nodes indicating an interaction between users and products. We will describe the network data format and perform some analysis on this type of data in the section below.

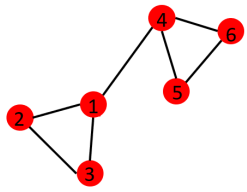
1.2 Network analysis

Network data is introduced in this section, along with a brief overview of the data numerous applications. Among these applications, we concentrate primarily on unsupervised node clustering tasks: true labels of nodes are absent.

1.2.1 Network data and examples

Networks are employed in a wide range of applications, from social media and email communications [Palla et al., 2007] to protein-protein interactions [Barabasi and Oltvai, 2004], because they are simple structures yet are capable of modeling complex systems.

Composed of nodes and edges connecting the nodes, a network or graph is denoted by $G = (V; E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ represents the set of nodes and $E = \{e_1, e_2, \cdots, e_m\}$ represents the set of edges. Typically, nodes are also called vertices or intersections, and edges are also referred to as links or arcs. The general graph representation is a quintuple: $G(V, E, A, X, D)$. $A^{N\times N}$ represents the adjacency matrix of the graph, where $A_{ij} = 1$ if there is a link between node i and node j, 0 otherwise. $X^{N\times F}$ denotes the feature matrix of nodes, for example, in a citation network each node is associated with an article, thus, the matrix X can encode the words used in each publication. $D^{N\times N}$ is the degree matrix which contains information about the number of edges attached to each vertex. N and F represent the number of nodes and the feature dimension



(a) Graph

0	1	1	1	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	0	0	0	1	1
0	0	0	1	0	1
0	0	0	1	1	0

(b) Adjacency matrix

3	0	0	0	0	0
0	2	0	0	0	0
0	0	2	0	0	0
0	0	0	3	0	0
0	0	0	0	2	0
0	0	0	0	0	2

(c) Degree matrix

Figure 1.5 – From left to right: a graph G with six nodes and seven edges; the corresponding adjacency matrix A; the corresponding degree matrix D.

of nodes, respectively. In some cases, the set of edges E can be associated with an additional covariate information, collected into a matrix Y. The generic entry of $Y^{|E|\times U}$, denoted y_{ij}, is a U-dimensional feature associated with the edge connecting i to j. For instance, y_{ij} could encode the text that author i sends to author j in a communication network.

Graphs come in a variety of formats. They can be directed or undirected depending on whether their edges have directions. The attributes of the edges are called weights. Depending on whether an edge includes weights, a graph can be classified as weighted or unweighted. Additionally, there are homogeneous and heterogeneous graphs. An example of an undirected, unweighted, homogeneous graph is shown in Figure 1.5.

Graph-structured data has a wide range of applications in many fields [Wu et al., 2020]. Depending on the level, the graph data can be analyzed either at the node level or at the edge level. Alternatively, we might have a collection of graphs, where one graph is considered as one observation. In this case, the analysis is performed at the graph level. Node classification [Bhagat et al., 2011; Xiao et al., 2022] and node clustering [Aggarwal and Wang, 2010; Malliaros and Vazirgiannis, 2013] are frequently employed applications at the node level. Classification is a standard supervised task with predefined node labels, whilst clustering seeks to group unlabeled nodes based on the node interactivity and features. At the edge level, we mention edge classification [Aggarwal et al., 2016], edge clustering [Cui et al., 2008], and link prediction [Lichtenwalter

et al., 2010; Kumar et al., 2020], i.e. predicting the existence of a link between two nodes in a network. At the graph level, researchers focus on graph classification [Kriege et al., 2020], graph clustering [Bader et al., 2013] and graph similarity analysis [Koutra et al., 2011; Ma et al., 2021]. As mentioned, in contrast to node level applications, which only take into account the nodes in a single graph, graph level applications accept a variety of graphs as input. From another perspective, depending on the field, applications can be classified into multiple categories, including pharmaceutical medicine [Wang et al., 2020a], where biological networks modeling protein-protein interactions, as well as disease networks describing relationships between diseases and biological factors, play a significant role. Other important areas of research include image processing [Ortega et al., 2018], where each pixel in an image can be considered as a node, and natural language processing [Vashishth et al., 2020], where each word in a document is one node and the whole corpus can be modeled by a network.

In this thesis, we focus on node clustering applications. However, these various tasks share some fundamental elements, making it simple to move from one task to another.

1.2.2 Clustering in networks

Unsupervised network analysis has emerged as a crucial sub-field of network analysis since real-world networks may lack knowledge regarding the true labels of each node.

In this context, our interest lies in node clustering which attempts to partition the nodes of the graph into different groups to extract patterns summarizing the data. Clustering is the process of grouping together elements/entities that appear to be closer to each other (than to the remaining elements) based on some similarity metric. In networks, the similarity measure is typically determined based on topological criteria, such as the graph structure, or according to the node locations. Nodes that are regarded to be similar based on this similarity are grouped together into clusters. Thus, each cluster consists of elements that share common characteristics. For instance, the graph in Figure 1.5a can be separated into two clusters based on their node connections and locations, as illustrated in Figure 1.6. The clusters are represented by different colors, and in each cluster, nodes have more internal connections and less external connections.

A long series of statistical methods [Schaeffer, 2007; Snijders, 2011] have been developed to discover the underlying communities in networks by learning the latent features of graph-structured data. More recently, deep learning-based models have emerged as a promising approach

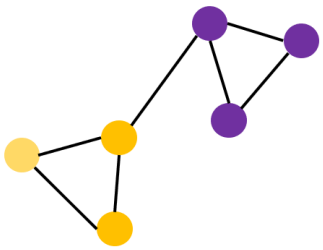


Figure 1.6 – A partition of nodes into two clusters.

for analyzing large-scale networks and they have shown their abilities for representation learning purposes on data with complex structures [Hamilton et al., 2017; Zhang et al., 2018]. Based on the learned graph embeddings, numerous deep methods have been proposed to achieve node clustering using external algorithms like K-means [Hartigan and Wong, 1979] or GMMs [Reynolds, 2009]. Furthermore, many efforts have been put in order to conduct the clustering in an end-to-end manner. We will go through some state-of-the-art methodologies in the following chapter.

1.3 Text analysis

In addition to the rating or network data we have already discussed, textual data is also quite prevalent in daily life. Both rating data as well as network data might be accompanied by textual information. Indeed, with the development of the Internet and the growth of social media, massive amounts of textual data are generated every day in the form of blogs, tweets, comments, and surveys. Additionally, the majority of consumer transactions now take place digitally, creating yet another enormous collection of texts. Most text information is unstructured and scattered across the web. Valuable knowledge can be gained from the textual data if it is properly collected, organized and analyzed.

1.3.1 Text data and examples

Let us look at some concrete examples : first, in recommender systems, many consumers also use texts to express various opinions in addition to ratings. Compared to a single score, reviews

Table 1.2 – A sample of Amazon food data.

ReviewerID	ProductID	Rating	ReviewText
A31N6KB160O508	B000G6RYNE	4	Pretty good tasting chip
A1CBNUBPZPWH5D	B0016FY6H6	5	Flavorful and refreshing
A1XG5WYLFMRRX1	B001LG945O	1	Artificial tasting
AQ6SHOW0VMZGF	B003VXFK44	2	Really disappointed

can contain crucial information from different aspects about the products. Table 1.2 demonstrates a sample from the Amazon product data set [He and McAuley, 2016], which includes product reviews and meta-data (users, products, ratings and texts) from Amazon. As we can see, a high score corresponds to a text review containing positive words, whereas a low rating is associated with some negative expression as "disappointed". Therefore, using these information can enhance the system capacity for recommendations. Second, in citation networks where each node represents a publication, the networks also contain some text information such as the title and abstract of each article. This additional textual data is a crucial source of information for graph learning tasks.

Then, how to encode the text ? One option is to rely on the frequency counts, such as the number of occurrences of distinct words in a corpus. For instance, the textual reviews described above can be encoded as in Table 1.3, displaying a traditional bag-of-words representation in a

Table 1.3 – Review representation in bag-of-words models.

ReviewText	pretty	good	tasting	···	disappointed
Pretty good tasting chip	1	1	1	···	0
Flavorful and refreshing	0	0	0	···	0
Artificial tasting	0	0	1	···	0
Really disappointed	0	0	0	···	1

document-term matrix. Each review in the Amazon data is now represented as a vector of its word counts.

A count variable is discrete because it consists of non-negative integers. Nonetheless, there is not one specific probability distribution that fits all count data sets. In the statistical community, count data is frequently subjected to probabilistic assumptions about the generative model: Poisson [Coxe et al., 2009], negative-binomial models [Greene, 2008], etc. More recently, the probabilistic treatment of data can be done via deep learning by using the neural networks to encode the latent parameters of probability distributions [Gan et al., 2015; Srivastava and Sutton, 2017] and fitting all of the weights through stochastic gradient descent [Bottou, 2010, 2012].

1.3.2 Challenges in text analysis

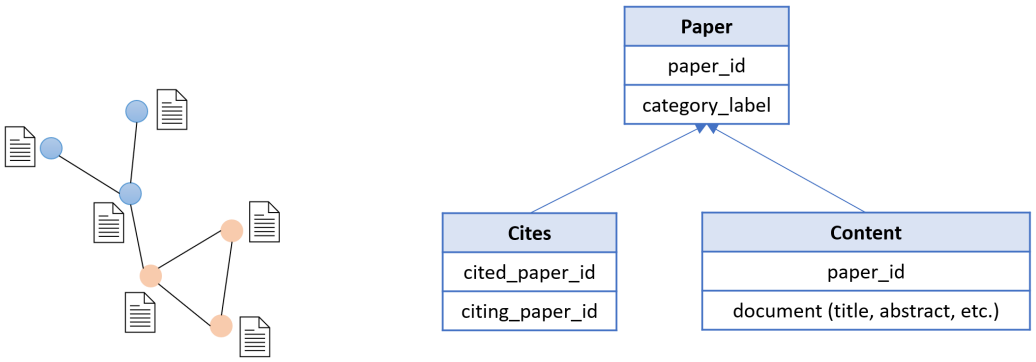
Text analysis refers to the representation, processing and modeling of text data in order to derive useful insights. This can be very challenging in practical applications. One challenge of text analysis lies in the high dimensionality of the data, another in the fact that text is primarily unstructured data. As a result, high-dimensional unstructured text must be carefully processed so that it has a specific structure for future analysis. Common text analysis techniques include content analysis [Stemler, 2000; Grimmer and Stewart, 2013], bag-of-words representation [Zhang et al., 2010] and natural language processing (NLP) [Chowdhary, 2020]. Content analysis and other dictionary-based methods are often performed by counting the frequency of words/phrases in a particular text. Following this approach, text data is compressed into phrase frequencies, and the index can be used to respond to more quantitatively focused research queries. Bag-of-words representation is a simple and common approach widely used in text analysis problems. It represents a document as a set of words, thereby transforming the document into a high-dimensional vector indicating the presence/absence of each word in the document. Last but not least, text analysis that uses natural language processing is frequently the most automated since this approach simulates how humans understand and process language. In contrast to bag-of-words, other NLP strategies considers word order to be important. In this thesis, we focus on the bag-of-words representation to transform the observed textual data into a document-term matrix, since it is a straightforward and typical method in text analysis and is simple to manipulate in numerous applications.

1.4 The need for methods to analyze heterogeneous data

As we saw in previous sections, different types of data exist and might occur simultaneously in applications. In this section, we discuss the rise of heterogeneous data and the necessity for techniques to analyze them.

1.4.1 Context

Heterogeneity is a key characteristic of modern data sets. Any data set with high variability of data types and formats (ordinal, categorical, count, etc.) is considered heterogeneous. The heterogeneous data, which originates from various and different sources, naturally possesses a wide range of categories and representational patterns. For example, scientific citation networks typically contain the following information : the paper ID and its category, the citation relationship with other papers, as well as a document that includes content about each article such as titles and abstracts. Figure 1.7 provides an illustration of a citation network consisting of three different sources of information. The citation relationships can be modeled as links in a graph, each article is associated with a document containing textual contents, and papers are also assigned a category



(a) A citation network example

(b) Information from the network

Figure 1.7 – Each node in a citation network represents a scientific paper, and the different colors indicate various categories. Each paper is associated with a document including its title, abstract, etc. Three different sources of information reflect paper categories, citation relationships between articles and text content.

label. Furthermore, in some recommender system applications, numerical ratings are usually given by users on products along with textual reviews. Under this context, it is important but challenging to combine different kinds of data to meet specific requirements. As a consequence, how to process and integrate the heterogeneous data has emerged as a critical issue in data analysis.

1.4.2 The rise of deep probabilistic methods

Heterogeneous data is frequently complex and multidimensional. Traditional data analysis approaches are sometimes ineffective as they can only model very simple data distributions. Recent achievements have been made by deep latent variable models (DLVMs) which combine probabilistic modeling and deep learning techniques. A deep probabilistic model should be capable of identifying the underlying structure of the data, such as hidden patterns or presence of clusters, and of generating new data with similar distributions to the observed ones. In the following are some examples of problems that we aim at tackling in this thesis :

— Consider some data collected from an e-commerce system like Amazon, we may want to create a recommender system that takes into account both the ordinal ratings of the products and textual reviews given by the users. A deep generative model can be used to discover the underlying communities of customers who share similar interests and to predict the missing ratings.

— Given a scientific citation network that includes the citation links between papers as well as the content of each publication, we may want to discover the latent topics that emerged from the documents and to divide the papers into several groups depending on the hidden features, with the aid of a deep probabilistic model.

These are simply a few illustrations. These types of data analysis are common in many scientific, economic, and industrial sectors, and deep probabilistic methods have been widely employed to address them. In the following part, we first discuss latent variable models (LVMs) and then explore how deep neural networks (DNNs) can be incorporated to estimate their parameters.

Latent variable models (LVMs) [Bishop, 1998] are a flexible tool for exploring data heterogeneity and have been widely used for jointly modeling mixed data based on latent patterns summarized from the observed data [Blei, 2014]. Given a set of high-dimensional data points x, learning the complicated distribution $p(x)$ is a challenging task in machine learning. Rather than directly modeling $p(x)$, we can introduce an unobserved latent variable z and define a conditional

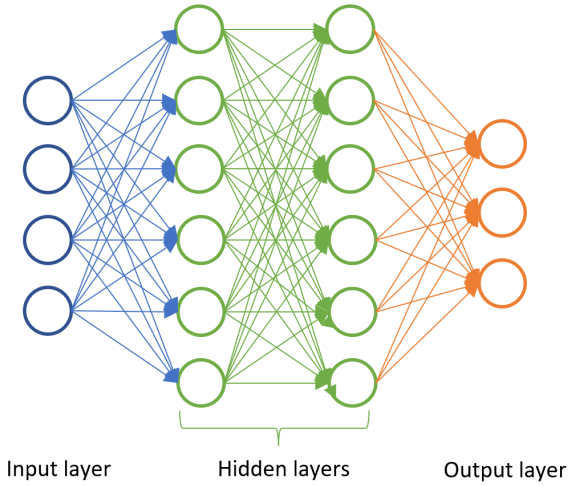


Figure 1.8 – Architecture of a neural network.

distribution $p(x|z)$ for the data. The posterior $p(z|x)$ for real-world data is typically intractable, and traditional statistical methods rely on Markov Chain Monte Carlo (MCMC) to sample from this distribution. Despite this, these models require substantial computational costs and are difficult to scale to large data sets.

Many recent works extend LVMs with deep neural networks (DNNs) due to the difficulty in the inference procedure and the limitation of linearity in probabilistic models. By combining the statistical basis of generative models with the approximation capabilities of deep learning techniques, deep latent variable models (DLVMs) enable us to handle massive and complex data accounting for non-linearity, in many fields. What exactly is the magic of deep learning ? In general, deep learning is a branch of machine learning that uses artificial neural network as the architecture to perform representation learning of data. Neural networks are composed of multiple neuron layers, including an input layer, one or more hidden layers, and an output layer. Each layer contains a number of artificial neurons. Figure 1.8 depicts the architecture of a simple neural network : it has four neurons at the input layer, two hidden layers are equipped with six neurons each and there are three neurons at the output layer. The number of neurons and the depth of layers can be modified depending on the task. Then, how do neural networks work ? Mathematically, the operation of passing messages through l layers can be thought of as a combination of multiple non-linear functions, where $F = \sigma(f_1) \circ \sigma(f_2) \circ \cdots \circ \sigma(f_l)$. The function σ is a non-linear activation function, such as ReLU ($\sigma(x) = \max(0, x)$) or Sigmoid ($\sigma(x) = 1/(1 + e^{-x})$). The non-linear factor

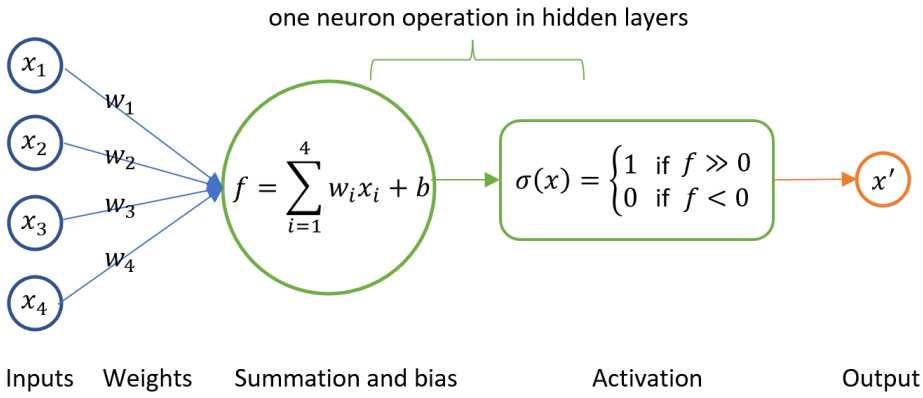


Figure 1.9 – An illustration of the operation of artificial neurons.

added by the activation function can overcome the limitation of insufficient expression ability of the linear model when the data structure is complicated. An illustration of the operation of neurons is shown in Figure 1.9. A simple activation is chosen in this network, which performs binary classification to output either a value of 1 or 0.

DLVMs rely on variational inference (VI) [Blei et al., 2017] to approximate the intractable probability through optimization. The main idea behind VI is to produce an approximation $q(z)$ as close to the true posterior $p(z|x)$ as possible by minimizing the Kullback-Leibler divergence (denoted by $\mathcal{KL}$) [Csiszár, 1975] between the two distributions. Furthermore, it is generally assumed that $q(z)$ comes from a family of tractable distributions. For example, it is defined as a Gaussian distribution $\mathcal{N}(z; \mu, \sigma^2 I)$ in variational auto-encoders (VAEs) [Kingma and Welling, 2014b, 2019], where the mean μ and the variance σ^2 of the approximate posterior are parametrized by an encoding neural network. The optimization of parameters is carried out automatically through stochastic gradient descent [Bottou, 2012; Kingma and Ba, 2014]. The use of DNNs accelerates the inference process and scales up to massive and complex data. More details about the inference procedure in DLVMs are described in Section 2.1.

1.5 Organization of the thesis

The structure of this thesis is:

— Chapter 2 reviews some basis of DLVMs and state-of-the-art models relevant to the problems;

— Chapter 3 presents our recommendation algorithm that takes into account both user ratings and textual reviews on products;

— Chapter 4 describes our approach to achieve an end-to-end node clustering and to better preserve the network topology;

— Chapter 5 proposes a document similarity-based graph convolutional network and introduces our methodology which combines word embeddings, topic modeling, and graph topology structure to accomplish clustering;

— Chapter 6 summarizes our contributions and leads to perspectives for the future work.

CHAPTER 2

Statistical and deep learning models for recommender systems and node clustering

This chapter reviews some fundamentals of deep latent variable models (DLVMs) and state-of-the-art methods related to the problems we mentioned in the previous chapter.

2.1 Warmup : a few fundamental DLVMs 21
2.1.1 Variational auto-encoders for continuous data 21
2.1.2 Topic modeling for count data 24
2.2 Construction of recommender systems 26
2.2.1 Matrix factorization models 26
2.2.2 Latent factor models for recommender systems 27
2.2.2.1 Rating-based recommender systems 27
2.2.2.2 Rating-with-text based recommender systems . . 28
2.2.3 Deep learning-based recommender systems 30
2.3 Network data analysis 32
2.3.1 Probabilistic graphical models 32
2.3.2 Graph neural networks 34
2.3.3 Deep probabilistic models for node clustering 37
2.4 Clustering in heterogeneous information networks 40

2.1 Warmup : a few fundamental DLVMs

In this section, we will introduce a few crucial DLVMs whose understanding is important in order to fully appreciate our contributions.

2.1.1 Variational auto-encoders for continuous data

We now introduce an important class of deep generative models : the variational auto-encoders (VAEs) [Kingma and Welling, 2014a; Rezende et al., 2014]. We first recall some basic facts about variational inference in mixture models before introducing the usage of neural networks in VAEs.

Variational inference in mixture models. Assume that $x = \{x_1, \cdots, x_m\}$, with $x_i \in \mathbb{R}^N$ are observations and $z = \{z_1, \cdots, z_m\}$, with $z_i \in \mathbb{R}^P$ are hidden variables. The main goal of variational approaches is to select a family of distributions $q(z)$ over the latent variables to approximate the true posterior $p(z|x)$ when it is not tractable. The closeness of the two distributions is measured by the Kullback-Leibler (KL) divergence, coming from information theory [Kullback, 1997]. The KL divergence is defined as :

$$\mathcal{KL}(q(z)||p(z|x)) = \mathbb{E}_{q(z)}\left[\log \frac{q(z)}{p(z|x)}\right]. \tag{2.1}$$

It can easily be shown that minimizing Eq. (2.1) with respect to $q(z)$ is equivalent to maximize the evidence lower bound (ELBO), obtained by applying the Jensen's inequality to the log-likelihood of the observed data :

$$\begin{aligned}
\log p(x) &= \log \int_z p(x,z) dz \\
&= \log \int_z p(x,z) \frac{q(z)}{q(z)} dz \\
&= \log \mathbb{E}_{q(z)}\left[\frac{p(x,z)}{q(z)}\right] \\
&\geq \underbrace{\mathbb{E}_{q(z)}[p(x,z)] - \mathbb{E}_{q(z)}[q(z)]}_{\text{ELBO}}.
\end{aligned} \tag{2.2}$$

Indeed, if we develop the KL divergence further, we get :

$$\begin{aligned}
\mathcal{KL}(q(z)||p(z|x)) &= \mathbb{E}_{q(z)}[\log q(z)] - \mathbb{E}_{q(z)}[\log p(z|x)] \\
&= \mathbb{E}_{q(z)}[\log q(z)] - \mathbb{E}_{q(z)}[\log p(x,z)] + \log p(x) \\
&= -(\mathbb{E}_{q(z)}[p(x,z)] - \mathbb{E}_{q(z)}[q(z)]) + \log p(x),
\end{aligned} \tag{2.3}$$

which is the negative ELBO plus the log marginal probability of x, not depending on $q(z)$. Thus, the log-likelihood for the observed data x could be written as:

$$\log p(x) = \mathbb{E}_{q(z)}\left[\log \frac{p(x,z)}{q(z)}\right] + \mathcal{KL}(q(z)||p(z|x)), \tag{2.4}$$

and the ELBO can be further developed as:

$$\begin{aligned} \text{ELBO} &:= \mathbb{E}_{q(z)}[\log p(x,z)] - \mathbb{E}_{q(z)}[\log q(z)] \\ &= \mathbb{E}_{q(z)}[\log p(x|z)] + \mathbb{E}_{q(z)}[\log p(z)] - \mathbb{E}_{q(z)}[\log q(z)] \\ &= \mathbb{E}_{q(z)}[\log p(x|z)] - \mathcal{KL}(q(z)||p(z)), \end{aligned} \tag{2.5}$$

where the first term of ELBO aims at maximizing the reconstruction likelihood, the second term encourages the learned distribution $q(z)$ to be similar to the prior distribution $p(z)$ and acts as a regularization term.

Generative model for VAEs. VAEs make some additional assumptions on $p(x|z)$ and $q(z)$, that we now detail. Firstly, the latent variables are assumed to be drawn from Gaussian distributions:

$$p(z_i) = \mathcal{N}(0, I_P), \tag{2.6}$$

where I_P denotes the identity matrix in $\mathbb{R}^P$. However, other choices are of course possible [Makhzani et al., 2016; Rolfe, 2017].

Next, VAEs also suppose that:

$$p(x|z) = \prod_{i=1}^{m} h(x_i; f_\tau(z)), \tag{2.7}$$

where $h(\cdot)$ denotes here a generic probability distribution function with parameters obtained via $f_\tau(\cdot)$, which is a fully-connected neural network, parametrized by τ. A graphical representation of the generative model can be seen in Figure 2.1.

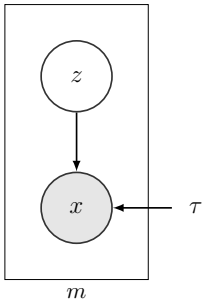


Figure 2.1 – Graphical representation of the generative model for VAEs.

Lastly, in order to maximize the ELBO, it is further assumed that:

$$q(z_i) = \mathcal{N}(z_i; \mu_i, \sigma_i^2 I_P), \tag{2.8}$$

where $[\mu_i, \sigma_i^2] = g_\phi(x_i)$ is the output of another neural network $g_\phi(\cdot)$, parametrized by ϕ.

With all assumptions (2.6)-(2.7)-(2.8), the ELBO in Eq. (2.5) can be further developed as:

$$\begin{aligned} \text{ELBO} &:= \mathbb{E}_{q(z)}[\log(p(x|z)] - \mathcal{KL}(q(z)||p(z)) \\ &= \sum_{i=1}^{m} \left[\mathbb{E}_{q(z)}[\log h(x_i; f_\tau(z))] - \mathcal{KL}\left(\mathcal{N}(z_i; \mu_i, \sigma_i^2 I_P)||\mathcal{N}(z_i; 0_P, I_P)\right)\right]. \end{aligned} \tag{2.9}$$

From the deep learning perspective. VAEs can be seen as an encoder-decoder architecture from the standpoint of deep learning, where the encoder compresses the input $x \in \mathbb{R}^N$ into a latent variable $z \in \mathbb{R}^P$ sampled from the variational probability distribution and lying in a low-dimensional latent space. Then, the decoder maps the latent component back into a reconstructed output y, as shown in Figure 2.2.

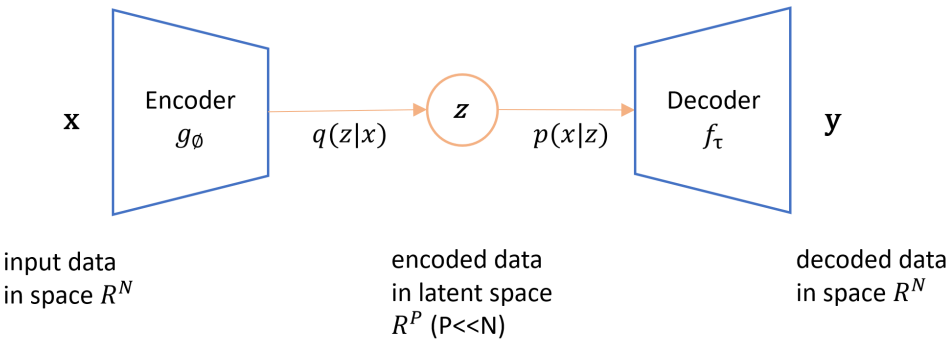


Figure 2.2 – A deep-learning view of VAEs.

To perform the optimization, the training loss is equivalent to the negative ELBO where the first term in Eq. (2.5) is a traditional reconstruction loss and the second term is a regularization loss. The neural nets parameters ϕ and τ are optimized through stochastic gradient descent.

VAEs are flexible architectures since they can be designed to conduct a variety of encoding-decoding tasks by properly choosing the prior and the observed data distributions. In Chapters 3 to 5, we will introduce three original generative models based on the VAE structure and demonstrate their applicability.

2.1.2 Topic modeling for count data

Topic modeling is a statistical analysis technique for revealing the underlying semantic structure in a collection of documents. The hidden groups of words are referred to as "topics," and each word in a document is associated with one or more topics. In the following, we review three commonly used probabilistic approaches for topic modeling.

Latent Dirichlet allocation. Latent Dirichlet allocation model (LDA) [Blei et al., 2003] is a well-known probabilistic model on which many methodologies have been built. Being an unsupervised learning approach, it does not require a manually labeled data set for training. A collection of documents and a specified number of topics are only needed. LDA assumes that the words in a document are sampled from a mixture distribution over latent topics. Consider a corpus of D documents $\{w_1, \cdots, w_D\}$, it posits T topics $\beta_{1:t}$ with a vocabulary that includes V unique terms. Let $w_{dm} \in \{1, \cdots, V\}$ denote the m-th word in the d-th document, the generative process of LDA is:

— Draw the topic proportions $\theta_d \sim \text{Dirichlet}(\alpha)$, with $\alpha \in \mathbb{R}^{+,T}$;

— For the m-th word in the d-th document:

(i) Sample its topic $z_{dm} \sim \mathcal{M}(1, \theta_d)$;

(ii) Sample word $w_{dm} \sim \mathcal{M}(1, \beta_{z_{dm}})$, with $\beta_t = (\beta_{t1}, \cdots, \beta_{tv})^\top$, $\sum_{v=1}^{V} \beta_{tv} = 1$, $\beta_{tv} > 0$.

The marginal likelihood for the document w_d which contains M_d words can be developed as:

$$p(w_d|\alpha, \beta) = \int_{\theta_d} \Big(\prod_{m=1}^{M_d} \sum_{z_{dm}=1}^{T} p(w_{dm}|z_{dm}, \beta) p(z_{dm}|\theta_d) \Big) p(\theta_d|\alpha) \mathrm{d}\theta_d. \tag{2.10}$$

Sampling techniques, such as collapsed Gibbs sampling [Teh et al., 2006], and variational techniques [Srivastava and Sutton, 2017] are the main inference tools for LDA. A graphical representation of LDA is shown in Figure 2.3.

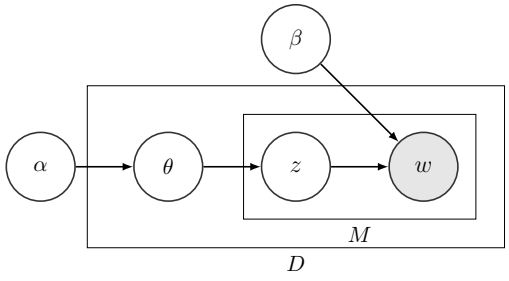


Figure 2.3 – Graphical representation of LDA.

Latent Dirichlet allocation with products of experts. Due to the difficulty in working with discrete variables like z_{dm}, Srivastava and Sutton [2017] proposed to collapse z_{dm} in Eq. 2.10 and simplifying the formulas into:

$$p(w_d|\alpha,\beta) = \int_{\theta_d} (\prod_{m=1}^{M_d} p(w_{dm}|\beta,\theta_d))p(\theta_d|\alpha)\mathrm{d}\theta_d, \tag{2.11}$$

where the variable z_{dm} is summed out and the distribution $p(w_{dm}|\beta,\theta_d)$ becomes a mixture of multinomial $\mathcal{M}(1,\beta\theta_d)$. However, this assumption may lead to the appearance of some topics that are of poor quality, which is a drawback shared by all mixture models [Hinton and Salakhutdinov, 2009].

To tackle this problem, Srivastava and Sutton [2017] also introduced a novel topic model called ProdLDA. ProdLDA develops a logistic normal variational distribution over θ_d with diagonal covariance during the inference procedure, where the mean $\mu_d = f_\mu(w_d,\phi)$ and the diagonal covariance $\Sigma_d = \mathrm{diag}(f_\Sigma(w_d,\phi))$ are obtained by two neural networks f_μ and f_Σ, parametrized by ϕ. Contrary to LDA, the topic proportions are generated from $q(\theta_d)$ by sampling $\delta_d \sim \mathcal{N}(0,I)$ and computing $\theta_d = \sigma(\mu_d + \Sigma_d^{\frac{1}{2}}\delta_d)$, with $\sigma(\cdot)$ a softmax function. Furthermore, by replacing the mixture assumption at the word level in LDA with a weighted product of experts, it posits a likelihood:

$$w_{dm}|\beta,\theta_d \sim \mathcal{M}(1,\sigma(\beta\theta_d)), \tag{2.12}$$

where the topic proportions θ_d lie in the simplex, whereas the constraint on the topic-word probability matrix β is relaxed, yielding a considerable improvement in topic coherence.

Embedded topic model. The embedded topic model (ETM) [Dieng et al., 2020] is a recent development that combines traditional topic modeling with word embeddings to discover the latent semantic structure of documents while also learning a good representation for the vocabulary. It includes two latent dimensions. First, it represents each document in terms of T latent topics, as in LDA. Second, it embeds words in the vocabulary in a L-dimensional latent space as word embeddings. The generative process for ETM is:

— Draw the topic proportions $\theta_d \sim \mathcal{LN}(0,I_T)$, independently, for each document in the corpus;

— For the m-th word in the d-th document:

(i) Sample its topic $z_{dm} \sim \mathcal{M}(1,\theta_d)$;

(ii) Sample word $w_{dm} \sim \mathcal{M}(1, \sigma(\rho^\top \alpha_{z_{dm}}))$,

where $\mathcal{LN}$ denotes the logistic-normal distribution and a draw θ_d from this distribution is produced as:

$$\delta_d \sim \mathcal{N}(0, I_T), \qquad \theta_d = \text{softmax}(\delta_d).$$

Additionally, ρ is a $L \times V$ word embedding matrix where V is the total number of words in the vocabulary. The t-th topic embedding in the semantic space of words is indicated by the vector $\alpha_t \in \mathbb{R}^L$, with $t \in \{1, \cdots, T\}$. $\beta = \sigma(\rho^\top \alpha_t)$ is a matrix whose entry β_{tv} represents the probability that word v occurs in topic t, where $\sigma(\cdot)$ denotes a softmax function so that $\sum_{v=1}^{V} \beta_{tv} = 1$. Given a corpus of documents where the d-th document is a collection of M_d words, the marginal log-likelihood of ETM is described as:

$$\mathcal{L}(\alpha, \rho) = \sum_{d=1}^{D} \log p(w_d | \alpha, \rho). \tag{2.13}$$

The marginal likelihood of each document, $p(w_d|\delta, \alpha, \rho)$, is intractable since it involves a difficult integral over the topic proportions:

$$p(w_d|\alpha, \rho) = \int p(\delta_d) \prod_{m=1}^{M_d} p(w_{dm}|\delta_d, \alpha, \rho) \mathrm{d}\delta_d. \tag{2.14}$$

In addition, the likelihood of the m-th word in the d-th document is:

$$p(w_{dm}|\delta_d, \alpha, \rho) = \sum_{t=1}^{T} \theta_{dt} \beta_{t w_{dm}}. \tag{2.15}$$

Finally, ETM adopts a variational inference to estimate the ELBO and optimize the model parameters α and ρ through stochastic gradient descent.

2.2 Construction of recommender systems

In this section, we review some popular state-of-the-art techniques to produce recommender systems, that were introduced, in very general terms in Section 1.1.

2.2.1 Matrix factorization models

Matrix factorization [Koren et al., 2009] is widely employed in collaborative filtering to determine "possible connection" between users and items in recommender systems. Assume for

instance that we want to forecast how a user would evaluate a product based on both the ratings that other users give to the product and the ratings that he gives to other products. Then, given a user-item rating matrix $Y \in \mathbb{R}^{M\times N}$ with M the number of users and N the number of items, it could be factorized into a latent-user matrix $R \in \mathbb{R}^{M\times P}$ and a latent-product matrix $C \in \mathbb{R}^{N\times P}$, where $P << \min\{M, N\}$ is the latent factor size. Through these two matrices, rating predictions are obtained by $\hat{Y} = RC^\top$. The objective function is defined by minimizing the mean squared error between predicted scores $\hat{Y}$ and actual ratings Y :

$$\underset{R,C}{\operatorname{argmin}} \left[\sum_{i=1}^{M} \sum_{j=1}^{N} (Y_{ij} - \hat{Y}_{ij})^2 \mathbb{1}\{Y_{ij} \text{ observed}\} \right]. \tag{2.16}$$

If the rating Y_{ij} is missing, the imputation is performed via $\hat{Y}_{ij}$. In addition, the number of latent factors P is a hyperparameter that need to be tuned properly to obtain optimal matrices R and C.

On the one hand, matrix factorization acts as a dimensionality reduction technique. On the other hand, it can effectively address the issue of matrix sparseness brought on by an excessive number of users and items. A long series of methods based on matrix factorization have been proposed, we cite here for example, non-negative matrix factorization (NMF) [Lee and Seung, 1999, 2000], and probabilistic matrix factorization (PMF) [Mnih and Salakhutdinov, 2007; Salakhutdinov and Mnih, 2008], etc.

2.2.2 Latent factor models for recommender systems

Considering the data types used to develop recommender systems, we briefly categorize existing latent factor methods into two groups : *rating-based* and *rating-with-text based*.

2.2.2.1 Rating-based recommender systems

Most algorithms that have been proposed in the literature only rely on the available ratings. For instance, hierarchical Poisson matrix factorization (HPF) [Gopalan et al., 2015] assumes that the observed rating matrix is drawn from a Poisson distribution :

$$\hat{Y}_{ij} | R_i, C_j \sim \text{Poisson}(R_i^\top C_j), \tag{2.17}$$

where user and item latent factors are modeled by a Gamma prior distribution. Once the posterior $p(R_{1:M}, C_{1:N}|Y)$ has been fitted using variational inference, user ratings for unconsumed products

can be estimated via the posterior expectation $\mathbb{E}[R_i^\top C_j|Y]$. We notice that HPF is a variant of probabilistic matrix factorization, but with positive weights for each user and item and a Poisson distribution in place of a Gaussian distribution.

Similar to HPF, hierarchical compound Poisson factorization (HCPF) [Basbug and Engelhardt, 2016] adopts the conjugated Gamma-Poisson structure to model latent factors. Contrary to HPF, HCPF introduces a zero-truncated compound Poisson random variable to account for sparsity in the data. Moreover, HCPF allows the user to select the response model from a variety of additive exponential dispersion models (EDMs), which includes Normal, gamma, inverse Gaussian, Poisson, binomial, negative binomial, and zero truncated Poisson (ZTP) distributions. The generative process in HCPF is :

— Sample count $n_{ij} \sim \text{Poisson}(R_i^\top C_j)$,

— Sample the response $\hat{Y}_{ij} \sim p_\Phi(\theta_{ij}, n_{ij}k)$ when $n_{ij} \neq 0$,

where $p_\Phi(\theta_{ij}, n_{ij})$ is an additive EDM choosing from one of the seven distributions above, θ and k are two model parameters.

However, HCPF assumes that the natural parameter θ_{ij} is the same for all observations. Therefore, the data-generating model is a fixed distribution. As an extension, coupled compound Poisson factorization (CCPF) [Basbug and Engelhardt, 2017] was introduced recently by coupling HPF with an arbitrary data-generating model among three different methods : mixture models, linear regression and matrix factorization. The generative process in CCPF is :

— Sample count $n_{ij} \sim \text{Poisson}(R_i^\top C_j)$,

— if n_{ij} is 0, then $\hat{Y}_{ij}$ is missing,

— else

(i) Sample the parameter θ_{ij} from the data-generating model,

(ii) Sample the observation $\hat{Y}_{ij} \sim p_\Phi(\theta_{ij}, \phi(n_{ij})k)$, with ϕ a linkage function.

In particular, one possible linkage function is exponential. Setting $\phi(n_{ij}) = 1 - c + cn_{ij}$ with $c > 0$ implies that, as the probability of non-missingness increases, we expect a greater dispersion. Fixing $c = 1$, we get a standard HCPF model.

2.2.2.2 Rating-with-text based recommender systems

Since the product ratings are usually paired with text reviews, another set of recommender systems exploit both ratings and texts to improve the predictions. For example, CTR [Wang and

Blei, 2011] relies on probabilistic matrix factorization (PMF) to generate ratings, and LDA [Blei et al., 2003] is used to model the topics in reviews. Standard PMF is as follows :

— Draw user latent vector $R_i \sim \mathcal{N}(0, \lambda_R^{-1} I)$,

— Draw item latent vector $C_j \sim \mathcal{N}(0, \lambda_C^{-1} I)$,

— Draw the response $\hat{Y}_{ij} \sim \mathcal{N}(R_i^\top C_j, c_{ij}^{-1})$,

where λ_R and λ_C are regularization parameters, c_{ij} is the precision parameter that acts as a rating confidence, a larger value indicates we trust $\hat{Y}_{ij}$ more.

Finally, to combine collaborative filtering and topic modeling, CTR assumes that the item latent vector is generated by :

$$C_j = \epsilon_j + \theta_j, \qquad \text{with} \quad \epsilon_j \sim \mathcal{N}(0, \lambda_C^{-1} I), \tag{2.18}$$

where θ_j denotes the topic proportions in LDA. Finally, a predicted rating is drawn from $\hat{Y}_{ij} \sim \mathcal{N}(R_i^\top C_j, c_{ij}^{-1})$ using this new item factor.

In the estimation procedure, CTR first determines the topic proportions θ via traditional LDA inference method, then PMF parameters are optimized depending on the outcomes θ. In order to conduct the estimations of the LDA and the PMF parts simultaneously, the online Bayesian inference algorithm for CTR model (obi-CTR) [Liu et al., 2017] realizes a joint optimization with an online approach where the two components can reinforce each other during learning. Nonetheless, neither of these models formulates any relationship between the user latent factor and latent topics.

The hidden factors topic model (HFT) [McAuley and Leskovec, 2013] integrates latent rating factors $\gamma_i \in \mathbb{R}^T$ (similar to R_i) with latent review topics θ_i in dimension T by specifying a transformation :

$$\theta_{it} = \frac{\exp(k\gamma_{it})}{\sum_{t'=1}^{T} \exp(k\gamma_{it'})}, \tag{2.19}$$

where the parameter k determines the "peakiness" of the transformation intuitively. A large value of k indicates that users only discuss the most important topic, whereas a small value means that users discuss all topics evenly. The objective function of HFT was then defined with the first term accounting for the rating reconstruction error and a penalization term involving the corpus likelihood. However, HFT is constrained by the requirement that the dimension of latent factors

should be equal to the number of latent topics T. This one-to-one correspondence relationship may not be ideal and restricts the flexibility of the approach.

The aspect-aware latent factor model (ALFM) [Cheng et al., 2018] breaks this limitation by associating latent factors with different aspects and each aspect is represented as a probability distribution of latent topics. For instance, for the "food" aspect, a related latent topic could be "breakfast" or "Italian cuisine". The overall rating is computed through a linear combination of all the aspect ratings :

$$\hat{Y}_{ij} = \sum_{a} \rho_{ija} Y_{ija} + b_i + b_j + b_0, \tag{2.20}$$

where ρ_{ija} denotes the importance of each aspect a, aspect rating Y_{ija} reflects the satisfaction of a user i towards an item j on the aspect a. Then, b_0 is the average rating, b_i and b_j are the user and item biases, respectively. Unfortunately, it turns out that the performance of ALFM is affected by a dispersed data distribution, it only functions properly when the data is concentrated around the mean, as a result of the addition of the average score b_0.

2.2.3 Deep learning-based recommender systems

Deep learning-based approaches have recently shown a remarkable potential on feature representations learning and have been extensively explored in the literature of recommender systems [Khan et al., 2021]. Among the state-of-the-art techniques, the deep cooperative neural network (DeepCoNN) [Zheng et al., 2017] relies on two convolutional neural networks (CNNs) to learn latent representations of users and products from user review text and item review text, respectively :

$$\begin{aligned} R_i &= \text{CNN}_R(text_i), \\ C_j &= \text{CNN}_C(text_j), \end{aligned} \tag{2.21}$$

where the latent factor of the i-th user is based on the text that he used, and the latent factor of the j-th item is depending on the text that it received. The ratings $\hat{Y_{ij}}$ are then generated by a concatenation (denoted by the symbol $\oplus$) of the two latent representations and feeding them into a factorization machine (FM) [Rendle, 2010] :

$$\begin{aligned} Z_{ij} &= R_i \oplus C_j, \\ \hat{Y}_{ij} &= \text{FM}(Z_{ij}). \end{aligned} \tag{2.22}$$

DeepCoNN assumes that for each user, the review of a target item is always available. However in real-world contexts, a product is recommended to a user before they have experienced it. To overcome this limitation, TransNet was introduced in Catherine and Cohen [2017] with two networks: a source network based on DeepCoNN, and a target network. The target network processes the target review $text_{ij}$ which denotes the review written by the user i for the item j. The target rating $\hat{Y}_{ij}^{target}$ is predicted as follows:

$$\begin{aligned} Z_{ij}^{target} &= \text{CNN}^{target}(text_{ij}), \\ \hat{Y}_{ij}^{target} &= \text{FM}^{target}(Z_{ij}^{target}). \end{aligned} \tag{2.23}$$

The source network is similar to the DeepCoNN model with two CNN text processors, for the purpose of obtaining two user-latent and item-latent representations:

$$\begin{aligned} R_i &= \text{CNN}_R(text_i - text_{ij}), \quad \text{use reviews by user } i \text{ without } text_{ij}, \\ C_j &= \text{CNN}_C(text_j - text_{ij}), \quad \text{use reviews for item } j \text{ without } text_{ij}. \end{aligned} \tag{2.24}$$

The source rating $\hat{Y}_{ij}^{source}$ is then generated through a concatenation operation, an additional transform layer, and a factorization machine, respectively:

$$\begin{aligned} Z_{ij}^{source} &= R_i \oplus C_j, \\ Z_{ij}^{transform} &= \text{TRANSFORM}(Z_{ij}^{source}), \\ \hat{Y}_{ij}^{source} &= \text{FM}^{source}(Z_{ij}^{transform}). \end{aligned} \tag{2.25}$$

The training phase of TransNet consists of three stages. In the first step, it trains the target network on the actual reviews by minimizing the loss between the target ratings $\hat{Y}^{target}$ and the actual ratings Y. Secondly, it learns the transform layer by reducing the loss $||Z^{transform} - Z^{target}||^2$. Finally, it trains the FM predictor through reducing the difference between the actual ratings Y and the predictions $\hat{Y}^{source}$, based on the learned transformed representation $Z^{transform}$. Additionally, the transform layer enables the model to generate approximate reviews during test phase when $text_{ij}$ is unavailable, and enhances the prediction performance.

We point out that in the two previous models, latent factors are solely obtained by learning reviews, without taking into account the observed ratings during the generative phase. Therefore, we propose a novel strategy that employs both ratings and reviews as model inputs to gain more information. We further introduce a user-majoring and a product-majoring encoders that simultaneously capture the user and product preferences, as detailed in Chapter 3.

2.3 Network data analysis

A number of probabilistic and deep learning approaches have been proposed for network analysis, some of the most popular among them are described in this section.

2.3.1 Probabilistic graphical models

This section reviews two major model-based approaches for network analysis: the stochastic block model (SBM) [Wang and Wong, 1987; Nowicki and Snijders, 2001] and the latent position model (LPM) [Hoff et al., 2002].

SBM is widely used to detect communities or more general clusters of nodes. It assumes that the nodes of a network (or graph) are spread into K different latent clusters and that the connection probability between each pair of nodes depends exclusively on their group memberships. The SBM introduced by Nowicki and Snijders [2001] assigns a latent vector Z_i, following a multinomial distribution, to each vertex i of the network:

$$Z_i \sim \mathcal{M}(1, \alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_k\}), \tag{2.26}$$

where α denotes the vector of cluster proportions. In more detail, the vector Z_i has all its components equal to zero except one such that $Z_{ik} = 1$ if vertex i belongs to cluster k. Moreover,

$$\sum_{k=1}^{K} \alpha_k = 1, \qquad \text{with} \quad \alpha_k > 0, \forall k. \tag{2.27}$$

Finally, edges between two nodes are drawn from independent Bernoulli distributions, given Z. We denote by A the $N \times N$ adjacency matrix, where $A_{ij} = 1$ if there is a link between nodes i and j, 0 otherwise. Then,

$$A_{ij} | (Z_{ik} Z_{jl} = 1) \sim \mathcal{B}(\Pi_{kl}), \tag{2.28}$$

where $\mathcal{B}(\cdot)$ denotes the Bernoulli distribution and Π is a $K \times K$ matrix whose entry Π_{kl} is the connection probability between any node in cluster k and any node in cluster l.

Based on SBM, many extensions looking for overlapping clusters have been proposed [Lee and Wilkinson, 2019]. For instance, the mixed-membership stochastic blockmodel (MMSB) [Airoldi et al., 2008] introduces a mixing weight vector π_i drawn from a Dirichlet distribution for each vertex i, where π_{ik} denotes the probability of node i belonging to cluster k. The connection

probability between node i and node j is then modeled as:

$$A_{ij}|Z_{i\to j}, Z_{i\leftarrow j} \sim \mathcal{B}(Z_{i\to j}^\top \Pi Z_{i\leftarrow j}) \tag{2.29}$$

where Π still is a $K \times K$ connection probability matrix, similar to SBM, and the membership indicator $Z_{i\to j} \sim \mathcal{M}(1, \pi_i)$ describes the class membership of vertex i in relation to vertex j. Similarly, the class membership of vertex j in reference to vertex i is represented by the membership indicator for the receiver $Z_{i\leftarrow j} \sim \mathcal{M}(1, \pi_j)$. Each vertex can therefore belong to a different cluster depending on its relationships with other vertices.

In the overlapping stochastic blockmodel (OSBM) [Latouche et al., 2011], each node is also allowed to belong to multiple clusters. Indeed, OSBM assumes that each node is associated with a binary latent vector drawn from a product of Bernoulli distributions:

$$Z_i \sim \prod_{k=1}^{K} \mathcal{B}(Z_{ik}; \alpha_k) = \prod_{k=1}^{K} \alpha_k^{Z_{ik}} (1 - \alpha_k)^{1 - Z_{ik}}. \tag{2.30}$$

The edges are then generated as:

$$A_{ij}|Z_i, Z_j \sim \mathcal{B}(\sigma(Z_i^\top \Pi Z_j + Z_i^\top U + V^\top Z_j + b)), \tag{2.31}$$

where $\sigma(\cdot)$ is a logistic sigmoid function, Π is a $K \times K$ matrix, U and V are K-dimensional vectors. The first term $Z_i^\top \Pi Z_j$ accounts for the interaction probability between two nodes, whereas the second term $Z_i^\top U$ models the overall probability that node i connects to other nodes and $V^\top Z_j$ captures the global tendency of node j to receive an edge connection. Lastly, b is a bias term.

All models discussed so far deal with binary-edged graphs. Other variants consider weighted graphs whose edges can be discrete [Mariadassou et al., 2010], categorical [Jernite et al., 2014] or textual [Bouveyron et al., 2018]. Moreover, some extensions [Xu and Hero, 2014; Matias and Miele, 2017; Corneli et al., 2019] focus on time-evolving networks, namely dynamic network analysis.

Whereas stochastic block models aim at clustering the nodes of a graph, other approaches look for likely positions of the nodes of the graph in a latent space, to better visualize the network. Originally proposed by Hoff et al. [2002], the latent position model (LPM) supposes that each node has an unknown position in a P-dimensional latent space and that the probability of a specific link between two nodes is modeled by some function of their positions. The following describes

the generative process. Firstly, for each node i, a latent position is sampled from a Gaussian distribution:

$$Z_i \sim \mathcal{N}(0, I_P). \tag{2.32}$$

Next, based on the distance between the latent positions of two nodes, a connection is sampled from a Bernoulli distribution independently, for each pair:

$$A_{ij}|Z_i, Z_j \sim \mathcal{B}(\sigma(\alpha + \beta Y_{ij} - |Z_i - Z_j|)), \tag{2.33}$$

where $\sigma(\cdot)$ is a logistic sigmoid function, Y_{ij} is a covariate accounting for additional information about the edge connecting i with j, if available, and α as well as β are two model parameters.

Afterwards, the latent position cluster model (LPCM) [Handcock et al., 2007] was introduced to incorporate a clustering structure into LPM by considering that the latent position of each node is drawn from a Gaussian mixture model (GMM):

$$Z_i \sim \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \sigma_k^2 I_P), \tag{2.34}$$

where π_k is the probability that a node belongs to the group k. Then, similarly to the original LPM, LPCM links the probability of an edge between nodes to their latent positions.

As a side note, we recall that in the last decades, several efforts have been made in order to extend LPMs to dynamic networks [Xu and Zheng, 2009; Sewell and Chen, 2017]. Further developments of LPMs exist and the reader is referred to Raftery [2017] for an extensive review.

The generative models described so far, often require a challenging inference procedure that primarily relies on variational approximations MCMC, which unavoidably results in high computational complexity. Scalability and adaptation to complex networks are major concerns for the researchers in the field. A more general overview of statistical models for clustering network data can be explored in Bouveyron et al. [2019, Chapter 10].

2.3.2 Graph neural networks

The methods described so far are widely used for clustering the nodes of a graph. Another challenging task consists in clustering the graphs themselves. Ad hoc deep learning techniques have been developed in recent years. Among various classes of deep learning techniques, graph neural networks (GNNs) are a type of neural networks that operates directly on graph structured

data, which have demonstrated effectiveness in representation learning on graphs. Since the seminal work of Scarselli et al. [2008], GNNs adopt a neighborhood aggregation (or message passing) scheme in which the representation vector of a node is computed by recursively aggregating and transforming the feature vectors of its neighbors. Mathematically, given the node feature matrix X, the l-th layer of a multi-layer GNN operates as follows:

$$\begin{aligned} N_i^{(l+1)} &= \text{AGGREGATE}^{(l+1)}\left(\left\{H_j^{(l)} : j \in \mathbf{N}(i)\right\}\right), \\ H_i^{(l+1)} &= \text{COMBINE}^{(l+1)}\left(H_i^{(l)}, N_i^{(l+1)}\right), \end{aligned} \tag{2.35}$$

where $H_i^{(l)}$ is the feature vector of node i at the l-th iteration with an initialization of $H_i^{(0)} = X$. The neighborhood of nodes adjacent to i is denoted by the symbol $\mathbf{N}(i)$. Different GNN versions emerge from various choices for the function AGGREGATE$(\cdot)$ and COMBINE$(\cdot)$ [Xu et al., 2019]. For node clustering, the node representation $H_i^{(l+1)}$ of the final iteration is used with an external clustering algorithm like K-means. For graph clustering, node features from the final iteration are gathered to obtain the entire graph representation:

$$H_G = \text{READOUT}(H_i^{(l+1)}, i \in G), \tag{2.36}$$

where READOUT can be a simple summation or a more complicated pooling function [Ying et al., 2018b]. In this thesis, we only concentrate on node-level clustering approaches, however by including a READOUT function, tasks at the graph level can be envisaged.

GCN. The graph convolutional networks (GCN) [Kipf and Welling, 2017] is a multi-layer convolutional neural network that works directly on graph data. GCN fuses the AGGREGATE and COMBINE phases in Eq. (5.15) to learn latent representation for each node. Thus, the $(l+1)$-th layer of a multi-layer GCN looks like:

$$H^{(l+1)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}\hat{W}^{(l)}), \tag{2.37}$$

where the i-th row of $H^{(l+1)}$ is the latest representation of node i at layer $l+1$, $\hat{A} = A + I_N$ is the adjacency matrix of the undirected graph G with an added identity matrix I_N representing self-connections, $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$ is the degree matrix and $\hat{W}^{(l)}$ is a weight matrix of learnable parameters of l-th layer. $\sigma(\cdot)$ denotes an activation function such as ReLU or Sigmoid. $H^{(l)}$ signifies the integrated representation in layer l, with $H^{(0)} = X$ is the node features matrix.

GraphSAGE. Most existing architectures, including GCN, are fundamentally "transductive": they require that every node of the graph is present during the training phase. GraphSAGE [Hamilton et al., 2017] introduces a general inductive framework to efficiently generate node embeddings for previously unseen nodes with the use of node features information. This technique operates by sampling a fixed-size neighborhood of each node, uniform drawn from the full neighborhood set, and then applying a specific aggregator to:

$$\begin{aligned} N_i^{(l+1)} &= \text{AGGREGATE}^{(l+1)}(\{H_j^{(l)} : j \in \mathbf{N}(i)\}), \\ H_i^{(l+1)} &= \sigma(\hat{W}^{(l+1)} \cdot (H_i^{(l)} \oplus N_i^{(l+1)})), \end{aligned} \tag{2.38}$$

where GraphSAGE concatenates the node current representation $H_i^{(l)}$ with the aggregated neighborhood vector $N_i^{(l+1)}$, and this concatenated vector is fed through a fully connected layer with non-linear activation function $\sigma(\cdot)$. A number of aggregator structures can be used to aggregate the neighbor representations. GraphSAGE examined three aggregation functions: mean aggregator, LSTM aggregator, and max-pooling aggregator, respectively.

GAT. All of the aforementioned approaches integrate the nodes representation equally with all of its neighbors. However, in order to measure the importance of distinct neighbors, the graph attention networks (GATs) [Veličković et al., 2018] assign different weights to neighbors of a node. For a *single* graph attentional layer, the attention coefficient α_{ij} that indicates the importance of the neighbor node j to node i is calculated as:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(a[\hat{W}h_i \oplus \hat{W}h_j]))}{\sum_{r \in \mathbf{N}(i)} \exp(\text{LeakyReLU}(a[\hat{W}h_i \oplus \hat{W}h_r]))}, \tag{2.39}$$

where h_i and h_j are attribute vectors associated with node i and j, a is the attention mechanism modeled by a single-layer neural network parametrized by a weight vector $\vec{a}$, $\hat{W}$ is a weight matrix of learnable parameters and LeakyReLU is a non-linear activation function ($:= \mathbb{1}_{\{x<0\}}(bx) + \mathbb{1}_{\{x\geq 0\}}(x)$, b is a small constant). Once obtained, the attention coefficients and the weight matrix are used to compute the final output features for every node:

$$\hat{h}_i = \sigma\left(\sum_{j \in \mathbf{N}(i)} \alpha_{ij} \hat{W} h_j\right), \tag{2.40}$$

where $\sigma(\cdot)$ denotes a non-linearity function. To stabilize the learning process, a *multi-head* attention mechanism is introduced. Specifically, the final output after applying L independent attention

mechanisms is:

$$\hat{h}_i = \sigma(\frac{1}{L}\sum_{l=1}^{L}\sum_{j\in\mathbf{N}(i)}\alpha_{ij}^{(l)}\hat{W}^{(l)}h_j). \tag{2.41}$$

GIN. The graph isomorphism network (GIN) [Xu et al., 2019] generalizes the Weisfeiler-Lehman (WL) test to address the challenging graph isomorphism problem [Weisfeiler and Leman, 1968] and thereby achieves maximum discriminative power among GNNs. In particular, GIN updates nodes representations as:

$$H_i^{(l+1)} = \text{MLP}((1+\epsilon^{(l)})H_i^{(l+1)} + \sum_{j\in\mathcal{N}(i)} H_j^{(l)}), \tag{2.42}$$

where ϵ is a learnable parameter or a fixed scalar, MLP stands for multi-layer perceptron.

MGAE. The marginalized graph auto-encoder (MGAE) [Wang et al., 2017] is developed based on GCN. MGAE proposes a marginalization mechanism by adding random noises to the content information. The model first arbitrarily removes some features (setting them to 0) from node features matrix X to get a corrupted version of the original node contents, referred to as $\tilde{X}$. Next, the adjacency matrix $\hat{A}$ and the corrupted node content $\tilde{X}$ are used as inputs to train a GCN. Then, a latent representation $Z^{(1)}$ is obtained by minimizing the error between the output of GCN and the initial X:

$$||X - \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}\tilde{X}\hat{W}^{(0)}||^2 + \lambda||\hat{W}^{(0)}||^2, \tag{2.43}$$

where $\hat{W}^{(0)}$ is a learnable weight matrix at the first layer, $||\hat{W}^{(0)}||^2$ is a regularization term with a trade-off coefficient λ, and $Z^{(1)} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}\tilde{X}\hat{W}^{(0)}$. By stacking multiple single-layer GCN encoders, the final representation is obtained as $Z^{(l+1)} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}Z^{(l)}\hat{W}^{(l)}$. Lastly, a spectral clustering is applied to the final representation of nodes to produce clusters.

2.3.3 Deep probabilistic models for node clustering

In this subsection, we focus on deep latent variable models that generalize the VAE structure to deal with graph data. Since these models are able to learn meaningful representations of the nodes of a graph in a latent space, they are often used for node clustering purposes. Also they can produce new data samples based on the latent features that they have learnt in the embedded space. We stress that variational auto-encoding is an entirely unsupervised learning approach that can be used in contexts like clustering where label information is unavailable.

The first approach that we consider is the variational graph auto-encoder (VGAE) [Kipf and Welling, 2016]. In VGAE, a GCN encoder is introduced in order to compress the network data into a low-dimensional representation in a latent space. A simple inner product decoder is then used to transfer the latent component into a rebuilt graph matrix. For more detail, VGAE first assumes that:

$$p(Z) = \prod_{i=1}^{N} p(z_i) = \prod_{i=1}^{N} \mathcal{N}(z_i|0, I_P), \tag{2.44}$$

where Z is a P-dimensional latent matrix where the i-th row Z_i represents the i-th node. A random edge between two nodes is sampled through:

$$p(A_{ij}|z_i, z_j) = \sigma(z_i^\top z_j), \tag{2.45}$$

where A is the graph adjacency matrix and $\sigma(\cdot)$ is a logistic sigmoid function. The term $\sigma(z_i^\top z_j)$ can be viewed as the *decoder* network in VGAE. The inference model is parametrized by a two-layer GCN:

$$q(Z|X, A) = \prod_{i=1}^{N} q(z_i|X, A) = \prod_{i=1}^{N} \mathcal{N}(z_i|\mu_i, \text{diag}(\sigma_i^2)). \tag{2.46}$$

Here, X is the node feature matrix, $\mu = \text{GCN}_\mu(X, A)$ is the matrix of mean vectors μ_i, similarly $\log\sigma = \text{GCN}_\sigma(X, A)$. The two-layer GCN *encoder* is defined as $\text{GCN}(X, A) = \tilde{A}\text{ReLU}(\tilde{A}X\hat{W}_0)\hat{W}_1$, where $\tilde{A} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix. As in VAE, the optimization is then performed by maximizing the variational lower bound $\mathcal{L}$:

$$\mathcal{L} = \mathbb{E}_{q(Z|X,A)}[\log p(A|Z)] - \mathcal{KL}(q(Z|X, A)||p(Z)). \tag{2.47}$$

Whereas variational auto-encoding based methods learn meaningful latent positions of the nodes relying on the minimization of the KL divergence between the approximate variational and the true posterior distribution of those positions, an alternative approach consists in using generative adversarial networks (GAN) [Goodfellow et al., 2020]. By incorporating an adversarial model into the generative process, the adversarially regularized graph auto-encoder (ARVGA) [Pan et al., 2018] enforces the latent representation to match a prior distribution. The adversarial model serves as a discriminator $\mathcal{D}$ to determine whether a latent factor is from the prior $p(Z) = \mathcal{N}(Z|0, I)$ (positive) or from the output of graph encoder $\text{GCN}(X, A)$ (negative). The embedding will finally be regularized and enhanced during the training phase by minimizing the cross-entropy cost for the binary classifier:

$$-\frac{1}{2}\mathbb{E}_{z\sim p(z)}[\log \mathcal{D}(Z)] - \frac{1}{2}\mathbb{E}_{x\sim p(x)}[\log(1 - \mathcal{D}(\text{GCN}(X, A)))]. \tag{2.48}$$

After obtaining the optimal latent positions of the nodes, ARVGA performs node clustering on the learned embeddings using the K-means algorithm.

In order to allow nodes to belong to multiple clusters, the deep generative latent feature relational model (DGLFRM) [Mehta et al., 2019] combines OSBM [Latouche et al., 2011] with GCN by positing each node of the graph to have an embedding modeled by a sparse vector:

$$z_i = b_i \odot r_i, \tag{2.49}$$

where $\odot$ denotes an element-wise multiplication, $b_i \in \{0,1\}^K$ is a binary vector with a stick-breaking process prior with a parameter α:

$$\begin{aligned} v_k &\sim \text{Beta}(\alpha, 1), \qquad k = 1, \cdots, K, \\ \tau_k &= \prod_{j=1}^{k} v_j, \qquad b_{ik} \sim \mathcal{B}(\tau_k), \end{aligned} \tag{2.50}$$

and $r_i \in \mathbb{R}^K$ is a real-valued vector with a Gaussian prior:

$$r_i \sim \mathcal{N}(0, \sigma^2 I), \tag{2.51}$$

In particular, $b_{ik} \in \{0,1\}$ and $r_{ik} \in \mathbb{R}$ indicate whether or not node i belongs to cluster k and the membership strength to the same cluster k, respectively. The VAE decoder then computes the link probabilities as:

$$p(A_{ij}|z_i, z_j) = \sigma(f(z_i)^\top f(z_j)), \tag{2.52}$$

where $f(\cdot)$ is a deep neural network equipped with a LeakyReLU activation in each hidden layer. Considering the inference model, the variational distributions are:

$$\begin{aligned} q_\phi(v_{ik}) &= \text{Beta}(c_{ik}, d_{ik}), \\ q_\phi(b_{ik}) &= \mathcal{B}(\pi_{ik}), \\ q_\phi(r_i) &= \mathcal{N}(\mu_i, \text{diag}(\sigma_i^2)), \end{aligned} \tag{2.53}$$

where $\{c_k, d_k, \pi_k, \mu_k, \sigma_k\}_{i=1}^N = \text{GCN}(X, A)$. These parameters are all the outputs of a GCN.

All of the aforementioned and other existing approaches [Tian et al., 2014; Nie et al., 2017; Zhang et al., 2019b] adopt a two-step clustering procedure, simply relying on *external* clustering algorithms (e.g. K-means) to group the embedded nodes, independently from the generative model. To tackle this problem, a self-training clustering strategy that enables the simultaneous

optimization of graph embeddings and clusters was developed in the deep attentional embedded graph clustering model (DAEGC) [Wang et al., 2019a]. DAEGC builds a graph attentional encoder based on GAT and utilizes an inner product decoder. Apart from calculating the reconstruction error L_r between the original and rebuilt graph adjacency matrices, it also suggested a self-optimizing clustering module that minimizes the following objective:

$$L_c = \sum_i \sum_k p_{ik} \log \frac{p_{ik}}{q_{ik}}, \tag{2.54}$$

where the target distribution p_{ik} is defined as:

$$p_{ik} = \frac{q_{ik}^2 / \sum_i q_{ik}}{\sum_k (q_{ik}^2 / \sum_i q_{ik})}, \tag{2.55}$$

and q_{ik} measures the similarity between node embedding z_i and the cluster center embedding μ_k:

$$q_{ik} = \frac{(1 + ||z_i - \mu_k||^2)^{-1}}{\sum_r (1 + ||z_i - \mu_r||^2)^{-1}}. \tag{2.56}$$

It can be viewed as a soft clustering assignment distribution of each node. One may obtain the estimated label for each node i using $\underset{k}{\operatorname{argmax}} q_{ik}$. The initial cluster centers μ are first initialized by applying K-means to the embedding matrix Z. Then they are optimized together with the latent embeddings through SGD. Finally, a total objective function is designed as follows to jointly optimize the graph embeddings and perform clustering of the nodes:

$$L = L_r + \lambda L_c, \tag{2.57}$$

with a coefficient λ that regulates the balance in between.

Despite the good achievements, all of the methods discussed above employ an inner-product-based decoder, whereas we argue that a different solution, accounting for the Euclidean distance between each pair of nodes in the latent space might be more suited. More details are provided in Chapter 4. As a side note, we also point out variants that made use of alternative decoder structures in Park et al. [2019]; Li et al. [2020].

2.4 Clustering in heterogeneous information networks

Heterogeneous mixed-type data is a common component in real-world networks. In a scientific article citation network, for example, textual information such as paper titles and abstracts

is included in addition to paper-paper interactions. How to incorporate this valuable information under the graph structure is crucial since it affects the quality of the latent embeddings. Most GNNs for heterogeneous information networks typically encode the two informations, the graph adjacency matrix A and the node attributes X, in a simple way. For instance, in GCN, the convolution operation is like $\tilde{A}X\hat{W}$ (recall that $\tilde{A} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$, $\hat{W}$ is a learnable weight matrix), which results in the loss of information and limits the ability to represent graph topology. This fact is demonstrated in Chapter 5 in detail.

AM-GCN. To overcome this problem, an adaptive multi-channel graph convolutional networks (AM-GCN) was proposed by Wang et al. [2020b]. The authors showed that the fusion capability of GCNs on network topological structures and node attributes is inadequate. The main concept of AM-GCN is that node embedding is simultaneously learned based on node features, topological structures, and their combinations. First, in order to capture the underlying structure of nodes in a feature space, a k-nearest neighbor (kNN) graph $G_f = (A_f, X)$ is built based on node feature matrix X, where A_f is the adjacency matrix of G_f. The l-th layer output can then be expressed as follows using the input graph (A_f, X):

$$H_f^{(l+1)} = \sigma(\hat{D}_f^{-\frac{1}{2}}\hat{A}_f\hat{D}_f^{-\frac{1}{2}}H_f^{(l)}\hat{W}_f^{(l)}), \tag{2.58}$$

where $\hat{A}_f = A_f + I_f$, $\hat{D}_f$ is the diagonal degree matrix of $\hat{A}_f$, $\sigma(\cdot)$ is the ReLU activation function, the initial $H_f^{(0)} = X$ and $\hat{W}_f^{(l)}$ is the learnable weight matrix of the l-th layer in GCN. In this way, the model learns the node embedding Z_f that captures the specific information in feature space. As for the graph topology, the model takes the original input graph $G_t = (A_t, X)$ (t stands for "topology"), and a GCN produces the latent embedding Z_t as the output.

Additionally, to extract the common information shared by the two spaces, a common-GCN adopts a parameter sharing strategy. First, the node embedding Z_{ct} from topology graph G_t is obtained by:

$$H_{ct}^{(l+1)} = \sigma(\hat{D}_t^{-\frac{1}{2}}\hat{A}_t\hat{D}_t^{-\frac{1}{2}}H_{ct}^{(l)}\hat{W}_c^{(l)}), \tag{2.59}$$

where $H_{ct}^{(0)} = X$ and $\hat{W}_c^{(l)}$ is the l-th layer learnable weight matrix of common-GCN. Then, by using the same weight matrix $\hat{W}_c^{(l)}$, the node embedding Z_{cf} from feature graph G_f is produced:

$$H_{cf}^{(l+1)} = \sigma(D_f^{-\frac{1}{2}}\hat{A}_f D_f^{-\frac{1}{2}}H_{cf}^{(l)}\hat{W}_c^{(l)}). \tag{2.60}$$

Next, the common embedding Z_c of the two spaces is determined as the barycenter of the two output embeddings from common-GCN :

$$Z_c = (Z_{ct} + Z_{cf})/2. \tag{2.61}$$

A final embedding Z is obtained by :

$$Z = \alpha_t Z_t + \alpha_c Z_c + \alpha_f Z_f, \tag{2.62}$$

where α_t, α_c, $\alpha_f \in \mathbb{R}^{N\times 1}$ indicate the attention values of N nodes with embeddings Z_t, Z_c, Z_f, respectively.

BiTe-GCN. An alternative approach based on the GCN architecture was introduced to investigate the word semantic structures in BiTe-GCN [Jin et al., 2021]. This model initially converts the original text-rich network into a bi-typed network with two sorts of nodes, namely the real nodes (e.g., document nodes in the original network) and the entity nodes (e.g., phrases or words extracted form documents). Afterwards, it generates three different kinds of edges : edges between real nodes, such as paper citations, edges between real nodes and entity nodes that reflect their inclusion relationships, and edges between entity nodes which indicate the semantic structure information in the documents. In particular, BiTe-GCN introduces a document network $G_D = (V_D, E_D)$ and another word network $G_W = (V_W, E_W)$. Each node in G_D is associated with a document d_i that describes it, and in G_W. The nodes V_W is a collection of all representative words or phrases extracted form the corpus. E_W is the edges connecting word pairs (w_i, w_j). Based on these two networks, a bi-typed network is constructed as :

$$G_{DW} = (V_D \cup V_W, E_{DW}), \tag{2.63}$$

with

$$E_{DW} = \{(d, w)|w \in d, \forall w \in V_W, \forall d \in V_D\}. \tag{2.64}$$

Three networks introduced above are then combined to form the entire network G :

$$G = G_D \cup G_{DW} \cup G_W = (V_D \cup V_W, E_D \cup E_{DW} \cup E_W). \tag{2.65}$$

BiTe-GCN finally performs convolution to enable message passing within the same kind of subnetwork :

$$H_t^{(l+1)} = \text{GCN}_t(H^{(l)}), \qquad \forall t \in \{D, W, DW\}, \tag{2.66}$$

here three different GCNs are adopted, following by an aggregation for each sort of sub-networks:

$$H^{(l+1)} = \text{AGGREGATE}(H_t^{(l+1)}), \qquad \forall t \in \{D, W, DW\}. \tag{2.67}$$

AS-GCN. As an extension of BiTe-GCN, an adaptive semantic architecture of GCNs, namely AS-GCN [Yu et al., 2021] was developed. It combines a GCN module with a neural topic model (NTM) that extracts word and topic semantics from raw text. Based on the VAE framework, NTM learns the latent topics through an encoding-decoding procedure. Specifically, a latent topic embedding Z is defined as:

$$Z \sim \mathcal{N}(\mu, \sigma^2 I_T), \tag{2.68}$$

where μ and σ^2 are obtained by a neural network *encoder* f_e. The topic proportions are then obtained as:

$$\theta = \text{softmax}(Z). \tag{2.69}$$

The predicted words W_d in the document d are generated by using the topic distribution θ as follows:

$$W_d \sim \text{softmax}(\beta^\top \theta_d), \tag{2.70}$$

where each entry β_{tv} represents the probability that the vocable v occurs in the t-th topic.

Subsequently, using words retrieved from raw text and distributions obtained from NTM, AS-GCN transforms the initial text-rich network into a tri-typed heterogeneous network. It includes three types of nodes: real nodes (e.g., document nodes in the original network), topic nodes (e.g., topics obtained from NTM) and entity nodes (e.g. words extracted from raw text) and four types of edges: edges between real nodes, edges between real nodes and topic nodes reflecting the topic distribution in documents, edges between topic nodes and entity nodes representing the word distribution in topics, as well as edges between entity nodes indicating the local word semantics. Finally, the augmented tri-typed network can be defined as:

$$G = (V_D \cup V_T \cup V_W, E_D \cup E_{DT} \cup E_{TW} \cup E_W). \tag{2.71}$$

The convolution and aggregation processes of AS-GCN are similar to those in BiTe-GCN. The main difference is that AS-GCN requires the calculation of an additional loss term for the NTM part.

Inspired by the combination of topic modeling and graph embeddings, as well as the capability to preserve graph topology with a latent position-based decoder developed in Chapter 4, we further proposed a new method to account for all these components. In our approach, a specialized document similarity-based graph convolutional network is used to encode both the graph topology structure and semantics in textual data. Two decoders are then employed to reconstruct both the graph adjacency matrix and the document-term matrix. More details on graph learning ability and node clustering performance are stated in Chapter 5.

CHAPTER 3

DeepLTRS: a deep latent recommender system based on user ratings and reviews

We introduce a deep latent recommender system named deepLTRS in order to provide users with high quality recommendations based on observed user ratings and texts of product reviews. Our approach adopts a variational auto-encoder (VAE) architecture as a deep generative latent model for an ordinal matrix encoding ratings and a document-term matrix encoding the reviews. Moreover, a user-majoring encoder and a product-majoring encoder are constructed to jointly capture user and product preferences. Due to the specificity of the model structure, an original row-column alternated mini-batch optimization algorithm is proposed to deal with user-product dependencies and computational burden. Numerical experiments on simulated and real-world data sets demonstrate that deepLTRS outperforms the state-of-the-art, in particular in contexts of extreme data sparsity.

This Chapter is related with one accepted communication in an international workshop, ***Missing rating imputation based on product reviews via deep latent variable models****, ICML Workshop on the Art of Learning with Missing Values (Artemiss), (2020), and one accepted journal version,* ***DeepLTRS****:* ***A deep latent recommender system based on user ratings and reviews****, Pattern Recognition Letters, vol. 152, pp. 267-274 (2021).*

3.1 Introduction 47
3.1.1 Organization of the chapter 48
3.2 A rating-and-review based recommender system 48
3.2.1 Framework and notations 48
3.2.2 Generative model of deepLTRS 49
3.3 Variational auto-encoding inference 52
3.3.1 Variational lower bound (ELBO) 52
3.3.2 Monte Carlo EM algorithm and mini-batching 54
3.4 Numerical experiments on simulated data 57
3.4.1 Architecture and simulation setup 57
3.4.2 DeepLTRS with and without text data 58
3.4.3 Benchmark and effect of data sparsity 61
3.5 Application on real-world data 62
3.6 Conclusion and perspectives 68

3.1 Introduction

As discussed in Chapter 1.1, in the current era of information explosion, recommendation systems have become central tools in a wide range of applications ranging from e-commerce [Huang et al., 2007] to the global positioning of IoT devices [Gao et al., 2019]. Examples of recommended objects include movies, songs, books, hotels, as well as restaurants to name just a few. At the core of the research on recommendation systems, we point out a widely adopted collaborative filtering (CF) approach [Su and Khoshgoftaar, 2009], which relies on similarities among user historical preferences on a set of items. Generally, by converting the list of users and products into a user-item rating matrix, a CF-based recommender system can be considered as completing the rating matrix based on observed entries [Ramlatchan et al., 2018].

While most users only ranked few products, the rating matrix is usually large and extremely sparse due to massive amounts of missing values. Considering that many consumers also use texts to express various opinions along with the scores, reviews can contain crucial information from different aspects about the products, compared to a single rating. Therefore, efforts have been recently put in developing algorithms capable of dealing not only with ratings but also with other sources of information like text reviews to address the matrix completion problem in the case of high data sparseness [Chen et al., 2015]. A long series of techniques have been proposed in the literature for recommendations and we discussed some of them in Section 2.2.

In order to both improve the robustness to data sparsity and the interpretability of recommendations, we introduce here a deep latent recommender system (deepLTRS), which takes into account both observed ratings and the textual information collected in product reviews as the model input. DeepLTRS extends the Probabilistic matrix factorization (PMF) by relying on recent auto-encoding extensions. Considering the review part, we use the ProdLDA model introduced in Section 2.1.2 to obtain higher quality topics with respect to a standard LDA. Our goal here is to show that, regardless the parsimony of our model, we are still able to improve performance compared to the state-of-the-art techniques. Our approach has the following key-features:

— a variational auto-encoder (VAE) architecture is used as a generative latent model for both an ordinal matrix encoding ratings and a document-term matrix encoding reviews;

— since the connection between latent factors and the review data is difficult to formulate, we adopt a neural network to capture the relationship between latent representations and latent topics;
— one user-majoring encoder and another product-majoring encoder are constructed to jointly capture user and product preferences. Then, two different decoders are designed for ratings and reviews separately;
— due to the specific model structure, an original strategy of alternating rows and columns in mini-batch optimization is proposed to deal with user-product dependencies and to reduce the computational costs. We further provide a theoretical proof of the unbiasedness of our empirical loss estimator.

3.1.1 Organization of the chapter

This chapter is organized as follows. In Section 3.2, the generative model of deepLTRS for both ratings and reviews is described. Section 3.3 details the auto-encoding variational inference procedure along with an original row-column alternated mini-batch strategy allowing us to reduce the computational burden. Our approach is then applied in Section 3.4 on simulated data sets to highlight its main features. DeepLTRS is also compared in this section with other state-of-the-art approaches in the contexts of extreme data sparsity. Section 3.5 presents a benchmark of deepLTRS and most efficient alternatives on real-world data sets from e-commerce systems. Finally, some conclusive remarks and possible further works are proposed in Section 3.6.

3.2 A rating-and-review based recommender system

In this section, we describe the deepLTRS accounting for both ratings and reviews for recommendations.

3.2.1 Framework and notations

We consider data sets involving M users who are scoring and reviewing N products. Such data sets can be encoded by two matrices: an ordinal data matrix Y accounting for the *scores* that users assign to products and a document-term matrix (DTM) W encoding the *reviews* that users write about products.

Ordinal data. The ordinal data matrix Y in $\mathbb{N}^{M\times N}$ is such that Y_{ij} corresponds to the score that the i-th user assigns to the j-th product. This matrix is usually extremely sparse in practice (most of its entries are missing) corresponding to users *not* scoring nor reviewing some products. Conversely, when a score is assigned, it takes values in $\{1,\dots,H\}$ with $H > 1$. Henceforth, we assume that an ordinal scale is consistently defined. For instance, when customers evaluate products, 1 always means "very poor" and H is always associated with "excellent" reviews. The number of ordered levels H is assumed to be the same for all (not missing) Y_{ij}. If it is not the case, a scale conversion pre-processing algorithm (see e.g. [Gilula et al., 2019]) can be employed to normalize the number of levels.

Text data. By considering all the available reviews, it is possible to store all the different vocables employed by the users into a *dictionary* of size V. Thenceforth, we denote by $W^{(i,j)}$ a row vector of size V encoding the review by the i-th user to the j-th product. The v-th entry of $W^{(i,j)}$, denoted by $W_v^{(i,j)}$, is the number of times (possibly zero) that the word v of the dictionary appears into the corresponding review. The document-term matrix W is obtained by concatenation of all the row vectors $W^{(i,j)}$. For the sake of clarity, we assume that the review $W^{(i,j)}$ exists if and only if Y_{ij} is observed. Note that, since each row in W corresponds to one (and only one) not missing entry in Y, the number of rows in the DTM is the same as the number of observed non-missing values in Y.

3.2.2 Generative model of deepLTRS

It is now assumed that each user i and product j have latent representations R_i and C_j, respectively, in a low-dimensional space $\mathbb{R}^P$, with $P \ll \min\{M,N\}$.

Ratings. The following generative model is adopted for the ratings

$$Y_{ij} = \langle R_i, C_j\rangle + b_i^u + b_j^p + \epsilon_{ij}, \quad \forall i = 1,\dots,M, \forall j = 1,\dots,N, \tag{3.1}$$

where $\langle\cdot,\cdot\rangle$ is the standard scalar product and b_i^u, b_j^p are two unknown real parameters accounting for biases specific to users and products respectively. Finally, the residuals ϵ_{ij} are assumed to be i.i.d. normally distributed random variables: $\epsilon_{ij} \sim \mathcal{N}(0,\eta^2)$.

In the following, R_i and C_j are seen as random vectors with zero mean and the identity matrix of dimension P as the variance, such that

$$\begin{aligned} R_i &\overset{\text{i.i.d}}{\sim} \mathcal{N}(0, I_P), \qquad \forall i, \\ C_j &\overset{\text{i.i.d}}{\sim} \mathcal{N}(0, I_P), \qquad \forall j, \end{aligned} \tag{3.2}$$

with R_i and C_j assumed independent. The unbiased version of this model (i.e. with $b_i^u = b_j^p = 0$) is the well known PMF. Note that, due to rotational invariance of PMF, the choice of isotropic prior distributions for R_i and C_j is in no way restrictive.

Proof.

From Eqs.(3.1)-(3.2), it easily follows that

$$Y_{ij} \sim \mathcal{N}(b_i^u + b_j^p, P + \eta^2). \tag{3.3}$$

Now, instead of assuming isotropic prior distributions for R_i and C_j, we set

$$\begin{aligned} R_i &\overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \Sigma_R), \qquad \forall i \\ C_j &\overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \Sigma_C), \qquad \forall j \end{aligned}$$

with Σ_R and Σ_C being symmetric positive definite matrices. Then

$$\Sigma_R = Q_R \Lambda_R Q_R^\top,$$

with Λ_R being the diagonal matrix with the eigenvalues of Σ_R on the main diagonal and Q_R the orthogonal matrix of the corresponding eigenvectors. Similarly

$$\Sigma_C = Q_C \Lambda_C Q_C^\top.$$

Then, the two random vectors $\overline{R}_i := \Lambda_R^{-\frac{1}{2}} Q_R^\top R_i$ and $\overline{C}_j := \Lambda_C^{-\frac{1}{2}} Q_C^\top C_j$ are independent and follow an isotropic Gaussian distribution each. Thus, if we set

$$Y_{ij} = \langle \overline{R}_i, \overline{C}_j \rangle + b_i^u + b_j^p + \epsilon_{ij},$$

we recover the marginal distribution in Eq. (3.3).

□

Reviews. We now extend the generative model outlined in the previous section to account for the document-term matrix W. Following the LDA model, each document $W^{(i,j)}$ is drawn from a mixture distribution over a set of T latent topics. In deepLTRS, topic proportions in the document $W^{(i,j)}$ are denoted by θ_{ij}, a vector lying in the $T-1$ simplex, i.e. $\theta_{ij} \in [0,1]^T$, such that $\sum_{t=1}^{T} \theta_{ij} = 1$. Moreover, we assume that

$$\theta_{ij} = \sigma(f_\gamma(R_i, C_j)), \qquad \forall i, j, \tag{3.4}$$

where $f_\gamma : \mathbb{R}^{2P} \to \mathbb{R}^T$ is a continuous function approximated by a neural network parametrized by γ to capture the relationship between latent factors and latent topics, $\sigma : \mathbb{R}^T \to \mathbb{R}^T$ denotes the softmax function defined by $(\sigma(z))_t = \frac{\exp(z_t)}{\sum_{l=1}^{T} \exp(z_l)}, t \in 1, \dots, T$, where $(\sigma(z))_t$ is the t-th entry of vector $\sigma(z) \in [0,1]^T$ and z denotes here a generic vector in $\mathbb{R}^T$.

As in LDA, each document $W^{(i,j)}$ is seen as a vector in $\mathbb{N}^V$ (we recall that V is the dictionary size) obtained as

$$W^{(i,j)} | \theta_{ij} \sim \text{Multinomial}(L_{ij}, \beta\theta_{ij}), \qquad \forall i, j, \tag{3.5}$$

where L_{ij} is the number of words in the review $W^{(i,j)}$ and $\beta \in [0,1]^{V \times T}$ is a matrix whose entry β_{vt} is the probability that vocable v occurs in topic t. By construction, $\sum_{v=1}^{V} \beta_{vt} = 1, \forall t$. In addition, conditionally to vectors θ_{ij}, all the reviews $\{W^{(i,j)}\}_{i,j}$ are independent random vectors.

A graphical representation of the generative model described so far can be seen in Figure 5.1.

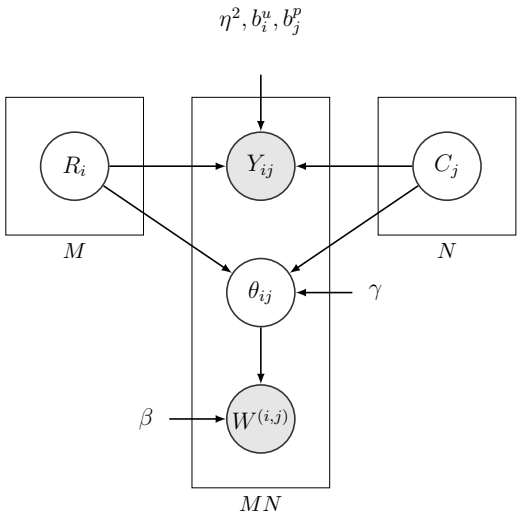


Figure 3.1 – Graphical representation of the generative model for deepLTRS (variational parameters are not included).

3.3 Variational auto-encoding inference

This section now details the auto-encoding variational inference procedure and proposes an original row-column alternate mini-batch strategy to reduce the computational burden.

3.3.1 Variational lower bound (ELBO)

Let us denote by $\Theta = \{\eta^2, \gamma, \beta, b^u, b^p\}$ the set of the model parameters introduced so far. A natural inference procedure associated with the proposed generative model would consist in looking for $\hat{\Theta}$ maximizing the (integrated) log-likelihood of the observed data (Y, W). Unfortunately, this quantity is not directly tractable and we rely on a variational lower bound to approximate it. Let us consider a joint distribution $q(\cdot)$ over the pair (R, C) of all $(R_i)_i$ and $(C_j)_j$. Thanks to the Jensen inequality, it holds that

$$\begin{aligned}
\log p(Y, W|\Theta) \geq & \mathbb{E}_{q(R,C)}\left[\log \frac{p(Y, W, R, C|\Theta)}{q(R, C)}\right] \\
= & \mathbb{E}_{q(R,C)}\left[\log p(W, Y|R, C, \Theta) + \log \frac{p(R, C)}{q(R, C)}\right] \\
= & \mathbb{E}_{q(R,C)}\left[\log p(W|R, C, \beta)\right] \\
& + \mathbb{E}_{q(R,C)}\left[\log p(Y|R, C, \gamma, \eta^2, b_u, b_p)\right] \\
& - \mathcal{KL}(q(R, C)||p(R, C)),
\end{aligned} \tag{3.6}$$

where $\mathcal{KL}$ denotes the Kullback-Leibler divergence between the variational posterior distribution of the latent row vectors $(R_i)_i, (C_j)_j$ and their prior distributions. The above inequality holds for every joint distribution $q(\cdot)$ over the pair (R, C). In order to deal with a tractable family of distributions, the following *mean-field* assumption is made

$$q(R, C) = q(R)q(C) = \prod_{i=1}^{M}\prod_{j=1}^{N} q(R_i)q(C_j). \tag{3.7}$$

Moreover, since R_i and C_j follow Gaussian prior distributions (Eq. (3.2)), $q(\cdot)$ is assumed to be as follows

$$q(R_i) = g(R_i; \mu_i^R := h_{1,\phi}(Y_i, W^{(i,\cdot)}), S_i^R := h_{2,\phi}(Y_i, W^{(i,\cdot)})), \tag{3.8}$$

and

$$q(C_j) = g(C_j; \mu_j^C := l_{1,\iota}(Y^j, W^{(\cdot,j)}), S_j^C := l_{2,\iota}(Y^j, W^{(\cdot,j)})), \tag{3.9}$$

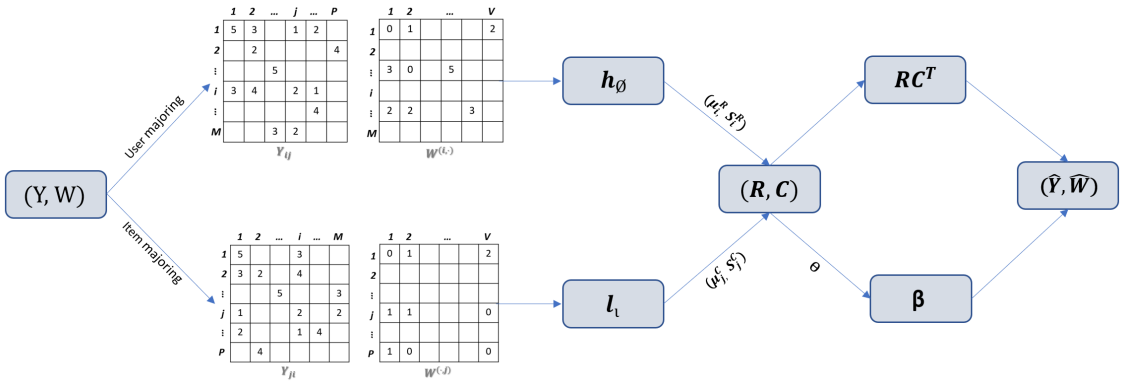


Figure 3.2 – A deep-learning-like model view of deepLTRS.

where $g(\cdot; \mu, S)$ is the pdf of a Gaussian multivariate distribution with mean μ and variance S. The two matrices S_i^R and S_j^C are assumed to be diagonal matrices with P elements. In addition, Y_i (respectively Y^j) denotes the i-th row (column) of Y, $W^{(i,\cdot)} := \sum_j W^{(i,j)}$ corresponds to a document concatenating all the reviews written by user i and $W^{(\cdot,j)} := \sum_i W^{(i,j)}$ corresponds to all the reviews about the j-th product. The functions $h_{1,\phi}$ and $h_{2,\phi}$ encode elements of $\mathbb{R}^{N+V}$ to elements of $\mathbb{R}^P$. Similarly, $l_{1,\iota}$ and $l_{2,\iota}$ encode elements of $\mathbb{R}^{M+V}$ to elements of $\mathbb{R}^P$. These functions are known as the network *encoders* parametrized by ϕ and ι, respectively.

Thanks to Eqs. (3.1)-(3.5)-(3.7)-(3.8)-(3.9) and by computing the KL divergence in Eq. (3.6), the *evidence lower bound* (ELBO) on the right hand side of Eq. (3.6) can be further developed as

$$\begin{aligned}
\text{ELBO}(\overline{\Theta}) = &\sum_{i,j} \left(\mathbb{E}_{q(R_i,C_j)} \left[-\frac{1}{2} \left(\frac{(Y_{ij} - (R_i^\top C_j + b_u^i + b_p^j))^2}{\eta^2} + \log \eta^2 \right) \right] \right) \\
&+ \sum_{i,j} \left(\mathbb{E}_{q(R_i,C_j)} \left[\left(W^{(i,j)} \right)^\top \log \left(\beta \sigma(f_\gamma(R_i, C_j)) \right) \right] \right) \\
&- \sum_i \left[-\frac{1}{2} \left(tr(S_i^R) + (\mu_i^R)^\top \mu_i^R - P - \log |S_i^R| \right) \right] \\
&- \sum_j \left[-\frac{1}{2} \left(tr(S_j^C) + (\mu_j^C)^\top \mu_j^C - P - \log |S_j^C| \right) \right] + \xi
\end{aligned} \tag{3.10}$$

where now $\overline{\Theta} := \{\eta^2, \gamma, \beta, b_u, b_p, \phi, \iota\}$ denotes the set of generative model *and* variational parameters, ξ is a constant term that includes all the elements not depending on $\overline{\Theta}$.

VAE structure. The deep view of deepLTRS is shown in Figure 3.2. We point out that, in the deep learning literature, the term *encoder* denotes a neural network that maps the observed data

into a lower dimension space [Kingma and Welling, 2019]. This is precisely what the functions $h_{1,\phi}, h_{2,\phi}$ and $l_{1,\iota}, l_{2,\iota}$ do, by mapping the observed data from $\mathbb{R}^{N+V}$ and $\mathbb{R}^{M+V}$, respectively, to the variational parameters in $\mathbb{R}^P$. Symmetrically, the term *decoder* denotes a neural network that maps the "compressed" data from the lower dimension space to the original dimension. In deepLTRS, this role is played by

— RC^T, the matrix product of R and C, that maps the lower dimension representations to the reconstructed ordinal data matrix $\hat{Y}$;

— β, which maps the topic proportions from $\mathbb{R}^T$ into vectors in $\mathbb{R}^V$ (the "reconstructed" rows of $\hat{W}$).

Unconstrained β. From a practical point of view, when optimizing the ELBO with respect to $\overline{\Theta}$, we remove the constraint on the columns of β, that have no longer to lie on the $V-1$ simplex. This assumption corresponds to the ProdLDA model. In order to obtain consistent parameters for the multinomial distribution followed by $W^{(i,j)}$, a softmax function $\sigma(\cdot)$ is applied to the product $\beta\theta_{ij}$ instead of θ_{ij} only.

3.3.2 Monte Carlo EM algorithm and mini-batching

The maximization of ELBO($\overline{\Theta}$) in Eq. (3.10) can be performed by means of a Monte Carlo EM algorithm that alternates a sampling step, to numerically approximate the expectations, with a maximization step to update the value of the parameters $\overline{\Theta}$. This algorithm was adopted previously for standard variational auto-encoders (VAEs) in Kingma and Welling [2014a]; Rezende et al. [2014]. As in those papers (and in contrast with what happens in simpler latent variable models), there is no close formula for the maximization step and gradient descent algorithms are employed to maximize the (Monte Carlo) lower bound with respect to $\overline{\Theta}$ [*]. Performing mini-batch optimization paired with stochastic gradient descent algorithms [Bottou, 2010] is necessary to reduce the computational burden when working with large data sets. However, there is a substantial difference between the model we adopt and standard VAEs. Whereas in a standard VAE the ELBO can be written as the sum of as many terms as the number of observations, ELBO($\overline{\Theta}$) in Eq. (3.10) does not factorize over the number of observations. In more detail, the model sees

∗. We point out that *maximizing* $ELBO(\overline{\Theta})$ with respect to $\overline{\Theta}$ is the same as *minimizing* $-ELBO(\overline{\Theta})$ and this is why we mention gradient *descent* algorithms.

the pair $(Y_{ij}, W^{(i,j)})$ as one observation. Assuming for simplicity that there is no missing data, the total number of observations is MN. The ELBO in Eq. (3.10) is unfortunately *not* the sum of MN terms due to the graphical structure of the generative model in Figure 5.1. Nevertheless, stochastic gradient descent can still be performed in our case thanks to what follows.

Let us define

$$\begin{aligned} z_i := & - \mathcal{KL}(q(R_i) \parallel p(R_i)) \\ & + \mathbb{E}_{q(R_i,C)} \left[\log p(Y_i | R_i, C, \overline{\Theta}) + \log p(W^{(i,.)} | R_i, C, \overline{\Theta}) \right], \end{aligned} \tag{3.11}$$

for all $i \in \{1, \dots, M\}$, with Y_i, R_i and $W^{(i,.)}$ previously defined. We now introduce a new random variable Z such that

$$\pi := \mathbb{P}\{Z = z_i\} = \frac{1}{M}. \tag{3.12}$$

A sample of Z corresponds to a uniformly at random extraction of one row in Y.

The proposition below states that the gradient of MZ with respect to **all parameters but** ι (the product encoder parameter) is an unbiased estimator of the ELBO's gradient with respect to the same parameters.

Proposition 1. *For the random variable Z, whose probability mass function is defined in Eq. (3.12), it holds that*

$$\mathbb{E}_\pi \left[\nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)} (MZ) \right] = \nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)} \left(\mathit{ELBO}(\overline{\Theta}) \right), \tag{3.13}$$

where $\nabla_x(f)$ denotes the gradient of a function $f(\cdot)$ with respect to the variable(s) x.

Proof.

First, let us notice that, due to the definition of z_i and the assumption in Eq. (3.7), it holds that

$$\begin{aligned} \nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)} z_i = \nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)} \; [& - \mathcal{KL}(q(R_i) \parallel p(R_i)) \\ & + \sum_{j=1}^{P} (\mathbb{E}_{q(R_i,C_j)} [\log p(Y_{ij} | R_i, C_j, \overline{\Theta})]) \\ & + \sum_{j=1}^{P} (\mathbb{E}_{q(R_i,C_j)} [\log p(W^{(i,j)}) | R_i, C_j, \overline{\Theta})]) \;]. \end{aligned} \tag{3.14}$$

Then, Eq. (3.12) leads to

$$\mathbb{E}_\pi \left(\nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)} MZ \right) = \sum_{i=1}^{M} \nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)} z_i,$$

and replacing Eq. (3.14) *into the above equation we obtain Eq.* (3.13)*, recalling that* $\nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)}(\cdot)$ *does not include the partial derivative with respect to* ι*. Thus*

$$\nabla_{(\eta^2,\gamma,\beta,\phi,b^u,b^p)}\left(-\mathcal{KL}(q(C_j) \parallel p(C_j)\right) = 0.$$

□

Similarly, the quantity

$$\begin{aligned} w_j := & -\mathcal{KL}(q(C_j) \parallel p(C_j)) \\ & + \mathbb{E}_{q(R,C_j)}\left[\log p(Y^j|R, C_j, \overline{\Theta}) + \log p(W^{(.,j)}|R, C_j, \overline{\Theta})\right], \end{aligned} \tag{3.15}$$

can be used to introduce an unbiased estimator of $\nabla_{(\eta^2,\gamma,\beta,\iota,b^u,b^p)}(\text{ELBO}(\overline{\Theta}))$, where the partial derivative of the lower bound with respect to ϕ (the user encoder parameter) is now not taken into account.

The above proposition justifies a maximization of the ELBO which alternates rows and columns mini-batching. First, rows of Y are extracted uniformly at random, with re-injection, in such a way that we obtain a collection of random variables $Z_1, Z_2, \ldots, Z_n, \ldots$, being i.i.d. copies of Z. Hence, each Z_n is used to update all the model parameters but ι. Moreover, when

ESTIM (Y, W)
$(R, C) \leftarrow \text{INIT}(\mathcal{N}(0, 1))$ */* Initialize with Gaussian distribution */*
Initialization parameters $\overline{\Theta}$
while $\log p\left(Y, W|R, C, \overline{\Theta}\right)$ increases **do**
  for all i: */* Row-majoring mini-batch */*
    $R_i \sim \mathcal{N}(h_{1,\phi}, h_{2,\phi})$ */* Sampling users */*
    $\overline{\Theta}_{-\iota} = \text{OPTIM}(\log p(Y, W|R_i, C, \eta^2, \gamma, \beta, b^u, b^p, \phi))$
  for all j: */* Column-majoring mini-batch */*
    $C_j \sim \mathcal{N}(l_{1,\iota}, l_{2,\iota})$ */* Sampling products */*
    $\overline{\Theta}_{-\phi} = \text{OPTIM}(\log p(Y, W|R, C_j, \eta^2, \gamma, \beta, b^u, b^p, \iota))$
end while
return $R, C, \overline{\Theta}$

Algorithm 3.1 – Mini-batch estimation of deepLTRS

performing row mini-batch, the current value of the whole matrix C is used, as well as all columns in Y and all the reviews by products. Conversely, when performing column mini-batch, the whole matrix R is used, as well as all rows of Y and all the reviews by users. In this case, the optimization is performed with respect to all the model parameters but ϕ. The pseudo code in Algorithm 4.1 summarizes the procedure of estimation detailed so far for deepLTRS.

3.4 Numerical experiments on simulated data

Before to go further, let us firstly describe the architecture of deepLTRS and the simulation setup that will be used later in this section.

3.4.1 Architecture and simulation setup

Architecture of deepLTRS. Our architecture involves three neural networks: two *encoders*, and an *internal neural net* working in the low dimensional space. The first encoder (user encoder) models h_ϕ in Eq. (3.8). It has two hidden layers, with 50 neurons each, equipped with a softplus activation function and followed by a dropout with a ratio of 0.2 and batch normalization. The second encoder (product encoder) models l_ι in Eq. (3.9). It also has two hidden layers, the number of neurons and subsequent operations are the same as the user encoder. The internal neural net models f_γ in Eq. (3.4). It has one hidden layer equipped with 80 neurons and a softmax activation function. In deepLTRS, the decoding phase is managed in a simpler way: i) *ratings*: a scalar product between R and C, followed by the intercepts summation decodes the ratings; ii) *reviews*: a linear map, involving the decoding matrix β, is followed by a softmax function in order to produce the probabilities of word occurrences. We stress that the decoding of reviews, from θ to the reconstructed matrix W, can be seen as a simple neural net with no hidden layers.

Simulation setup. An ordinal data matrix Y with $M = 750$ rows and $N = 600$ columns is simulated according to a latent continuous cluster model. The rows and columns of Y are randomly assigned to two latent groups, in equal proportions. Then, for each pair (i, j) corresponding to an entry of Y, the sampling of a Gaussian random variable Z_{ij} is detailed in Table 3.1a. In addition, the following thresholds $t_0 = -\infty, t_1 = 1.5, t_2 = 2.5, t_3 = 3.5, t_4 = +\infty$ are used to sample the

Table 3.1 – Simulation of ratings and reviews

(a) Score assignments.

	cluster 1	cluster 2
cluster 1	$\sim \mathcal{N}(2,1)$	$\sim \mathcal{N}(3,1)$
cluster 2	$\sim \mathcal{N}(3,1)$	$\sim \mathcal{N}(2,1)$

(b) Topic assignments.

	cluster 1	cluster 2
cluster 1	A	B
cluster 2	C	D

note $Y_{ij} \in \{1, ..., 4\}$ as

$$Y_{ij} = \sum_{k=1}^{4} k\mathbf{1}(Z_{ij})_{]t_{k-1},t_k[}. \tag{3.16}$$

Next, regarding the simulation of text reviews, four different texts from the BBC news are used to build a message for each note Y_{ij} according to the scheme summarized in Table 3.1b. One text is about the birth of Princess Charlotte, the second one is about black holes in astrophysics, the third one focus on British politics and the last one is about cancer diseases in medicine (denoted by A, B, C, D respectively).

Thus, when the user i in cluster $X_i^{(R)} = 2$ rates the product j in cluster $X_j^{(C)} = 1$, a random variable $Z_{ij} \sim \mathcal{N}(3,1)$ is sampled, then Y_{ij} is calculated via Eq. (3.16) and the review $W^{(i,j)}$ is built by random extraction of words from message C. All the sampled messages have an average length of 100 words. Finally, in order to introduce some noise, only 80% of words are extracted from the main topics, while the remaining 20% are extracted from the other topics uniformly at random.

The time complexity of deepLTRS is linearly affected by the number of users, products and the vocabulary. For instance, for data with $M = 750, N = 600, V = 1034$, the training time of 20 epochs is $1111.57s$, while with $V = 558$, the execution takes $637.96s$.

3.4.2 DeepLTRS with and without text data

A first experiment highlights the interest of using the reviews to make more accurate rating predictions. To do so, 10 data sets are simulated according to the above simulation setup, with sparsity rates of Y (i.e. proportion of missing data over MN entries) varying in the interval $[0.5, 0.99]$. The ratios of training set, validation set and test set are 80%, 10% and 10% of the observed (i.e. not missing) simulated data. Two versions of deepLTRS are fitted to the simulated

data, the first one accounting for both ordinal and text data (corresponding to the generative model described in Section 3.2) and the second one not using text, based on the PMF in Eq. (3.1). For the sake of simplicity, in both versions, as well as in all the experiments shown in this paper, b^u and b^p are fixed (and not estimated) to the average rating by users and products, respectively.

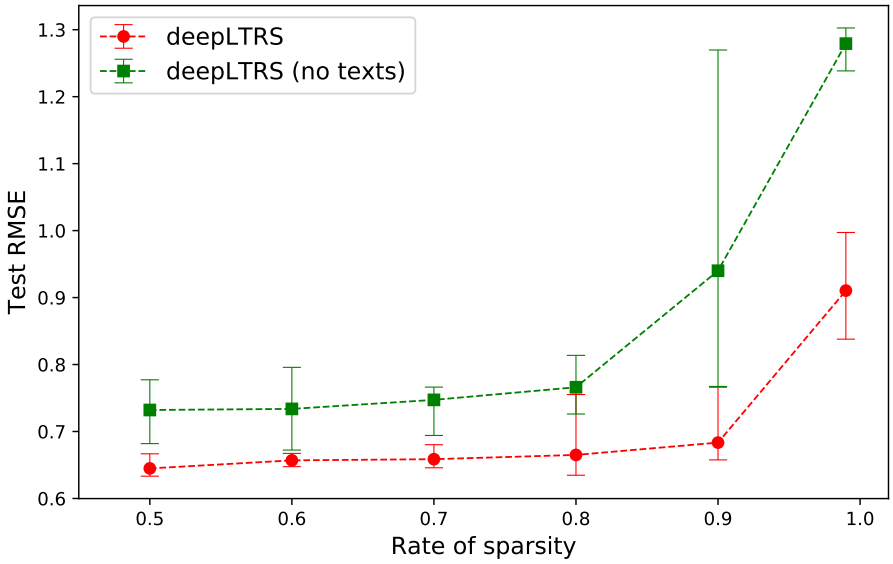


Figure 3.3 – Comparison of deepLTRS with and without text information.

Figure 3.3 shows the evolution of the test RMSE of deepLTRS, with and without using text data for rating forecasts, versus the data sparsity level. We can observe that, even though both models suffer from the high data sparsity (increasing RMSE), the use of the text greatly helps deepLTRS to maintain a high prediction accuracy for data sets with many missing values. Furthermore, the use of text reviews tends to reduce the variance of the deepLTRS predictions.

More intuitively, the visualization of user and product embeddings of deepLTRS with and without text data is provided as following. Figure 3.4 and Figure 3.5 show the t-SNE representations of R_i and C_j for deepLTRS with and without text data, respectively, with data sparsity of 0.99. We note that the two (row and column) clusters are well separated despite the large degree of sparsity in Figure 3.4, which is well representative of the simulation setup. However, without text data, the model does not capture well the structure of simulated data, as shown in Figure 3.5. The visualization effects demonstrate that the addition of text information is important and useful for deepLTRS.

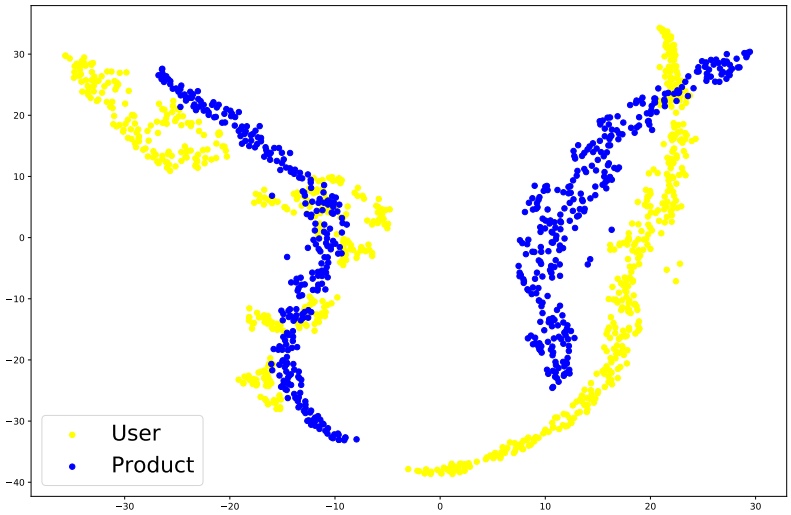


Figure 3.4 – Visualization of user and product embeddings of deepLTRS with text data (sparsity of 0.99).

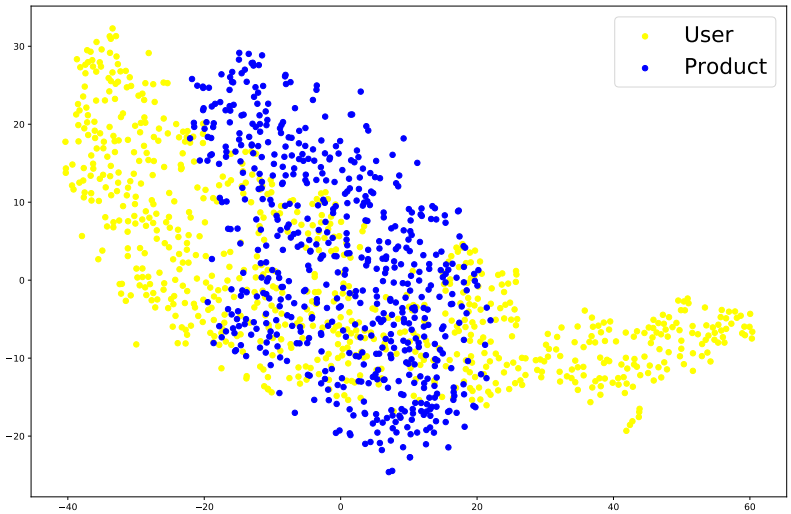


Figure 3.5 – Visualization of user and product embeddings of deepLTRS without text data (sparsity of 0.99).

3.4.3 Benchmark and effect of data sparsity

In this part, we benchmark deepLTRS by comparing with some state-of-the-art methods, in condition of high data sparsity. The same experimental setup was used to benchmark deepLTRS (from now on, always accounting for text data). Our model is here compared to HFT, HPF, CCPF and ALFM, introduced in Section 2.2. Since for CCPF, many combinations of sparsity and response models exist, we select the pair of models having the best performance.

Figure 3.6 shows the evolution of the test RMSE for deepLTRS and its competitors. We first remark that, although HFT accounts for the text reviews, it does not perform very well in our simulated scenario and turns out to be very sensitive to the data sparsity. Second, HPF also appears to be quite sensitive to the data sparsity and it always performs worse than CCPF, ALFM and deepLTRS. Finally, although CCPF and ALFM present less sensitivity to the data sparsity, as the changes in the RMSE are small along with the sparsity level, generally HFT performs better than them when the sparsity is less than a certain threshold. In addition, even if the sparsity reaches 0.99, deepLTRS still outperforms all other models. Let us recall that the simulation setup does not follow the deepLTRS generative model and therefore does not favor any method here.

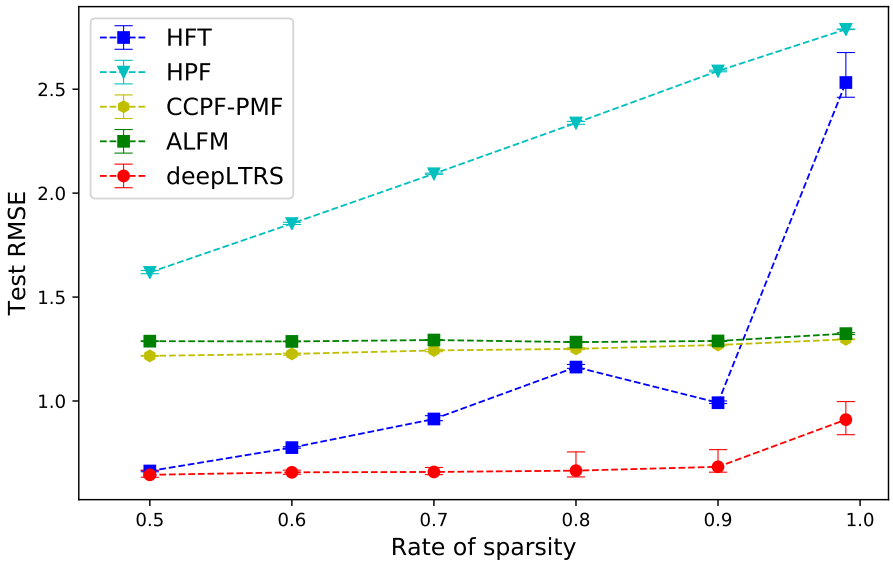


Figure 3.6 – Test RMSE of models with different sparsity level on simulated data.

3.5 Application on real-world data

We now consider applying deepLTRS to real-world datasets consisting of different product reviews from Amazon [†] [‡]. The data includes reviews (ratings, text), product metadata (descriptions, category, price, brand and image features) and links. In this section, deepLTRS is compared with previously mentioned models: HFT, HPF, CCPF, ALFM and TransNet.

Data pre-processing. In the pre-processing step, we kept records including user and product information, ratings and a plain-text review. For Amazon Fine Food dataset, we only considered users with more than 20 reviews and products reviewed by more than 50 users to obtain more meaningful information; for other three categories of Amazon product data, we used the reduced 5-core version where each of the remaining users and items have at least five reviews. Retained data were processed by removing all punctuations, numbers and stop words, then we deleted the words that appeared less than three times in the entire vocabulary for all data sets. The statistics of the processed data sets are given in Table 3.2.

Table 3.2 – Statistics of evaluation data sets.

Dataset	#users	#items	#reviews	#total_words	#rest_words	sparsity
Fine Foods	1643	1733	32811	5743	3047	98.85%
Musical Instruments	1429	900	10254	15050	5846	99.20%
Patio	1686	962	13258	22441	8746	99.18%
Automotive	2928	1835	20467	20113	7737	99.62%

Settings. Five independent runs of the algorithm were performed. For each run, we randomly selected 80% of the data as the training set, 10% samples for validation and the remaining 10% data as the test set. We trained our model for 100 epochs. As a method for stochastic optimization, we adopted an Adam optimizer [Kingma and Ba, 2014], with a learning rate of $2e^{-3}$. The RMSE is calculated on both the validation and test set. Reported test RMSE is obtained when the RMSE on the validation set was the lowest, as for all methods.

†. https://snap.stanford.edu/data/web-FineFoods.html
‡. https://jmcauley.ucsd.edu/data/amazon/

Rating prediction. Table 3.3 presents the test RMSE for deepLTRS and its competitors on the predicted ratings for Amazon data sets. Since HFT is restricted by the fact that the numbers of latent factors and topics should be equal, we set $D = T = 50$. First of all, HPF and CCPF only considered the user rating information. By replacing the single Poison distribution in HPF with a mixture model, CCPF has made great improvements in RMSE. Next, the remaining four methods all consider both ratings and reviews. Among them, TransNet and deepLTRS are deep-learning based models. It can be seen that, in general, ALFM and deepLTRS always have better performance than HFT and TransNet.

It is worth mentioning that deepLTRS outperforms ALFM on two data sets, Fine food and Patio, while ALFM has better performance on the other two data sets since when the data has many positive ratings, ALFM setup benefits from this configuration. Indeed, in the score generation phase, ALFM introduces the average of all ratings to the formula, which leads the predictions to a higher ranking level. When most of the scores of the experimental data are very positive, for example, a lot of scores are equal to 4 in Amazon data, ALFM can achieve very good results thanks to this average bias parameter. Nevertheless, when the score distribution of the data is more dispersed, ALFM cannot perform well, as seen in the following analysis.

Table 3.3 – Test RMSE on Amazon data sets.

Data sets	HFT	HPF	CCPF-PMF
Fine Food	1.4477 ($\pm$0.0465)	2.9528 ($\pm$0.0144)	1.2913 ($\pm$0.0105)
Musical Instruments	1.3505 ($\pm$0.0061)	4.0926 ($\pm$0.0164)	1.1151 ($\pm$0.0242)
Patio	1.2183 ($\pm$0.0096)	3.8782 ($\pm$0.0051)	1.1353 ($\pm$0.0174)
Automotive	1.0844 ($\pm$0.0084)	4.3252 ($\pm$0.0041)	1.0105 ($\pm$0.0186)
Average	1.2752 ($\pm$0.3729)	3.8122 ($\pm$0.5220)	1.1381 ($\pm$0.1020)

Data sets	ALFM	TransNet	deepLTRS
Fine Food	1.0705 ($\pm$0.0014)	1.3783 ($\pm$0.0012)	**0.9788** ($\pm$0.0215)
Musical Instruments	**0.8929** ($\pm$0.0013)	1.0912 ($\pm$0.0057)	0.9702 ($\pm$0.0143)
Patio	1.0219 ($\pm$0.0027)	1.0589 ($\pm$0.0009)	**0.9855** ($\pm$0.0319)
Automotive	**0.8797** ($\pm$0.0016)	1.0649 ($\pm$0.0012)	0.9299($\pm$0.0511)
Average	0.9663 ($\pm$0.0819)	1.1483($\pm$0.1334)	**0.9661** ($\pm$0.0392)

Analysis of the predictions. We analyze the differences between the prediction of ALFM and deepLTRS with the real ratings on the simulated data (discussed in 3.4.1) with the sparsity of 0.9. It is worth mentioning that, here we only compared ALFM with deepLTRS since ALFM has achieved significant performance on the Amazon data sets.

Table 3.4 – Statistics of the predictions

Data set	Models	Min.	Max.	Mean
Simulated data_0.9	ALFM	1.8843	3.1557	2.5075
	deepLTRS	0.6013	4.3340	2.5638

Table 3.4 demonstrates the statistics of the predictions. Different from ALFM that all predicted ratings are concentrated in the range of $[2, 3]$, the scores predicted by deepLTRS are well distributed in the interval $[1, 4]$, which is consistent with our initial setup. Moreover, Figures 3.7-3.8 illustrate the comparisons of the predictions and actual ratings, for ALFM and deepLTRS, respectively. It can be seen that the method used to calculate the predicted ratings in ALFM makes all predictions concentrated near the average. Thus, when the distribution of scores is very dispersed (as in the simulated data), the test RMSE will become very large. While many scores in the Amazon data are equal to 4, ALFM can achieve very good results.

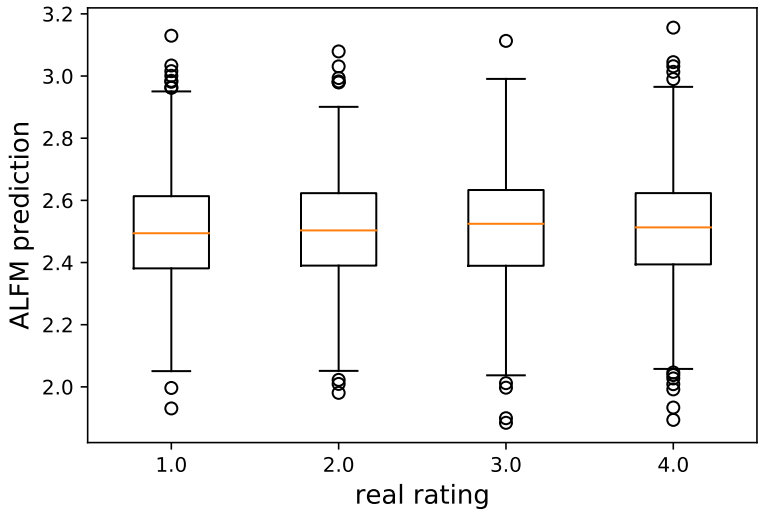


Figure 3.7 – Comparisons of the predictions of ALFM and actual ratings.

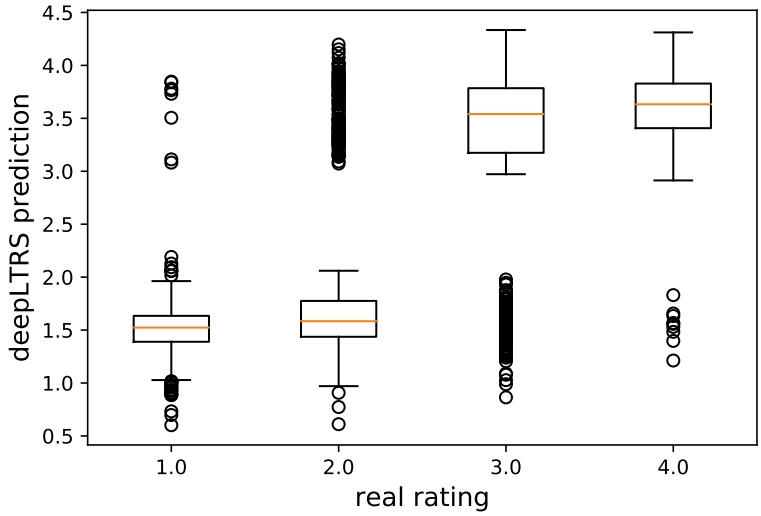


Figure 3.8 – Comparisons of the predictions of deepLTRS and actual ratings.

Interpretability on Amazon Fine Food. Figure 3.9 further presents a visualization with t-SNE of the high-dimensional latent representations ($P = T = 50$) of the users and products for the Amazon Fine food data. The overlapping regions of user and product representations correspond to users that are likely to comment on the corresponding products.

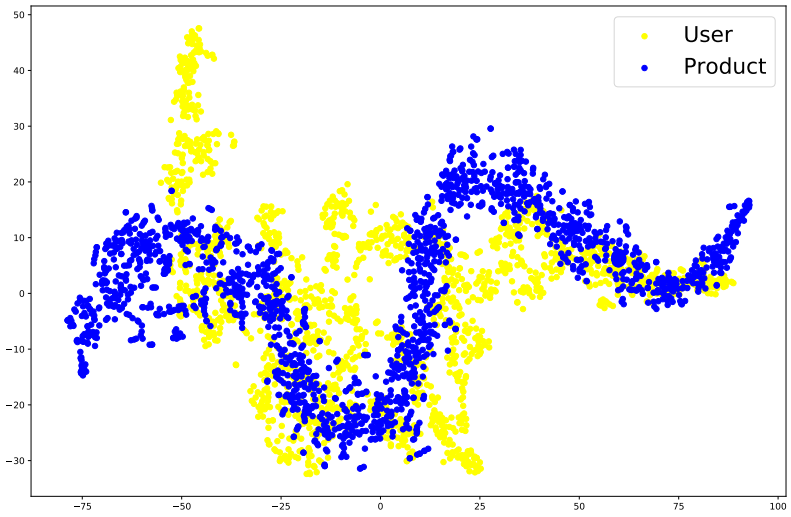


Figure 3.9 – Projection with t-SNE of user and product latent representations for the Amazon Fine Food data set.

In order to deeper understand the latent representations meaning, we provide in Figure 3.10 and 3.11 the visualization of user latent positions on two specific latent variables (variable 3 and 11) that can be easily interpreted according to average ratings and numbers of reviews the users give to products. Indeed, it clearly appears that variable 11 captures the rating scale of Fine food users whereas variable 3 seems to encode the user activity (number of reviews).

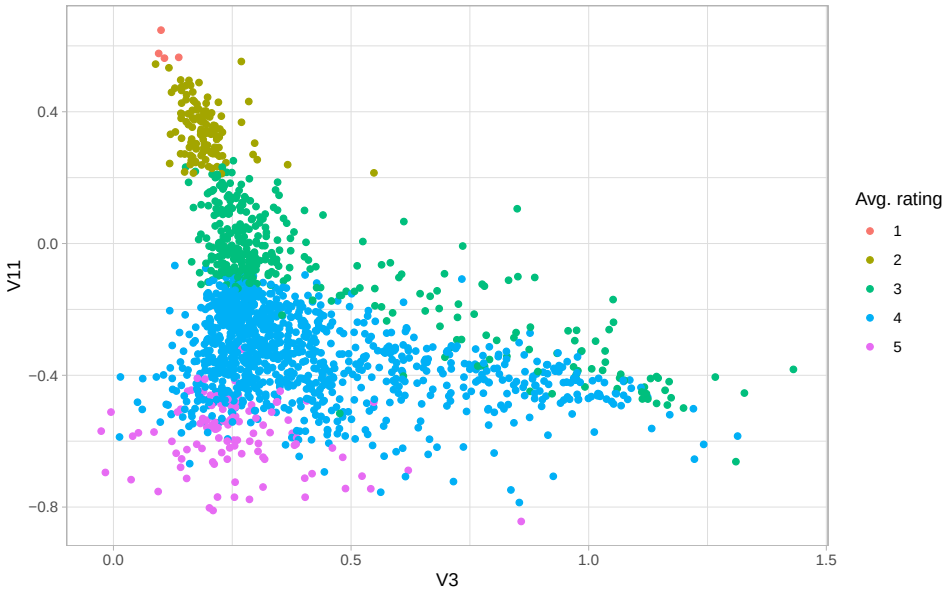


Figure 3.10 – Latent representation of users on variable 3 according to average ratings.

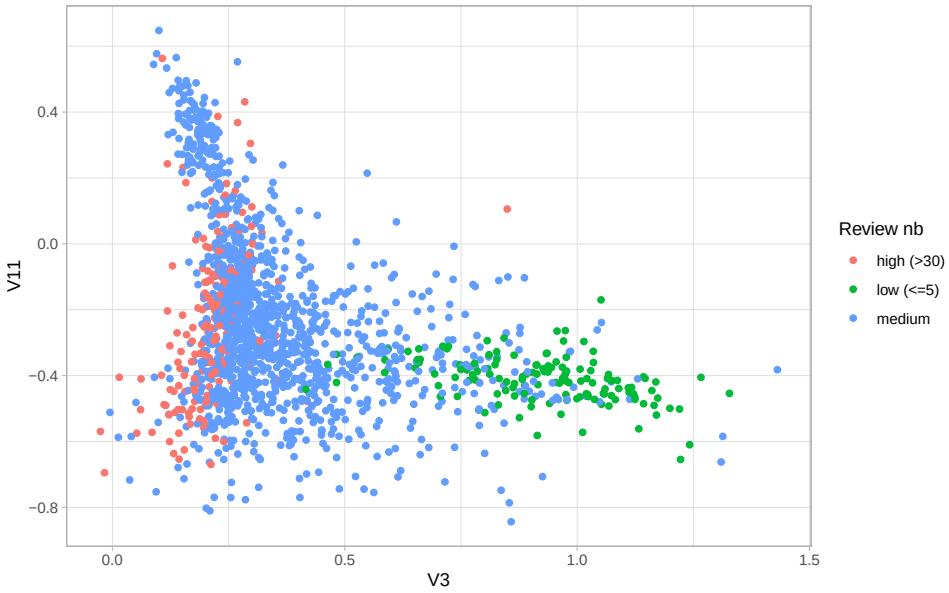


Figure 3.11 – Latent representation of users on variable 11 according to numbers of reviews they give to products.

3.6 Conclusion and perspectives

We introduced the deepLTRS model for rating recommendation using both the ordinal and text data available. Our approach adopted a VAE architecture as the deep generative latent model for both an ordinal matrix encoding the ratings, and a document-term matrix encoding the reviews. DeepLTRS presents the advantage to jointly learn representations of users and products through the alternated mini-batch optimization and a neural network was introduced to capture the relationship between latent factors and latent topics. Even with a simple topic model for the text part, we are still able to improve performance compared to the state-of-the-art techniques. Numerical experiments on simulated and real-world data sets show that our model outperforms other competitors in the context of high data sparsity.

We finally outline some research perspectives. First, although we mainly focused on the rating matrix, by exploiting the document-term matrix, the further ability of deepLTRS to predict the top words used by reviewers to comment products could be inspected in future works. Furthermore, in order to improve the modeling of text, we might replace LDA by a deep latent generative model, possibly involving RNNs [Agarap and Grafilon, 2018] or BERT [Xu et al., 2020] for review prediction. Moreover, the inference of the model parameters could be fine-tuned by means of the particle swarm optimization algorithm [Trelea, 2003] for self-adaptation of the hyper-parameters and some recent researches [Luo et al., 2019a,b, 2020] focusing on computational efficiency in the context of high-dimensional and sparse matrices in recommender systems could be considered in order to speed-up the learning process.

CHAPTER 4

Clustering by deep latent position model with graph convolutional networks

In this chapter, we introduce the deep latent position model (DeepLPM), an end-to-end generative clustering approach which combines the widely used latent position model (LPM) for network analysis with a graph convolutional network (GCN) encoding strategy. Moreover, an original variational inference procedure is introduced to explicitly optimize the posterior clustering probabilities and to implicitly optimize the other model parameters of encoder/decoder using stochastic gradient descent. Numerical experiments on simulated scenarios highlight the ability of DeepLPM to self-penalize the evidence lower bound for selecting the intrinsic dimension of the latent space and the number of clusters, demonstrating its clustering capabilities compared to state-of-the-art methods. Finally, DeepLPM is further applied to an ecclesiastical network in Merovingian Gaul and to the citation network Cora to illustrate the interest of our methodology to explore large and complex real-world networks.

This Chapter is related with one accepted communication in an international conference, ***Deep latent position model for node clustering in graphs****, Proceedings of the 30th European Symposium on Artificial Neural Networks (2022), and a submitted journal version,* ***Clustering by Deep Latent Position Model with Graph Convolutional Network****, Preprint HAL-03629104 (2022).*

4.1 Introduction 73
4.1.1 Organization of the chapter 75
4.2 Deep latent position model 75
4.2.1 Notations 75
4.2.2 Generative model 75
4.3 Model inference 77
4.3.1 Variational auto-encoding inference 77
4.3.2 Links with related models 78
4.3.3 Optimization 79
4.3.4 Model selection 83
4.4 Numerical experiments 84
4.4.1 Simulation setup 84
4.4.2 Benchmark study 85
4.4.3 Model selection 88
4.5 Analysis of a medieval network 90
4.5.1 Dataset 90
4.5.2 Results without covariates 91
4.5.3 Results with covariates 93
4.6 Cora citation network 96
4.6.1 Dataset 96
4.6.2 Results without covariates 97
4.6.3 Results with covariates 99
4.7 Conclusion and perspectives 104

4.1 Introduction

Networks are employed in a wide range of applications, from social media and email communications to protein-protein interactions, because they are simple structures yet are capable of modeling complex systems. In this context, vertex clustering is a key branch of clustering which attempts to partition the nodes of the graph into different groups to extract patterns summarizing the data. We consider the task of clustering the nodes of a network in this chapter.

As discussed in Section 2.3.1, a long series of statistical methods have been developed to discover the underlying communities in networks by learning the latent features of graph-structured data. More recently, deep learning-based models have emerged as a promising approach for analyzing large-scale networks and they have shown their abilities for representation learning purposes on data with complex structures, as shown in Section 2.3.2. The existing approaches for vertex clustering in networks can be split into two categories: probabilistic graphical models and deep latent variable models. We refer to Section 2.3 for a review of these methods.

In order to overcome the limitations of the methods described in that section, while exploring their benefits, we introduce a new deep latent position model (DeepLPM) for network data, allowing to simultaneously learn vertex representations and obtain node partitions. The LPM (see Section 2.3.1) assumes that a connection between two nodes depends on the distance between their positions in a latent space, which is intuitive and easy to understand. By combining a GCN encoder with a LPM-based decoder, our model aims at capturing the best of both worlds described so far: it is a flexible representation learning tool based on the deep learning architecture, yet comprehensive and interpretable thanks to the adopted statistical model. DeepLPM, that we propose here, has the following key-features:

— a LPM-based decoder models the probability of interactions between a pair of nodes as a function of the distance between them in a latent space. Compared with a standard inner-product-based decoder, this choice better preserves the network topology in different scenarios (see Figure 4.1);

— DeepLPM performs an *end-to-end* clustering of the nodes by estimating the posterior probabilities for cluster memberships. Thus, the inference procedure can automatically assign each node to a group without using any additional algorithms;

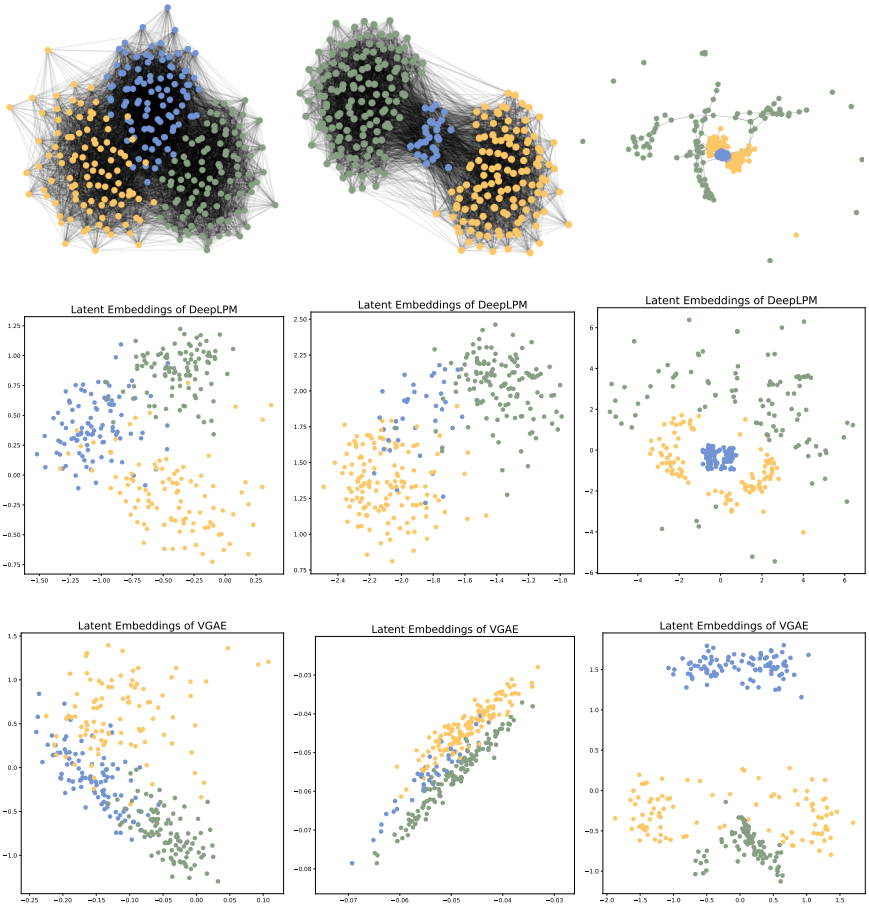


Figure 4.1 – Simulated networks and learned embeddings in three scenarios. From top to bottom: the original simulated graphs, the latent embeddings learned by DeepLPM and latent embeddings learned by VGAE (see Section 2.1.1). To facilitate the visualization, the latent dimension is here set to 2.

— an original estimation algorithm is designed to integrate the expectation maximization of the posterior clustering probabilities (explicit) and the stochastic gradient descent optimization for graph reconstruction (implicit);

— by combining the substantial representations learned by GCN with the position information, we point out the self-penalizing capability of DeepLPM in selecting the number of clusters and the latent space dimensionality, and demonstrate its effectiveness in performing different clustering tasks.

4.1.1 Organization of the chapter

In Section 4.2, we introduce the generative model DeepLPM. The variational inference and the original optimization algorithm are discussed in Section 4.3. Numerical experiments are provided in Section 4.4, highlighting the main features of our proposed approach and validating its self-penalization ability in model selection. An application to a real-world network coming from the Medieval history of Europe is presented in Section 4.5 and an analysis of the citation network Cora is described in Section 4.6. Section 4.7 finally concludes with a summary of this chapter.

4.2 Deep latent position model

In this section, the DeepLPM for end-to-end node clustering and network representation is first introduced.

4.2.1 Notations

In this work, networks are modeled as undirected, unweighted graphs $G = (V; E)$ with $N = |V|$ nodes. We introduce an $N \times N$ adjacency matrix A, where $A_{ij} = 1$ if there is a link between node i and node j, 0 otherwise. The set of edges E can be associated with an additional covariate information, collected into matrix $Y \in \mathbb{R}^{|E| \times U}$. The generic entry of Y, denoted y_{ij}, is a U-dimensional feature associated with the edge connecting i to j. For instance, y_{ij} could encode the text that author i sends to author j in a communication network. We aim at learning well-represented, latent, node embeddings Z in a lower dimension P and to partition the nodes into K clusters.

4.2.2 Generative model

As in LPM, we assume that each node $i = \{1, \cdots, N\}$ has an unknown position $z_i \in \mathbb{R}^P$ in a latent space and that the edges in the network are sampled independently given these positions. Moreover, the probability of a link between two individuals is modeled as a function of the distance between the two nodes, in the latent space. The generative process is as follows.

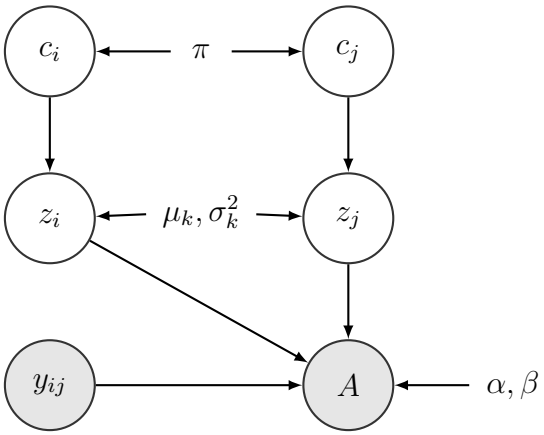


Figure 4.2 – Graphical representation of DeepLPM (variational parameters are not included).

First, each node is assigned to a cluster via a random variable c_i encoding its cluster membership

$$c_i \overset{\text{i.i.d}}{\sim} \mathcal{M}(1, \pi), \quad \text{with} \quad \pi \in [0,1]^K, \sum_{k=1}^{K} \pi_k = 1. \tag{4.1}$$

Then, conditionally to its cluster membership, a latent embedding vector z_i is generated

$$z_i | c_{ik} = 1 \sim \mathcal{N}(\mu_k, \sigma_k^2 I_P), \quad \text{with} \quad \sigma_k^2 \in \mathbb{R}^{+*}, \tag{4.2}$$

independently for each node, where μ_k and σ_k^2 denote the mean and variance for each cluster, I_P denotes an identity matrix in $\mathbb{R}^P$.

Finally, the probability of a connection between nodes i and j, as represented by adjacency matrix entry A_{ij}, is modeled through a Bernoulli random variable related with the distance between the corresponding latent positions

$$A_{ij} | z_i, z_j \sim \mathcal{B}(f_{\alpha,\beta}(z_i, z_j)), \tag{4.3}$$

with

$$f_{\alpha,\beta}(z_i, z_j) = \sigma(\alpha + \beta^\top y_{ij} - ||z_i - z_j||^2), \tag{4.4}$$

where $f_{\alpha,\beta}$ can be seen as a *decoding*, one-layer, neural network parametrized by α and β. Moreover, $\sigma(\cdot)$ is the logistic sigmoid function and y_{ij} is the covariate of the edge connecting i with j. A graphical representation of the generative model described so far can be seen in Figure 5.1.

4.3 Model inference

This section details the variational auto-encoding inference procedure and proposes an original estimation method which combines the explicit optimization of the posterior clustering probabilities and the implicit optimization of the neural network parameters.

4.3.1 Variational auto-encoding inference

Before getting into the details of the inference, we first denote by $\Theta = \{\pi, \mu_k, \sigma_k^2, \alpha, \beta\}$ the set of the model parameters introduced so far. A natural procedure would consist in maximizing the integrated log-likelihood of the observed data A with respect to Θ (and, possibly, Y, which is omitted to keep the notation uncluttered)

$$\log p(A|\Theta) = \log \int_Z \sum_C p(A, Z, C|\Theta) dZ. \tag{4.5}$$

Unfortunately, Eq. (5.9) is not tractable and we have to rely on a variational approach to approximate it

$$\log p(A|\Theta) = \mathcal{L}(q(Z, C); \Theta) + \mathcal{KL}(q(Z, C)||p(Z, C|A, \Theta)), \tag{4.6}$$

where $\mathcal{KL}$ denotes the Kullback-Leibler divergence between the true and approximate posterior distributions of (Z, C) given the data and model parameters. Then, in order to deal with a tractable family of distributions, $q(Z, C)$ is assumed to fully factorize (*mean-field* assumption)

$$q(Z, C) = q(Z)q(C) = \prod_{i=1}^{N} q(z_i)q(c_i). \tag{4.7}$$

Moreover, to benefit from the representational learning capabilities of GCN, we assume

$$q(z_i) = \mathcal{N}(z_i; \tilde{\mu}_\phi(\tilde{A})_i, \tilde{\sigma}^2_\phi(\tilde{A})_i I_P), \tag{4.8}$$

where $\tilde{\mu}_\phi(\cdot) : \mathbb{R}^{N\times N} \mapsto \mathbb{R}^{N\times P}$ (respectively $\tilde{\sigma}^2_\phi(\cdot) : \mathbb{R}^{N\times N} \mapsto \mathbb{R}^{+*}$) is the function mapping the normalized adjacency matrix $\tilde{A} = \hat{D}^{-\frac{1}{2}}(A + I_N)\hat{D}^{-\frac{1}{2}}$(we denote the degree matrix as $\hat{D}$ here to distinguish it from the edge feature dimension D) into the matrix of the variational means (and standard deviations). In the above equation, $\tilde{\mu}_\phi(\tilde{A})_i$ denotes the i-th row of $\tilde{\mu}_\phi(\tilde{A})$, corresponding to the variational mean for the latent position z_i (similarly for $\tilde{\sigma}^2_\phi(\tilde{A})_i$). We assume that the functions $\tilde{\mu}_\phi(\cdot)$ and $\tilde{\sigma}^2_\phi(\cdot)$ are parametrized by the GCN *encoder* g_ϕ.

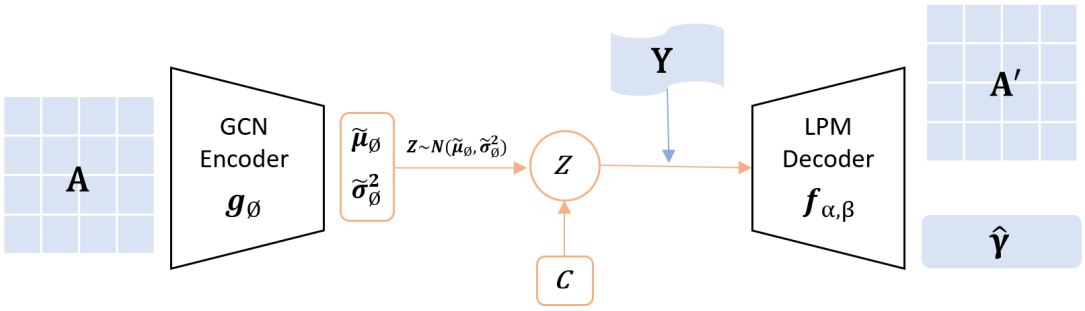


Figure 4.3 – A deep-learning-like model view of DeepLPM.

Finally, a standard assumption is made for the variational clustering probabilities

$$q(C) = \prod_{i=1}^{N} \mathcal{M}(c_i; 1, \gamma_i), \tag{4.9}$$

where γ_{ik} represents the variational probability that node i is in cluster k, with $\sum_{k=1}^{K} \gamma_{ik} = 1, \forall k = 1, \cdots, K$.

Model architecture. The variational structure of DeepLPM is shown in Figure 4.3. Within the framework of VAE, first the graph adjacency matrix A is taken as the model input and normalized; then, through the two-layer GCN encoder, we obtain the mean and variance of each node embedding; next, by minimizing the Kullback-Leibler divergence between the variational and the posterior distributions, we get the learned latent representations; finally, through the LPM-based decoder, we can reconstruct the matrix A via A' and obtain the cluster probability matrix $\hat{\gamma}$.

4.3.2 Links with related models

At this point, DeepLPM can be linked with the following models and therefore be seen as a generalization of theses approaches:

— In LPCM, a multivariate Gaussian prior distribution for the parameters denoted β is introduced, and the estimation is conducted using MCMC sampling. Conversely in DeepLPM, we introduce a decoding neural network $f_{\alpha,\beta}$, where the two parameters α and β are automatically optimized though stochastic gradient descent.

— Both VGAE and DeepLPM rely on the VAE architecture. However, instead of using a simple inner product decoder as in VGAE, DeepLPM involves a latent position-based $f_{\alpha,\beta}$

decoding strategy, which makes intuitive and understandable sense given that two nodes are more likely to connect if they are close in the latent space. DeepLPM also integrates the cluster memberships to achieve an end-to-end clustering.

— Both VaDE [Jiang et al., 2016] and DeepLPM model the data generative procedure with a Gaussian Mixture Model and a deep neural network, whereas in VaDE, both the decoder and the encoder are convolutional neural networks, limiting the model to image data. The decoding network $f_{\alpha,\beta}$ in DeepLPM is based on latent positions, while the encoder g_ϕ is a two-layer GCN that allow to model graph-structured data.

4.3.3 Optimization

In this part, we focus on maximizing the evidence lower bound (ELBO)

$$\mathcal{L}(A|\Theta) = \int_Z \sum_C q(Z,C) \log \frac{p(A,Z,C|\Theta)dZ}{q(Z,C)} \tag{4.10}$$

with respect to the model parameters Θ and the variational parameters ϕ. Thanks to Equations (5.11)-(4.8)-(5.14), Eq. (5.16) can be further developed as

$$\begin{aligned}
\mathcal{L} &= \int_Z \sum_C q(Z,C) \log \frac{p(A|Z,\alpha,\beta)p(Z|C,\mu_k,\sigma_k^2)p(C|\pi)dZ}{q(Z,C)} \\
&= \mathbb{E}\left[\log p(A|Z,\alpha,\beta)\right] + \mathbb{E}\left[\log p(Z|C,\mu_k,\sigma_k^2\right] \\
&+ \mathbb{E}\left[\log p(C|\pi)\right] - \mathbb{E}\left[\log q(Z|A)\right] - \mathbb{E}\left[\log q(C)\right] \\
&= \mathbb{E}\left[\log p(A|Z,\alpha,\beta)\right] + \mathbb{E}\left[\log \frac{p(Z|C,\mu_k,\sigma_k^2)}{q(Z)}\right] \\
&\quad + \mathbb{E}\left[\log \frac{p(C|\pi)}{q(C)}\right] \\
&= \mathbb{E}\left[\sum_{i\neq j} A_{ij}\log\eta_{ij} + (1-A_{ij})\log(1-\eta_{ij})\right] \\
&- \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik}\mathcal{KL}(\mathcal{N}(\tilde{\mu}_\phi(\tilde{A})_i, \tilde{\sigma}_\phi^2(\tilde{A})_i I_P)||\mathcal{N}(\mu_k,\sigma_k^2 I_P)) \\
&\quad + \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik}\log(\frac{\pi_k}{\gamma_{ik}}),
\end{aligned}$$

where $\eta_{ij} = \sigma(\alpha + \beta^\top y_{ij} - ||z_i - z_j||^2)$, $\mathcal{KL}(\cdot)$ denotes the KL divergence and the expectation is taken with respect to the variational probability $q(\cdot)$.

Explicit optimization. On the one hand, an *explicit* optimization of the ELBO with respect to the parameters $\gamma_{ik}, \pi_k, \mu_k$ and σ_k can be performed via Proposition 3.

Proposition 2. *The following variational updates can be obtained*:

$$\hat{\gamma}_{ik} = \frac{\pi_k \mathrm{e}^{-\mathcal{KL}_{ik}}}{\sum_{l=1}^{K} \pi_l \mathrm{e}^{-\mathcal{KL}_{il}}}, \tag{4.11}$$

where $\mathcal{KL}_{ik} = \frac{1}{2}\left\{\log \frac{(\sigma_k^2)^P}{(\tilde{\sigma}_\phi^2(\tilde{A})_i)^P} - P + \frac{\tilde{\sigma}_\phi^2(\tilde{A})_i}{\sigma_k^2} + \frac{1}{\sigma_k^2}||\mu_k - \tilde{\mu}_\phi(\tilde{A})_i||^2\right\}$.

Then

$$\hat{\pi}_k = \sum_{i=1}^{N} \gamma_{ik}/N, \tag{4.12}$$

$$\hat{\mu}_k = \sum_{i=1}^{N} \tilde{\mu}_\phi(\tilde{A})_i \gamma_{ik} / \sum_{i=1}^{N} \gamma_{ik}, \tag{4.13}$$

and

$$\hat{\sigma}_k^2 = \frac{\sum_{i=1}^{N} \gamma_{ik}(\sigma_\phi^2(\tilde{A})_i + ||\mu_k - \tilde{\mu}_\phi(\tilde{A})_i||^2)}{P \sum_{i=1}^{N} \gamma_{ik}}. \tag{4.14}$$

Proof.

Detailed derivations are given as follows. Under the equality constraint $\sum_{k=1}^{K} \gamma_{ik} = 1, \forall k$, *we use the method of Lagrange multipliers. Firstly, we introduce a Lagrange multiplier* λ_i

$$\tilde{\mathcal{L}} = \mathcal{L} - \sum_{i=1}^{N} \lambda_i \left(\sum_{k=1}^{K} \gamma_{ik} - 1 \right),$$

then, we derive $\tilde{\mathcal{L}}$ *according to* γ_{ik}

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \gamma_{ik}} = \log \pi_k - \log \gamma_{ik} - \frac{\gamma_{ik}}{\gamma_{ik}} - \mathcal{KL}_{ik} - \lambda_i = 0,$$

thus, we have

$$\begin{aligned} \log \gamma_{ik} &= \log \pi_k - 1 - \mathcal{KL}_{ik} - \lambda_i, \\ \gamma_{ik} &= \mathrm{e}^{\{\log \pi_k - 1 - \mathcal{KL}_{ik} - \lambda_i\}} = \frac{\mathrm{e}^{\{\log \pi_k - \mathcal{KL}_{ik}\}}}{\mathrm{e}^{\{1+\lambda_i\}}}. \end{aligned} \tag{4.15}$$

By using the constraint on $\sum_{k=1}^{K} \gamma_{ik}$*, we can get*

$$\sum_{k=1}^{K} \gamma_{ik} = \frac{\sum_{k=1}^{K} \mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}}}{\mathrm{e}^{\{1+\lambda_i\}}} = 1$$

$$\log \sum_{k=1}^{K} \mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}} = \log \mathrm{e}^{\{1+\lambda_i\}}$$

$$\lambda_i = \log \sum_{k=1}^{K} \mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}} - 1.$$

After putting the value of λ_i *into Eq. 5.22*

$$\gamma_{ik} = \frac{\mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}}}{\mathrm{e}^{\{1+\log \sum_{k=1}^{K} \mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}} - 1\}}} = \frac{\mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}}}{\sum_{k=1}^{K} \mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}}}.$$

Finally, we obtain

$$\hat{\gamma}_{ik} = \frac{\pi_k \mathrm{e}^{-\mathcal{K}\mathcal{L}_{ik}}}{\sum_{l=1}^{K} \pi_l \mathrm{e}^{-\mathcal{K}\mathcal{L}_{il}}}. \tag{4.16}$$

Similarly, since $\sum_{k=1}^{K} \pi_k = 1, \forall k$*, we introduce another Lagrange multiplier, let us say* τ

$$\tilde{\mathcal{L}} = \mathcal{L} - \tau \left(\sum_{k=1}^{K} \pi_k - 1 \right),$$

then, we derive $\tilde{\mathcal{L}}$ *according to* π_k

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \pi_k} = \sum_{i=1}^{N} \frac{\gamma_{ik}}{\pi_k} - \tau = 0,$$

next, we use the equality constraint to find the value of τ

$$\sum_{k=1}^{K} \sum_{i=1}^{N} \gamma_{ik} = \sum_{k=1}^{K} \pi_k \tau$$

$$\tau = N,$$

and finally, we have

$$\hat{\pi}_k = \sum_{i=1}^{N} \gamma_{ik} / N. \tag{4.17}$$

Lastly, we need to calculate the derivatives for μ_k and σ_k^2. We start by deriving $\tilde{\mathcal{L}}$ according to μ_k

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \mu_k} = -\frac{1}{2}\sum_{i=1}^{N}\gamma_{ik}\{\frac{1}{\sigma_k^2}(2\mu_k - 2\tilde{\mu}_\phi(\tilde{A})_i\} = 0,$$

then, we obtain

$$\begin{aligned} \mu_k \sum_{i=1}^{N}\gamma_{ik} &= \sum_{i=1}^{N}\tilde{\mu}_\phi(\tilde{A})_i\gamma_{ik}, \\ \hat{\mu}_k &= \frac{\sum\limits_{i=1}^{N}\tilde{\mu}_\phi(\tilde{A})_i\gamma_{ik}}{\sum\limits_{i=1}^{N}\gamma_{ik}}, \end{aligned} \tag{4.18}$$

and finally for σ_k^2, we have

$$\begin{aligned} \frac{\partial \tilde{\mathcal{L}}}{\partial \sigma_k^2} = -\frac{1}{2}\sum_{i=1}^{N}\gamma_{ik}\{\frac{P}{\sigma_k^2} - \frac{1}{\sigma_k^4}(\sigma_\phi^2(\tilde{A})_i + ||\mu_k - \tilde{\mu}_\phi(\tilde{A})_i||^2)\} = 0 \\ P\sum_{i=1}^{N}\frac{\gamma_{ik}}{\sigma_k^2} = \sum_{i=1}^{N}\frac{\gamma_{ik}}{\sigma_k^4}(\sigma_\phi^2(\tilde{A})_i + ||\mu_k - \tilde{\mu}_\phi(\tilde{A})_i||^2) \\ P\sum_{i=1}^{N}\gamma_{ik}\sigma_k^2 = \sum_{i=1}^{N}\gamma_{ik}(\sigma_\phi^2(\tilde{A})_i + ||\mu_k - \tilde{\mu}_\phi(\tilde{A})_i||^2) \\ \hat{\sigma}_k^2 = \frac{\sum\limits_{i=1}^{N}\gamma_{ik}(\sigma_\phi^2(\tilde{A})_i + ||\mu_k - \tilde{\mu}_\phi(\tilde{A})_i||^2)}{P\sum\limits_{i=1}^{N}\gamma_{ik}}. \end{aligned} \tag{4.19}$$

□

Implicit optimization. On the other hand, the *implicit* optimization of the encoder parameter ϕ and decoder parameters α, β is performed via stochastic gradient descent. In this work, it is implemented using the Adam optimizer [Kingma and Ba, 2014].

Algorithm. In the estimation process, we first conduct a pre-training step to avoid the model getting stuck in a local minima or a saddle point at the beginning of training. Then, the initial weights and biases after pre-training are saved for use in the training phase. Once we obtain the mean $\tilde{\mu}_\phi(\tilde{A})_i$ and variance $\tilde{\sigma}_\phi^2(\tilde{A})_i$ of each node, we use them to update the cluster information γ_{ik} by minimizing the KL divergence between the variational and the posterior distributions of each node.

Input: adjacency matrix A, edge features Y

pretrain_model = *pretrain*(A, 50 epochs) */* pre-training to initialize encoder/decoder weights */*

while $\mathcal{L}$ increases **do**

$\tilde{\mu}_\phi, \tilde{\sigma}^2_\phi = \text{GCN}(\tilde{A})$

update $\hat{\gamma}_{ik}$ by Equation (5.18) */* explicit optimization */*

update $\hat{\pi}_k, \hat{\mu}_k, \hat{\sigma}^2_k$ by Equations (5.19)-(5.20)-(5.21) */* explicit optimization */*

calculate the training loss (negative ELBO) $-\mathcal{L}$

update encoder parameter ϕ and decoder parameters α, β */* implicit optimization */*

end while

return reconstructed graph A', cluster probability matrix $\hat{\gamma}$

Algorithm 4.1 – Estimation of DeepLPM

Next, we adjust the mixture component π_k, mean μ_k and variance σ_k for each cluster according to the previous steps. Finally, the total loss is computed and the parameters of the encoder/decoder are optimized via stochastic gradient descent. More details are reported in Algorithm 4.1.

4.3.4 Model selection

The ELBO introduced in the previous section allows the estimation of the posterior law of (Z, C) for a fixed value of the latent dimension P and a fixed number of clusters K. If we vary these two parameters, the model can be considered as completely different. Therefore, choosing appropriate values for P and K is a crucial model selection task.

We emphasize that the self-regularization property of VAEs has already been observed in a number of studies [Kingma et al., 2016; Dai et al., 2017]. In the following Section 4.4, we conduct several experiments to show that DeepLPM does indeed benefit in practice from this property and that it induces a penalization on the ELBO in both the latent space variable (Z) and the clustering variable (C), thus allowing us to select the intrinsic dimension P of the latent space as well as the number K of clusters.

4.4 Numerical experiments

This section aims at emphasizing the effectiveness of this work on three synthetic datasets and at proving the validity of the estimation algorithm proposed in the previous Section 5.3.3.

4.4.1 Simulation setup

In order to simplify the characterization and to facilitate the reproducibility of the experiments, we designed three types of synthetic networks based on the generative models LPCM, SBM and from circle data, respectively:

— scenario A simulates data according to LPCM [Handcock et al., 2007]. 3 communities are considered and edges are generated based on the distance between each node position in dimension $P = 2$. We set a parameter $\delta \in [0.2, 0.95]$ to represent the rate of proximity between the clusters where a larger δ means that the three clusters are better separated. In this experiment, we set the mean of each cluster to

$$\begin{cases} \mu_1 = [0, 0] \\ \mu_2 = [1.5 * \delta, 1.5 * \delta] \\ \mu_3 = [-1.5 * \delta, 1.5 * \delta] \end{cases}$$

— scenario B simulates data according to SBM [Nowicki and Snijders, 2001]. It consists of one cluster with large probability of external connectivity and two communities that have a higher tendency to link within subset than across subsets. The connection probabilities are

$$\Pi = \begin{pmatrix} b & a & a \\ a & a & b \\ a & b & a \end{pmatrix}$$

where $a = 0.25$, $b = 0.01 + (1 - \delta^{'}) * (a - 0.01)$. We set another parameter $\delta^{'} \in [0.4, 1.0]$ to measure the degree of closeness where a larger $\delta^{'}$ means less overlap among the three clusters.

— scenario C considers networks created from 3 circular-structured data positions in dimension 2. Three circles have the same center and the different radius are 1, 5, and 10, respectively. Links are then generated based on the distance between node positions.

By varying the values of δ and $\delta^{'}$ in scenario A (assortative) and scenario B (dissortative), we can model the proximity between each cluster and thus test the robustness of our model in both simple and difficult cases. Then, contrary to standard communities, with strong transitivity (your-friend-is-my-friend effect), scenario C describes the construction of three groups of nodes with little transitivity in each.

4.4.2 Benchmark study

In this part, we aim at benchmarking DeepLPM with SBM [Nowicki and Snijders, 2001], LPCM [Handcock et al., 2007], VGAE [Kipf and Welling, 2016] and ARVGA [Pan et al., 2018] on simulated datasets in three scenarios. To facilitate the experiments, we do not consider the covariate information Y in simulated data, thus β in Eq. (4.4) is set to 0.

Datasets. In the "Easy" situation, scenario A was used with $\delta = 0.95$ and data from scenario B was created with $\delta^{'} = 0.9$. For the "Hard 1" situation, the values of δ and $\delta^{'}$ were set to be 0.6 for both scenario A and B. The value 0.4 was chosen in the situation "Hard 2". The number of nodes for scenario A and B were fixed to 300 and 600, respectively. Finally, in scenario C we simulated networks with 300 nodes.

Results. For each situation, we generated ten different networks and calculated the averaged adjusted rand index (ARI) [Hubert and Arabie, 1985]. Experimental results of clustering are shown in Table 5.1.

Table 4.1 – Experimental clustering results on 7 datasets.

	Easy		Hard 1	
Method	Sc.A	Sc.B	Sc.A	Sc.B
SBM	0.945±0.03	**1.000**±0.00	0.683±0.06	0.950±0.09
LPCM	0.922±0.03	0.769±0.15	0.613±0.06	0.540±0.04
VGAE	0.935±0.03	0.999±0.01	0.481±0.07	0.754±0.03
ARVGA	0.884±0.04	0.993±0.00	0.278±0.07	0.792±0.06
DeepLPM	**0.959**±0.01	**1.000**±0.00	**0.730**±0.03	**0.984**±0.01

	Hard 2		
Method	Sc.A	Sc.B	Sc.C
SBM	0.305±0.04	0.644±0.08	0.443±0.00
LPCM	0.324±0.07	0.345±0.03	0.415±0.20
VGAE	0.206±0.05	0.386±0.09	0.610±0.03
ARVGA	0.065±0.01	0.239±0.08	**0.631**±0.04
DeepLPM	**0.373**±0.04	**0.857**±0.02	0.625±0.03

First, focusing on scenario A, we can see that although the networks are simulated according to the LPCM model, LPCM does not exhibit the best performance. It only outperforms ARVGA in the simple cases; in Hard 1, it has better performance than VGAE; and in the more difficult Hard 2 case, it outperforms SBM. The ARVGA always obtains the worst performance in scenario A, which means it is not adaptive to assortative networks. Instead, DeepLPM always outperforms other competitors with different rate of proximity δ.

Second, considering scenario B, SBM is expected to have good performance in all models since the networks are simulated according to SBM. Indeed, it shows better performance than LPCM, VGAE and ARVGA in three situations. As a matter of fact, LPCM cannot find clusters on dissortative network structures and VGAE as well as ARVGA only work well in the simple situation. Again, DeepLPM shows the best performance in all cases with high ARI values.

Lastly, on the circular-structured data, all deep learning-based methods perform better than the ones based on statistical models. ARVGA presents the highest ARI compared to the other deep models. DeepLPM and VGAE have a slightly lower ARI. As networks based on the simulation of scenario C contains data from three circular structures, we cannot separate them linearly. Since deep learning-based models introduce non-linearity in the learning process, three deep models are more capable of clustering the nodes in scenario C than two statistical models.

Robustness. To further demonstrate the robustness of DeepLPM compared to other competitors, we varied the parameter δ from 0.2 to 0.95 and $\delta^{'}$ from 0.4 to 1 to compare the clustering performances for all models. Figure 4.4 illustrates the evolution of the clustering ARI in scenario A. DeepLPM consistently surpasses its competitors with different values of δ. VGAE and ARVGA, the other two deep models, perform worse than SBM and LPCM, the other two relying on statis-

tical modeling. Then, Figure 4.5 shows the evolution of the ARI in scenario B. As we can see, when $\delta^{'}$ is greater than 0.6, both DeepLPM and SBM can recover the true node partitions perfectly (ARI=1), whereas SBM cannot maintain its robustness when $\delta^{'}$ is less than 0.6. LPCM is unsuitable for this type of data, performing worse than VGAE and ARVGA. Additionally, DeepLPM has the highest ARI with a small variance in all situations. Finally, Figure 4.6 provides the embeddings learned by ARVGA, VGAE and DeepLPM with latent dimension equal to 2 in scenario C. It is clear that, different from the other two deep models, DeepLPM preserves the network topology of the three circular structures more effectively, by employing a latent position-based decoding strategy.

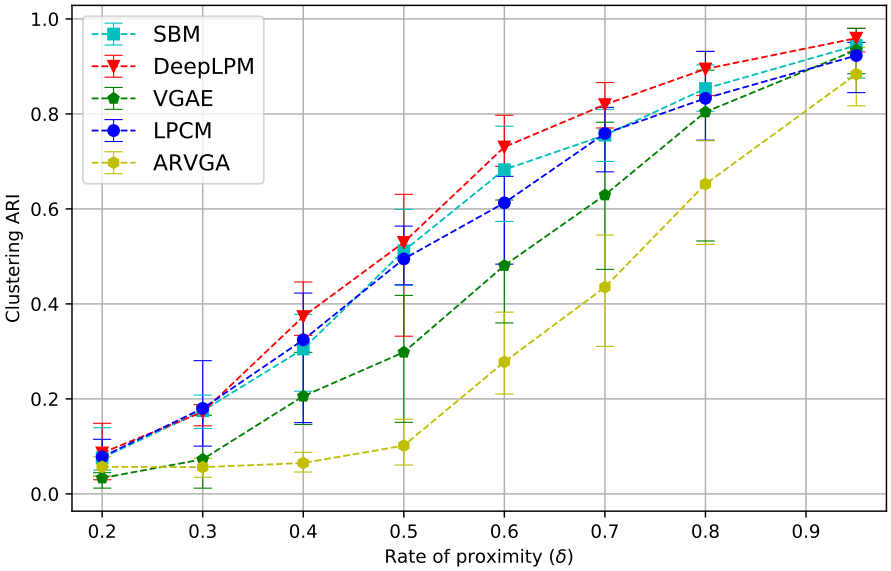


Figure 4.4 – Clustering ARI with different proximity rate δ in Sc.A on ten generated networks for each value of δ and averaged the values of ARI obtained.

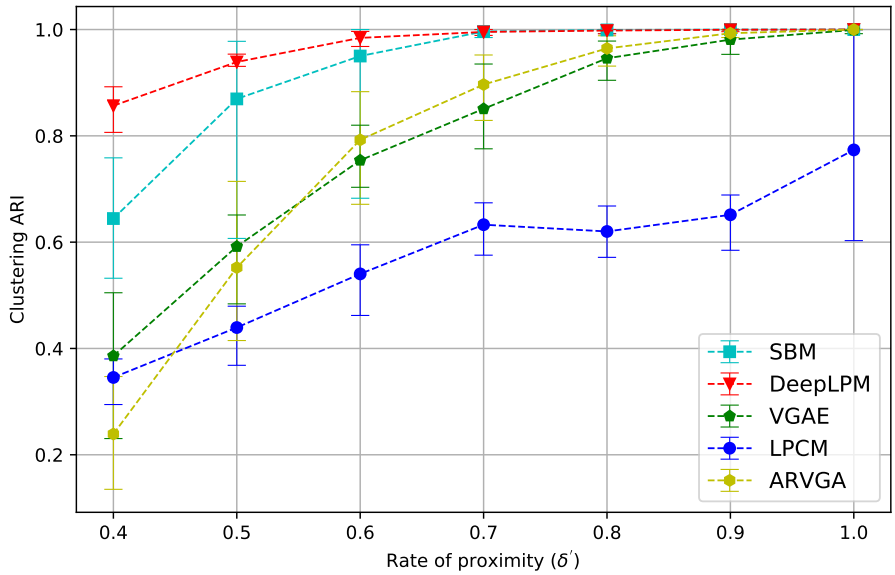


Figure 4.5 – Clustering ARI with different proximity rates δ' in Sc.B on ten generated networks for each value of δ' and averaged the values of ARI obtained.

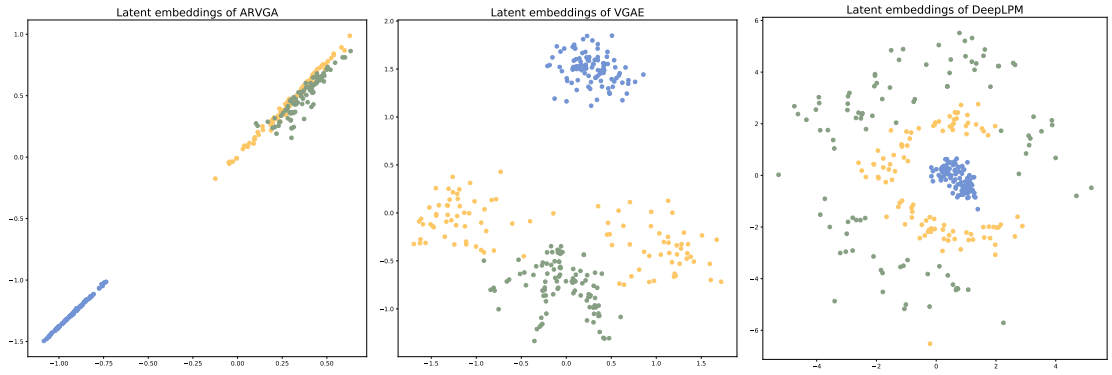


Figure 4.6 – From left to right: embeddings learned by ARVGA, VGAE and DeepLPM with latent dimension equal to 2 in Sc.C.

4.4.3 Model selection

A key element of an unsupervised learning technique such as DeepLPM is to be able to automatically determine both the latent dimension (P) and the number of clusters (K). We highlight here the ability of our methodology to auto-penalize the ELBO for selecting both the intrinsic dimension of the latent space and the number of groups appropriately.

Figure 4.7 shows the averaged training loss (negative ELBO) and ARI on 50 networks simulated according to scenario B ($\delta^{'} = 0.5$) with different latent dimensions ($P \in \{2, 4, 8, 16, 32\}$). We fixed the number of clusters to the actual value $K = 3$. As we can see, DeepLPM shows a minimal value of the negative ELBO when $P = 16$, which is also associated with the highest ARI.

Similarly, by varying the number of clusters from 2 to 6, Figure 4.8 illustrates how the training loss can also be used to find the appropriate number of clusters. In this experiment, we trained another 50 synthetic data in scenario B ($\delta^{'} = 0.5$) with the latent dimension $P = 16$. The results show that when $K = 3$, the training loss is minimal, thus recovering the actual value of K for the simulation setting.

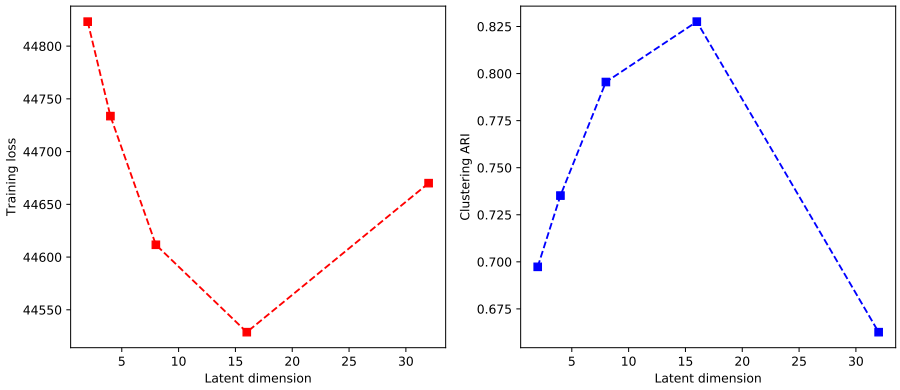


Figure 4.7 – Averaged training loss (negative ELBO) and ARI with different latent dimensions on 50 networks based on scenario B. When $P = 16$, a clear minimum of the negative ELBO can be noticed, corresponding to the highest averaged ARI.

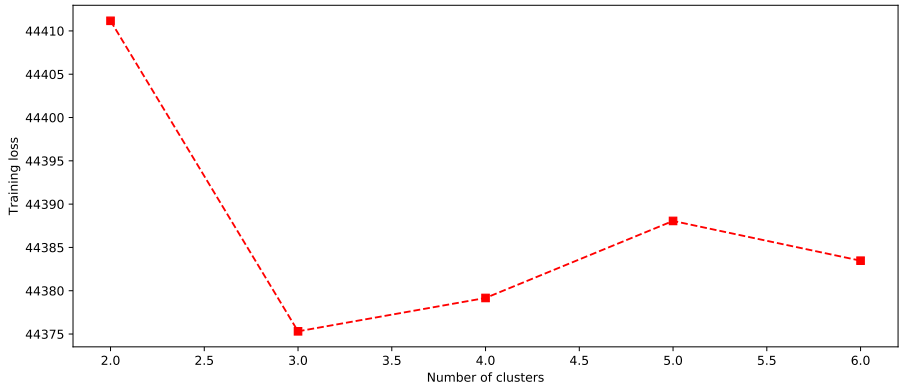


Figure 4.8 – Averaged training loss (negative ELBO) with different number of clusters on 50 synthetic data in scenario B. DeepLPM was able to estimate $K = 3$ by displaying a clear minimum of the negative ELBO, which recovered the actual number of clusters.

4.5 Analysis of a medieval network

As an illustration of the practical application of DeepLPM, it is first applied on a real-world dataset coming from historical science in this section.

4.5.1 Dataset

We consider the data set proposed by Jernite et al. [2014], which reports the ecclesiastical councils that took place in Merovingian Gaul during the 5th and 6th centuries. A council is an ecclesiastical meeting, usually called by a bishop, where issues regarding the church or the faith are addressed. The composition of these councils is known thanks to the acts written at the end of the meeting, and which were signed by all the attending members. The network contains $N = 1287$ individuals who held one or several offices in Gaul between the years 480 and 614, and who either have been related or have at least met during their lifetime. The number of edges is equal to 33,384. Figure 4.9 shows a visualization of the network highlighting the importance of the temporality in the relationships.

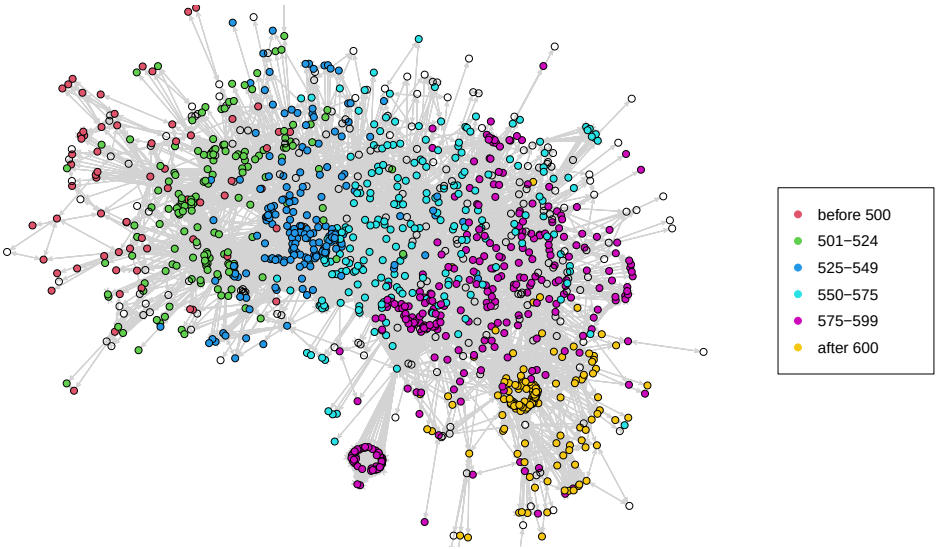


Figure 4.9 – Visualization of the ecclesiastical network, highlighting the temporality of the relationships. People living in distinct time periods during the 5th and 6th centuries are represented by different colors.

In addition to the interaction data, the data set also contains information about the individuals: period of activity, type of position and location. From this covariate information, we were able to build a 3-dimensional tensor Y encoding the similarities and differences between individuals. Thus, $Y_{ij}^{(1)}$ is equal to the number of years for which i and j have been active at the same time or, alternatively, the negative time lag (in years) between their period of activity; $Y_{ij}^{(2)} = 1$ if i and j were in the same region, -1 otherwise; $Y_{ij}^{(3)} = 1$ if i and j held a similar position (noble, ecclesiastical or other), -1 otherwise. As a result, those who share a greater number of active years, live in the same location, or hold similar positions are more likely to interact.

4.5.2 Results without covariates

We first analyze hereafter the clustering results without taking into account the covariate information encoded in Y. DeepLPM was applied to this network for various numbers of groups K (varying from 2 to 10) and a fixed number of latent space dimensions ($P = 16$). When we ignore the covariate information, the evolution of the training loss shows a clear minimum at $K = 9$.

Figure 4.10 depicts the visualization of node partitioning into 9 groups found by DeepLPM without the use of covariates. It is worth noticing that DeepLPM has not been influenced by the temporality since it was able to detect communities that played a similar role in the network

at different periods. For instance, the groups #3 and #5 gather people who lived at different and not overlapping time periods. In Figure 4.11, we also show the distributions in each group when personal roles are taken into consideration. In particular, we can notice that the group #5 is essentially made of ecclesiastics contrary to all the other clusters. We also point out that most nobles (in proportion) were found in the group #4. In addition, it can be seen that citizens played a significant role in the group #2.

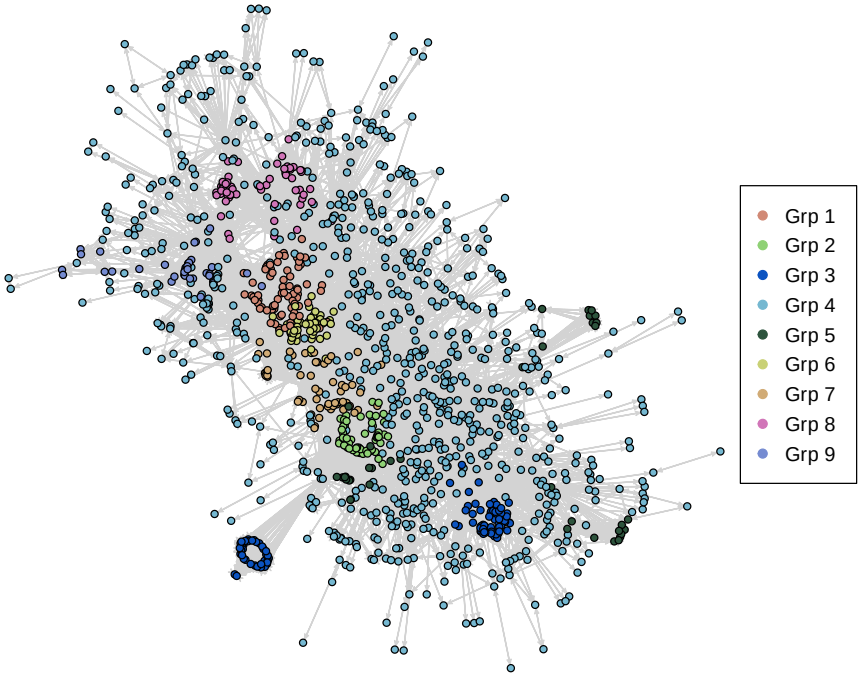


Figure 4.10 – Node partitions without the covariate information on medieval data. Nine clusters are represented in different colors. DeepLPM was able to find communities at multiple time periods without being influenced by the network temporality; groups #3 (dark green), #4 (cyan), and #5 (dark blue) in particular are clusters that share the same color while lying in distinct periods.

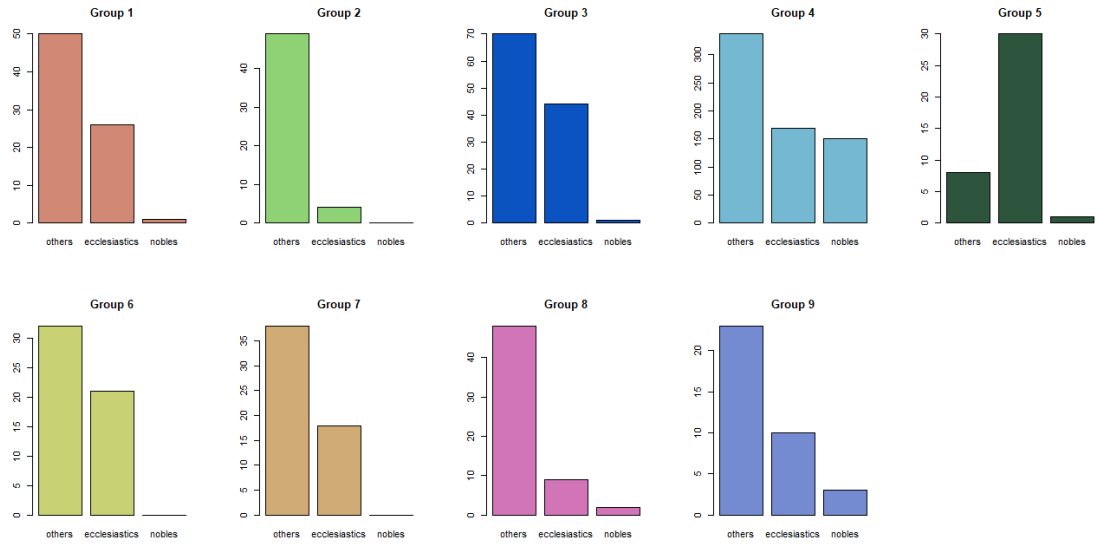


Figure 4.11 – Distributions in each group based on personal roles on medieval data. Nobles are generally found in groups #4 and #9, ecclesiastics play a large role in group #5, and civilians predominate in group #2.

4.5.3 Results with covariates

To demonstrate the interest of taking into account the covariates, we integrated the covariates encoded in a 3-dimensional vector Y into DeepLPM and performed clustering on this network. The number of groups also varied between 2 to 10, with a same intrinsic dimension equals to 16. When edge features are added to the methodology, the number of groups is estimated to be $K = 8$ with minimal loss. Compared to previous Section 4.5.2, the number of groups is reduced by one.

Confusion matrix. We first plot the confusion matrix between the predicted labels at $K = 9$ without covariates and $K = 8$ with covariates to investigate the fusion or dispersion between multiple clusters, as shown in Figure 5.10. We can see that, by introducing the covariate, the cluster $N1$ (for new $\#1$) gathers people from clusters $O3$ (for old $\#3$ without covariates), $O6$ and $O8$ together; then, it separates the individuals from $O8$ into $N1$, $N3$ and $N6$; and all the people in $N5$ and $N8$ come from $O4$ and $O9$, respectively. Therefore, the exploitation of the covariate information results in personnel switching between groupings and a reduction in the number of clusters, allowing DeepLPM to focus on patterns that are not explained by the covariates alone.

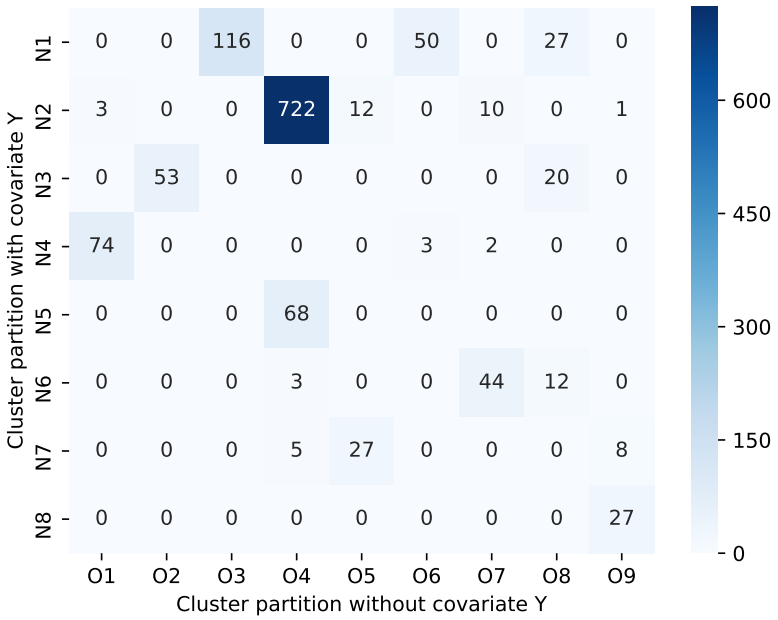


Figure 4.12 – Confusion matrix between cluster partitions with and without covariate. Individuals from $O3$ (for old $\#3$ without covariates), $O6$, and almost half of $O8$ congregated in $N1$ (for new $\#1$ with covariates), reducing one cluster.

Visualization and analysis. Figures 4.13 shows the clustering visualization obtained by DeepLPM for 8 groups with covariates. We can see that DeepLPM was able to detect communities that played a similar role in the network at different periods. Indeed, the groups $N1$ (black), $N2$ (red), $N3$ (green), $N5$ (cyan), $N6$ (pink) and $N7$ (yellow), all gather people who lived at different and not overlapping time periods. The red group $N2$ is particularly representative of this since it covers the whole period (480-614 of our era).

In addition, distributions within each group based on personal functions are also shown in Figure 4.14. Firstly, we can observe that the majority of people in groups $N3$ and $N8$ were civilians; then, it can be noticed that ecclesiastics was a significant component of the group $N7$; Furthermore, we point out that groups $N2$ and $N8$ included individuals from all strata of society.

Comparisons of results without and with Y. We further analyze the results by comparing the partitions without and with covariates in Figures 4.10 and 4.13. Firstly, we can see that the group $N1$ (black) in Figure 4.13 extracted some individuals from groups $O6$ (for old $\#6$) and $O8$ in

Figure 4.10 and kept people in the group $O3$. Combining with Figures 4.11 and 4.14, the group $N1$ in Figure 4.13 is specific since it only gathers people from clergy and civilians, who were probably discussing some central questions about the faith during different periods. Then, the group $N3$ (green) retained individuals from cluster $O2$ and retrieved some civilians from cluster $O8$, indicating that they were possibly addressing civilian issues. Similarly, the group $N6$ (pink) kept the majority of its members in $O7$ while also bringing in people from $O4$ and $O8$. They may share religious concerns in relation to their positions. Likewise, the group $N5$ (cyan) in Figure 4.13 is specially separated from the big group $O4$ in Figure 4.10, which is made of a relatively significant proportion of nobles, in particular kings and queens, implying that this group was discussing political or nobility matters.

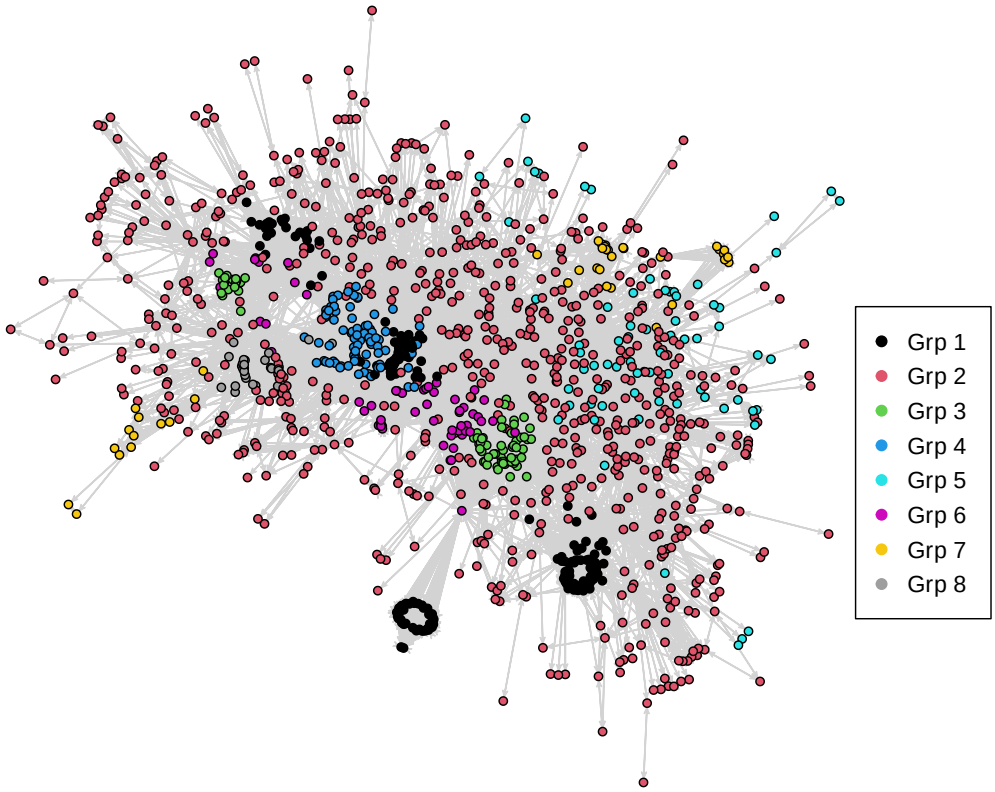


Figure 4.13 – Visualization of cluster partitions with covariates on medieval data. Eight clusters are represented in different colors. DeepLPM was able to find communities at multiple time periods without being influenced by the network temporality; groups $N1$, $N2$, $N3$, $N5$, $N6$ and $N7$, for example, are clusters with the same colors but located at different times.

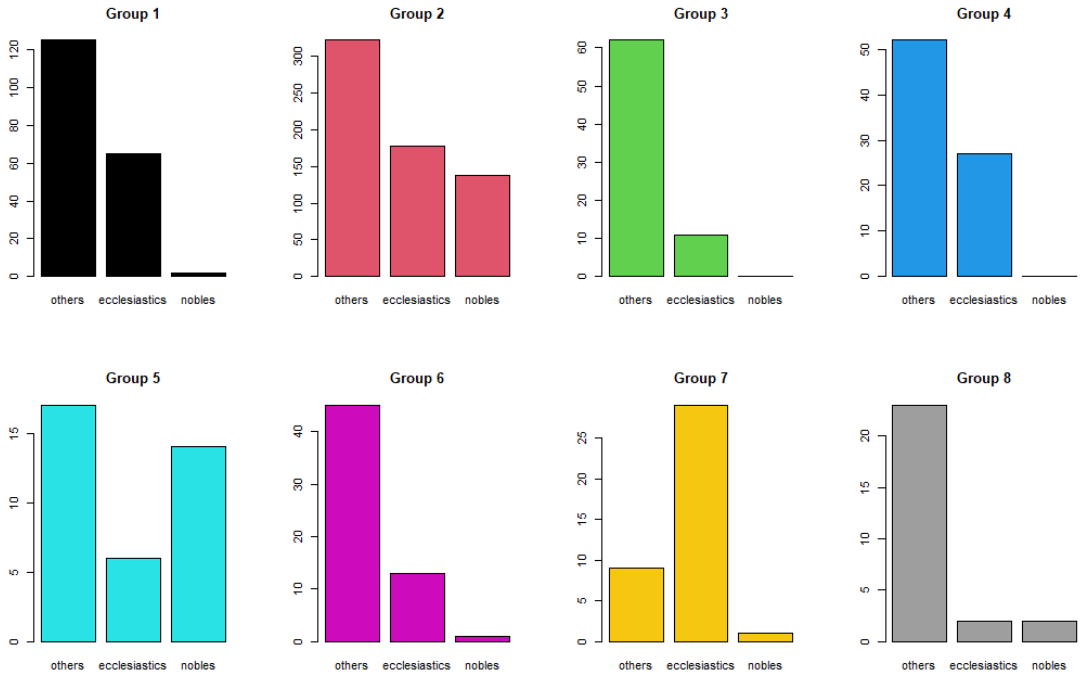


Figure 4.14 – Distributions in each group based on personal roles on medieval data. Civilians are generally found in groups $N3$ and $N8$, ecclesiastics play a large role in group $N7$, and all social classes are represented in groups $N2$ and $N5$.

4.6 Cora citation network

In this section, we also conduct an unsupervised analysis on a widely used scientific citation network.

4.6.1 Dataset

The Cora dataset [†] has been analyzed with several embedding and clustering (deep) methods. The dataset contains 2,708 scientific publications classified in seven classes: *case based*, *genetic algorithms*, *neural networks*, *probabilistic methods*, *reinforcement learning*, *rule learning* and *theory*. The citation network consists of 5,429 links and each publication is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from a dictionary.

†. https://relational.fit.cvut.cz/dataset/CORA

Most related works [Pan et al., 2018; Mehta et al., 2019] assume that the number of clusters is equal to the number of classes used in supervised classification tasks, whereas we argue that the class labels might not be in a one-to-one relation with the detected communities in unsupervised clustering. Instead, an appropriate cluster number should be obtained through model selection. Thus, we decided to use the class membership of each paper to build a tensor Y of dimension $D = 7 \times 7$ encoding the similarities and differences between articles. For each pair of papers i and j with category labels s_i and s_j, $Y_{s_i s_j} = 1$ indicates that paper i belongs to the class s_i and j belongs to the class s_j, 0 otherwise.

4.6.2 Results without covariates

We first performed clustering without considering the covariate information. DeepLPM was fitted to this network for different numbers of groups, ranging between 5 and 11, and fixed latent dimension ($P = 16$) for the latent space. The number of groups is estimated to be $K = 9$ with minimum loss.

A visualization of the latent embeddings learned by DeepLPM is shown in Figure 4.15. We can see that even though groups #4 and #5 are very close to each other, and there are some overlaps between groups #2 and #9, DeepLPM globally produces discriminant embeddings. Then, to get an idea of the composition of each group, the distribution of the papers in the nine clusters according to the seven categories is given in Figure 4.16. In particular, groups #1, #4, #6, #7 and #8 focus on subjects related to *neural networks*, *theory*, *probabilistic methods*, *case based* and *genetic algorithms*, respectively; the group #9 is mainly based on *reinforcement learning*.

Without considering covariates, it is clear that most clusters, such as #1, #4, #6, #7, #8 and #9 contain primarily one category of papers, which coincides with the known supervised information. However, we also see groups #2, #3 and #5 containing more categories. This is not surprising. Indeed, when looking at papers from some peculiar categories (e.g. neural nets, probabilistic methods or theory) we discover that they cover topics from other categories. For instance, several probabilistic approaches are build upon neural networks, or some theoretical papers can refer to neural network-based techniques. The clusters containing papers from different categories, clearly account for this "contaminations". Therefore, we want to give DeepLPM the known class labels and let the model dig for more information hidden behind them.

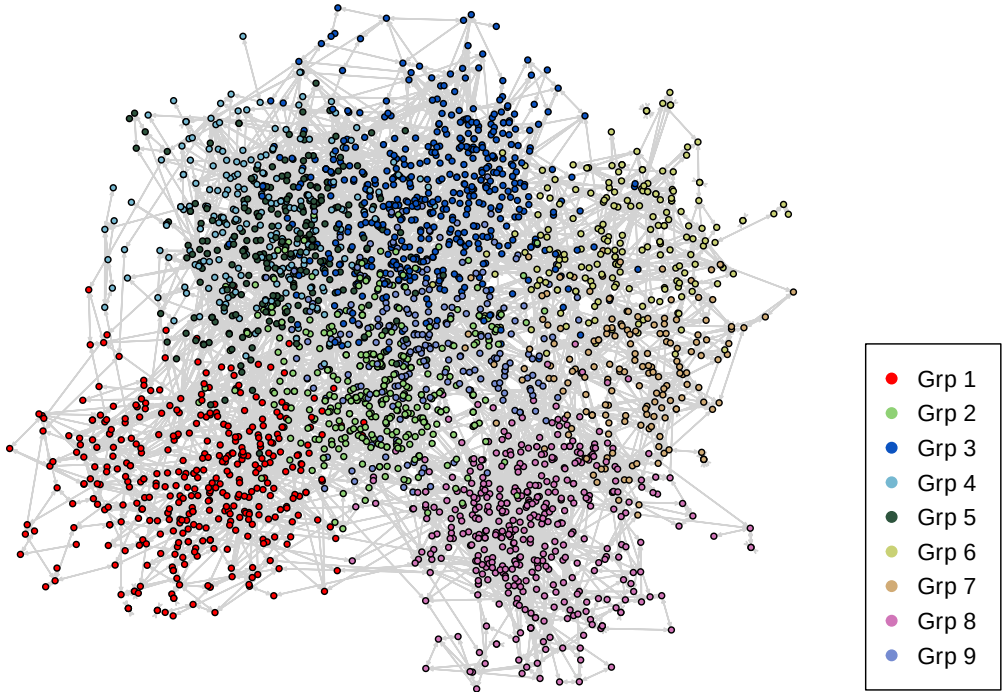


Figure 4.15 – Visualization of the clustered embeddings without covariates on Cora. Nine clusters are represented in different colors.

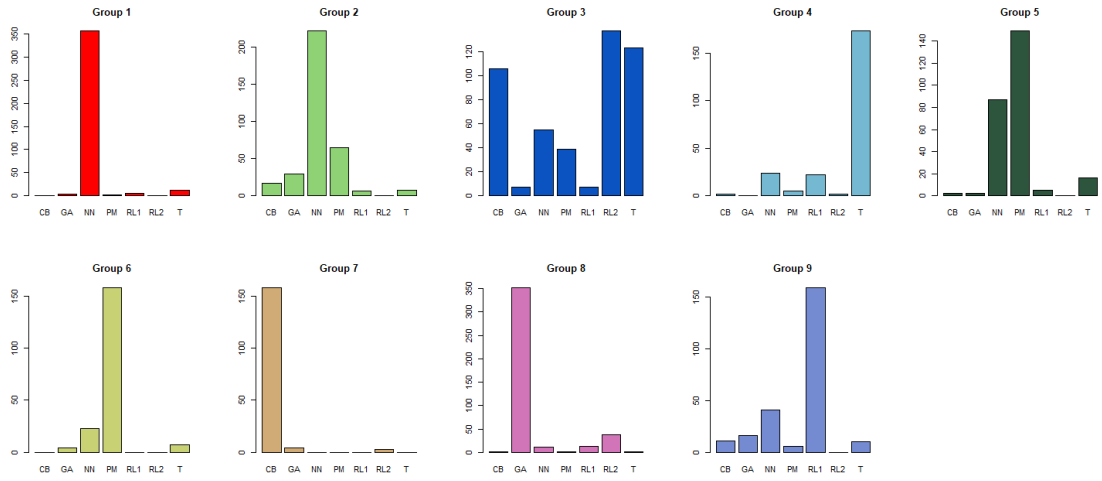


Figure 4.16 – Partitions without covariates taking into account the classes in each group on Cora. Most groups consist primarily of one type of paper, while a few groups include multiple categories. Here CB: Case_Based, GA: Genetic_Algorithms, NN: Neural_Networks, PM: Probabilistic_Methods, RL1: Reinforcement_Learning, RL2: Rule_Learning, T: Theory.

4.6.3 Results with covariates

To show the impact of the covariate information, we now adopt the covariates Y. The model selection was also conducted by varying the number of clusters from 5 to 11, with the dimensionality of the latent space equal to 16. Based on the evolution of the training loss, the number of groups was estimated to be $K = 6$. Thus, with this additional covariate, the clusters number is reduced by three.

Confusion matrix. We first plot the confusion matrix between the predicted labels at $K = 9$ (without covariates) and $K = 6$ (with covariates) to investigate the fusion or dispersion between multiple clusters, as shown in Figure 4.17. As we can see, the cluster $N1$ (for new #1) extracted papers from 9 different clusters $O1$ (for old #1) to $O9$, especially from $O4$ and $O9$; $N2$ assembled publications mainly from clusters $O1$ and $O2$; besides, most of the items in $N3$ and $N5$ come from $O7$ and $O3$, respectively; finally, $N4$ consists of quantitative papers from $O1$, $O2$ and $O8$, $N6$ is mainly composed of articles of $O2$, $O5$ and $O6$. The fact that each cluster now contains multiple categories of publications demonstrates that the addition of covariates helps to reveal hidden patterns behind supervised class information when performing clustering.

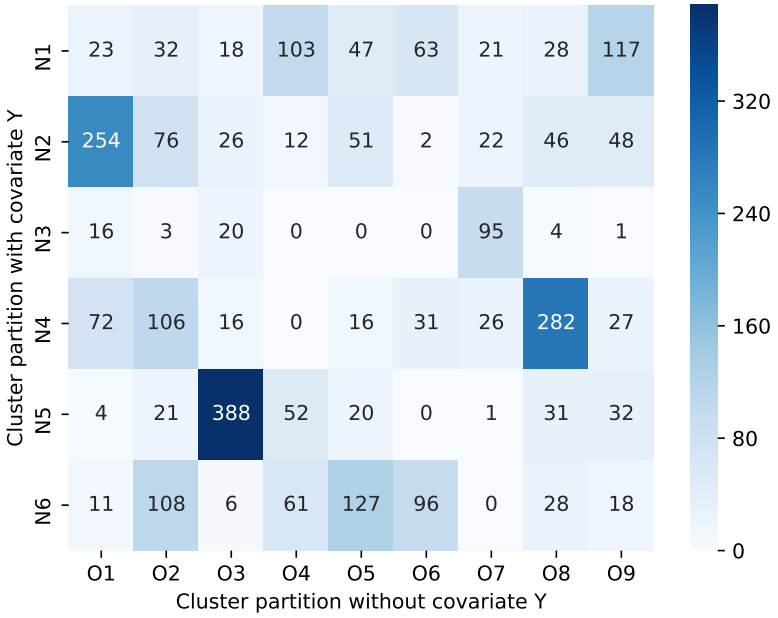


Figure 4.17 – Confusion matrix between the estimated clusters with and without covariates. The previous nine clusters are reduced to six with the addition of covariates.

Visualization and analysis. The visualization of the latent embeddings learned by DeepLPM is shown in Figure 4.18. Figure 4.19 shows the paper distributions when considering the class labels for six groups. In contrast to Figure 4.16 where each group contains principally one or two different classes, it is clear that, due to the introduction of paper labels as covariates new similarities between papers in different categories emerge.

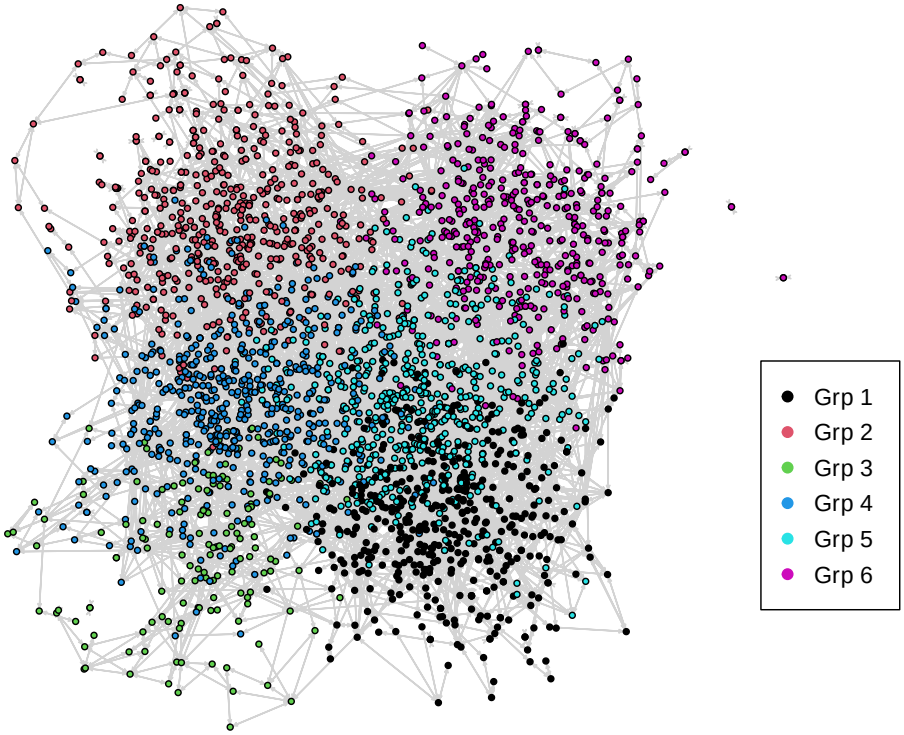


Figure 4.18 – Visualization of the clustered embeddings with covariates on Cora. Six clusters are represented in distinct colors.

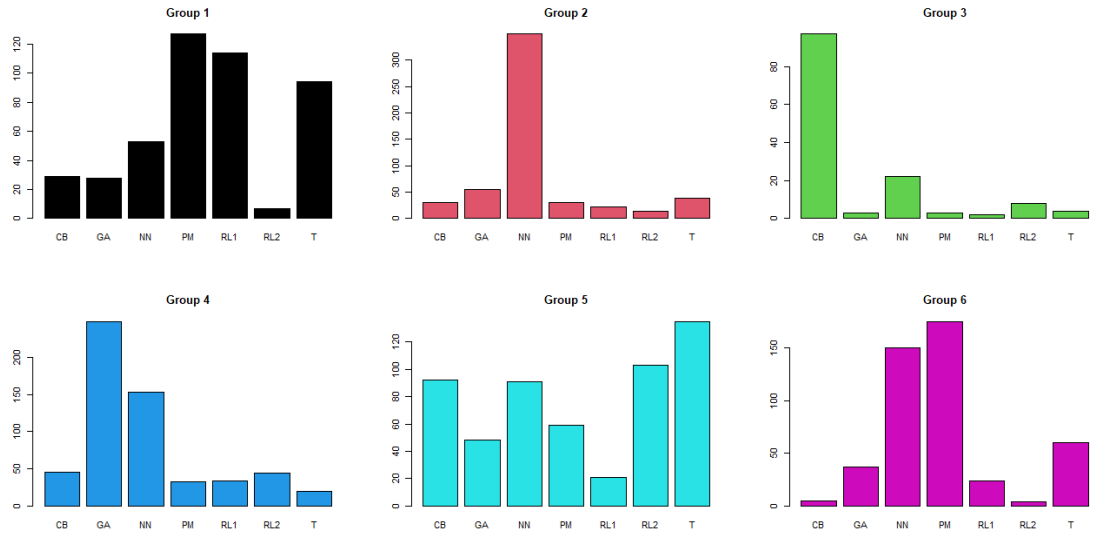


Figure 4.19 – Partitions with covariates taking into account classes in each group on Cora. Each group now contains a variety of categories that represent the hidden patterns discovered through the addition of covariates.

Next, to better understand the clustering results, more analysis on the obtained clusters are performed. We first plotted the latent positions learned by DeepLPM using PCA with a projection of the first two principal eigenvectors in Figure 4.20, highlighting nodes with degrees higher than 10. Those papers are more often cited by other papers and can be more representative. Based on the publications ID, we selected several articles with relatively large degree from each group and reported the information in Table 4.2. According to paper titles, it can first be seen that group $N1$ (red) focuses on dynamic or temporal learning algorithms using probabilistic methods or reinforcement learning; group $N2$ (green) then discusses different aspects of neural networks, such as self-organization, adjusting or rules; in group $N3$ (blue), the papers are largely based on the analysis and development of case studies; next, group $N4$ (cyan) contains articles on applications of genetic algorithms and neural networks; while in group $N5$ (purple), papers consist of rule learning and inductive methods; finally, group $N6$ (yellow) typically involves statistical and machine learning models.

Interestingly, when looking at Figure 4.20 from left to right, the content is changing from applied research to more theoretical learning, and then from bottom to top, the topic of the articles

is changing from case-based methods and reinforcement learning to genetic algorithms, and finally to neural networks and statistical models.

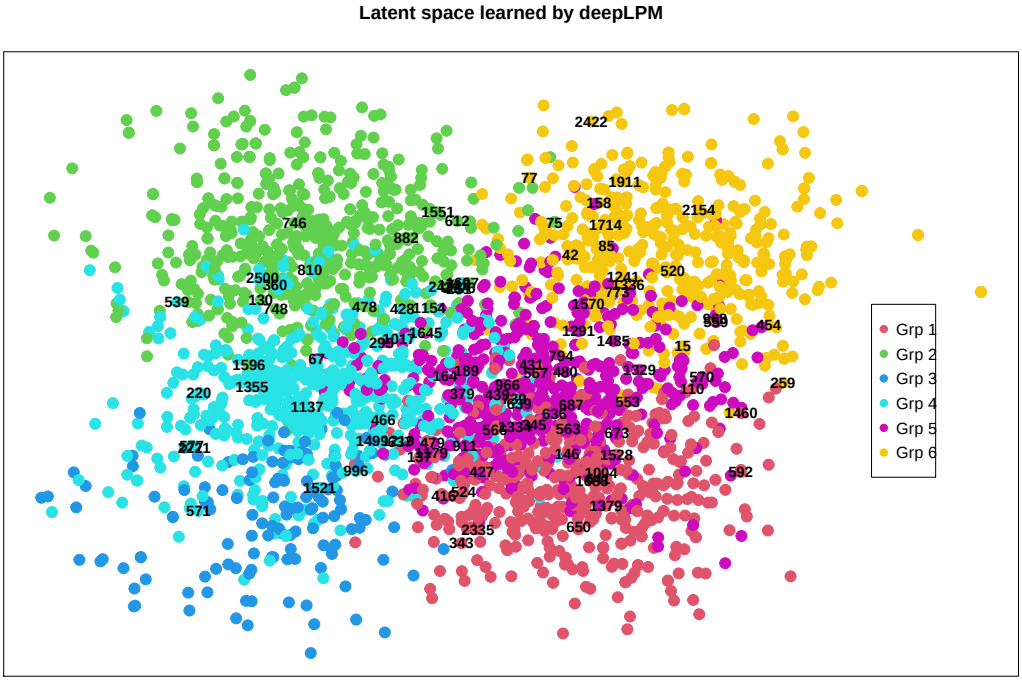


Figure 4.20 – Learned hidden space (PCA compression on the first two principal components), highlighting the nodes with degrees higher than 10.

Table 4.2 – Inspection of some nodes/documents having large degree

Groups	Node IDs	Paper titles	Degrees
Grp 1	#524	*Studies in machine learning using the game of checkers*	30
	#553	*Learning to act using real-time dynamic programming*	42
	#566	*Learning to predict by the methods of temporal differences*	78
	#567	*Integrated architectures for learning, planning, and reacting based on approximating dynamic programming*	32

	#673	*Cryptographic limitations on learning boolean formulae and finite automata*	21
Grp 2	#130	*Evolving networks*: *using the genetic algorithm with connection learning*	15
	#295	*Neuronlike adaptive elements that can solve difficult learning control problems*	32
	#746	*Self-organized formation of topologically correct feature maps*	33
	#748	*Self-organization and associative memory*	74
	#810	*Self-adjusting dynamic logic module*	14
	#882	*Proben1	A set of neural network benchmark problems and benchmarking rules*	14
Grp 3	#137	*Theory refinement combining analytical and empirical methods*	19
	#1499	*Inferential theory of learning*: *developing foundations for multistrategy learning*	12
Grp 4	#164	*Genetic algorithms in search, optimization and machine learning*	168
	#428	*Introduction to the theory of neural computation*	65
	#571	*A new learning algorithm for blind signal separation*	19
	#1355	*The structure-mapping engine*: *algorithm and examples*	23
	#1521	*Adaptive nonlinear PCA algorithms for blind source separation without prewhitening*	18
Grp 5	#345	*Learning logical relations from definitions*	31
	#379	*An empirical comparison of selection measures for decision-tree induction*	26
	#431	*Irrelevant features and the subset selection problem*	36
	#636	*Learning with many irrelevant features*	21
	#911	*Learning sequential decision rules using simulation models and competition*	22

Grp 6	#15	*Hidden Markov models in computational biology*: *applications to protein modeling*	19
	#42	*Markov chain Monte Carlo convergence diagnostics*: *a comparative review*	14
	#75	*Hierarchical mixtures of experts and the EM algorithm*	40
	#77	*A view of the EM algorithm that justifies incremental, sparse, and other variants*	16
	#454	*How to use expert advice*	23
	#794	*A survey of evolution strategies*	23

We close this section emphasizing once more that, in unsupervised problems, we cannot determine the number of clusters solely based on the number of the classes that are used in supervised tasks. Conversely, when selecting the number of clusters via model selection (that VAEs seem to perform intrinsically), we are able to discover interesting new similarities between the nodes of a graph.

4.7 Conclusion and perspectives

We introduced DeepLPM to perform node clustering on network data in an end-to-end manner with covariates (if available). By integrating the GCN encoder with the LPM-based decoder, we retain the interpretability of the statistical model while also enjoying the excellent performance of neural networks in representation learning. An original estimation procedure combined the explicit optimization via variational inference and the implicit optimization using stochastic gradient descent. Numerical experiments show that DeepLPM outperforms state-of-the-art methods and highlight its capabilities in terms of model selection. Real-world applications on a historical network and a scientific citation network were also proposed to illustrate the interest of the method for unsupervised analysis.

In this chapter, we treat edge features as a type of covariates used in the decoding phase. For future work, inspired by Bouveyron et al. [2018], we are interested in analyzing textual edges by incorporating topic modeling into the generative process.

CHAPTER 5

The graph embedded topic model

Most of existing graph neural networks (GNNs) developed for the prevalent text-rich networks typically treat texts as node attributes. This kind of approach unavoidably results in the loss of important semantic structures and restricts the representational power of GNNs. In this chapter, we introduce a document similarity-based graph convolutional network (DS-GCN) encoder to combine graph convolutional networks and embedded topic models for text-rich network representation. Then, a latent position-based decoder is used to reconstruct the graph while preserving its topology. Similarly, the document matrix is rebuilt using a decoder that takes both topic and word embeddings into account. By including a cluster membership variable, we thus develop an end-to-end clustering technique relying on a new deep probabilistic model called graph embedded topic model (GETM). Numerical experiments on three simulated scenarios emphasize the ability of GETM in fusing the graph topology structure and the document embeddings, and highlight its node clustering performance. Moreover, an application on the Cora-enrich citation network is conducted to demonstrate the effectiveness and interest of GETM in practice.

This Chapter is related with a submitted journal version, ***The graph embedded topic model****, Preprint HAL-03942487 (2023).*

5.1 Introduction 109
5.1.1 Organization of the chapter 110
5.2 The graph embedded topic model 110
5.2.1 Notations 110
5.2.2 Generative model 111
5.3 Inference model 113
5.3.1 Variational inference 113
5.3.2 Document similarity-based GCN. 114
5.3.3 Optimization 116
5.4 Numerical experiments 119
5.4.1 Simulation setup 120
5.4.2 Benchmark study 121
5.4.3 A more detailed example 122
5.4.4 Model selection 126
5.5 Application on real-world network 128
5.5.1 Model selection 128
5.5.2 Visualisation and analysis 130
5.6 Conclusion 134

5.1 Introduction

Heterogeneous mixed-type data is a common component of real-world networks. In a scientific article citation network, for example, textual information such as paper titles and abstracts are included in addition to paper-paper interactions as well as the category of each article. The way to incorporate these valuable information under the graph structure is crucial and would affect the quality of the network representation through latent embeddings. Currently available graph neural networks (see Section 2.3.2) for heterogeneous information networks typically treat these knowledge as node attributes, which inevitably results in the loss of node feature characteristics and restricts the representation ability on graph topological structure [Wang et al., 2020b].

This chapter focuses on the modeling and clustering of ubiquitous text-rich networks, where each node in the graphs is associated with a document that contains textual information about that node. Recently, numerous efforts have been made to combine graph embedding learning with text analysis techniques like word embedding and topic modeling (see Section 2.4). Even though these well-established techniques yield satisfactory results, they merely discover various ways to mix document and graph embeddings without modifying the structure of GNNs. To date, these joint convolutions are carried out using standard graph convolutional networks (GCNs). For instance, BiTe-GCN [Jin et al., 2021] and AS-GCN [Yu et al., 2021] directly apply GCN on the augmented bi-typed (document nodes with word nodes) and tri-typed (document, topic and word nodes) networks, while AM-GCN [Wang et al., 2020b] introduces a shared weight matrix of learnable parameters in the proposed common-GCN for feature and graph spaces without changing the GCN structure, as discussed in Section 2.4.

In this chapter, we propose a graph embedded topic model (GETM) to integrate graph embeddings, topic modeling and node clustering approach in an end-to-end manner. Then, we examine the ability of GCNs in fusing graph topological structure and node features, and further introduce a new document similarity-based GCN to better account for these two aspects and improve the performance of node clustering in networks.

The main contributions of the GETM that we propose here include:

— a document similarity-based GCN (DS-GCN) encoding approach is presented to address the information loss that occurs when merely considering documents as node attributes;

— a latent position-based decoder is employed to preserve the graph topology and to reconstruct the graph adjacency matrix more accurately;
— another embedded topic model(ETM)-based decoder is proposed to combine topic modeling and word embeddings for more efficient reconstruction of the document-term matrix;
— a joint optimization is carried out for both document embedding learning and graph topology learning based on the VAE architecture;
— an *end-to-end* node clustering procedure is performed by estimating the posterior probabilities for cluster memberships. Thus, the inference procedure can automatically assign each node to its group without using any additional algorithms.

5.1.1 Organization of the chapter

In Section 5.2, the generative model behind GETM is introduced firstly. Then, a variational inference strategy and an optimization algorithm are discussed in Section 5.3, as well as an introduction about the novel structure of DS-GCN. Numerical experiments are reported in Section 5.4, highlighting the main features of our methodology and validating its ability in exploiting both the graph topology and topic modeling for node clustering in simulated networks. An application on a real-world network Cora-enrich is presented in Section 5.5. Finally, Section 5.6 provides some concluding remarks and future work.

5.2 The graph embedded topic model

The two building blocks of our GETM are GCN and ETM. The reader is referred to the Sections 2.1.2 and 2.3.2 for a review of these models. In our proposed GETM, we combine the representations learned by a document similarity-based GCN and the document analysis capability of ETM to obtain a joint embedding that takes both the graph topology and document semantics into account, and to further perform an end-to-end clustering of the nodes.

5.2.1 Notations

In this chapter, each network is modeled as an undirected, unweighted, graph G with N nodes. We introduce an $N \times N$ adjacency matrix A to encode the network topology, where $A_{ij} = 1$ if there is a link between node i and node j, 0 otherwise. In addition, each node is associated with

a specific document. We introduce a corpus of N documents with a vocabulary that contains V unique terms. For clarity, we set the number of documents denoted by D in ETM to $D = N$ (the number of nodes) since each node is assumed to be associated with a single document. W is a document-term matrix where each vector W_i, $i \in \{1, \cdots, N\}$ encodes the document of node i containing a collection of M_i words. W_{iv}, where $v \in \{1, \cdots, V\}$; counts the number of times that the vocable v in the dictionary appears in the i-th document.

We aim at learning latent, joint, node/document embeddings Z in a lower dimension P. Then, using this learned embedding Z, our goal is to convert it into a graph embedding in dimension F and a document embedding in dimension T, which allow us to in turn reconstruct the graph adjacency and the document-term matrices, as well as to partition the nodes of the network into K clusters. We emphasize that GETM is capable of simultaneously performing node clustering and embedding construction.

5.2.2 Generative model

The generative process for GETM is now detailed. First, each node is assumed to be assigned to a cluster via a random variable c_i encoding its cluster membership

$$c_i \overset{\text{i.i.d}}{\sim} \mathcal{M}(1, \pi), \quad \text{with} \quad \pi \in [0,1]^K, \sum_{k=1}^{K} \pi_k = 1. \tag{5.1}$$

Then, conditionally to its cluster membership, a latent, joint embedding z_i is generated as

$$z_i | (c_{ik} = 1) \sim \mathcal{N}(\mu_k, \sigma_k^2 I_P), \quad \text{with} \quad \sigma_k^2 \in \mathbb{R}^{+*} \text{ and } \mu_k \in \mathbb{R}^P, \tag{5.2}$$

independently for each node $i = \{1, \cdots, N\}$.

Based on this joint embedding, a graph topology embedding is generated as

$$\eta_i = h_\iota^{(G)}(z_i), \tag{5.3}$$

where $h_\iota^{(G)}(\cdot)$ is a neural network with parameters ι to map the P-dimensional vector z_i into dimension F.

Next, the probability of a connection between nodes i and j is modeled by a distance function between two graph topology embeddings

$$A_{ij} = 1 | \eta_i, \eta_j \sim \mathcal{B}(f_\tau(\eta_i, \eta_j)), \tag{5.4}$$

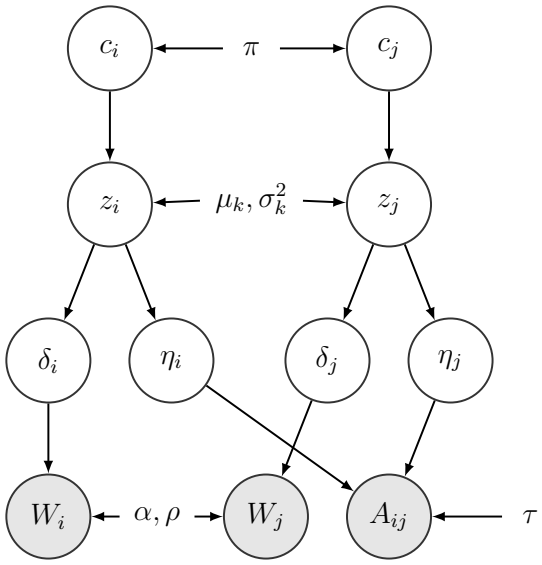


Figure 5.1 – Graphical representation of GETM (variational parameters are not included).

with

$$f_\tau(\eta_i, \eta_j) = \sigma(\tau - ||\eta_i - \eta_j||^2), \tag{5.5}$$

where $\sigma(\cdot)$ denotes a logistic sigmoid function. Here $f_\tau(\cdot)$ can be seen as the *graph decoder* parametrized by τ, which encodes the prior probability to connect.

Similarly, a document embedding is assumed to be generated based on z_i

$$\delta_i = h_\nu^{(T)}(z_i), \tag{5.6}$$

where $h_\nu^{(T)}(\cdot)$ is a neural network with parameter ν to map the P-dimensional vector z_i into dimension T. The topic proportions of each document i are then obtained as

$$\theta_i = \text{softmax}(\delta_i). \tag{5.7}$$

Finally, each document is assumed to be drawn from

$$W_i|\theta_i \sim \mathcal{M}(M_i; \theta_i^\top \beta), \quad \text{with} \quad \beta = \text{softmax}(\alpha^\top \rho). \tag{5.8}$$

As in ETM, α is a topic embedding representing topics in an L-dimensional space and ρ is a $L \times V$ word embedding matrix obtained typically via word2vec [Mikolov et al., 2013a] or any other word embedding approach. Moreover, the product $\theta^\top \beta$ can be viewed as a *document decoder* to map the topic and word embeddings into a reconstructed document-term matrix. A graphical representation of the generative model described so far can be seen in Figure 5.1.

5.3 Inference model

In this section, we detail the developed variational inference and the optimization algorithm, and introduce the proposed document similarity-based graph convolutional network (DS-GCN).

5.3.1 Variational inference

Before getting into the details of the inference, we first denote by $\Theta = \{\pi, (\mu_k, \sigma_k^2)_k, \iota, \nu, \tau, \alpha, \rho\}$ the set of the model parameters introduced so far. A natural procedure would consist in maximizing the integrated log-likelihood of the observed data A and W with respect to Θ

$$\log p(A, W|\Theta) = \log \int_Z \sum_C p(A, W, Z, C|\Theta) dZ. \tag{5.9}$$

Unfortunately, Eq. (5.9) is not tractable and we rely on a variational approach to approximate it

$$\log p(A, W|\Theta) = \mathcal{L}(q(Z, C); \Theta) + \mathcal{KL}(q(Z, C)||p(Z, C|A, \Theta)), \tag{5.10}$$

where $\mathcal{KL}$ denotes the Kullback-Leibler divergence between the true and approximate posterior distributions of (Z, C) given the data and model parameters. Then, in order to deal with a tractable family of distributions, $q(Z, C)$ is assumed to fully factorize (*mean-field* assumption)

$$q(Z, C) = q(Z)q(C) = \prod_{i=1}^{N} q(z_i)q(c_i). \tag{5.11}$$

Moreover, to benefit from the representational learning capabilities of graph neural networks, our inference strategy relies on a two-layer DS-GCN to encode the graph adjacency matrix and the document-term matrix into a joint embedding

$$q(z_i|A, W) = \mathcal{N}(z_i; \mu_i, \sigma_i^2 I_P), \tag{5.12}$$

where $\mu_i : \mathbb{R}^{N\times N} \mapsto \mathbb{R}^{N\times P}$ (respectively $\sigma_i^2 : \mathbb{R}^{N\times N} \mapsto \mathbb{R}^{+*}$) is the function mapping the normalized adjacency matrix $\tilde{A} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ into the matrix of variational means (and standard deviations), parametrized by a two-layer DS-GCN defined as g_ϕ:

$$\begin{aligned} H^{(1)} &= \sigma(\tilde{A} \odot (\tilde{W}\tilde{W}^\top)\hat{W}^{(0)}), \\ H^{(2)} &= \tilde{A} \odot (H^{(1)}{H^{(1)}}^\top)\hat{W}^{(1)}, \end{aligned} \tag{5.13}$$

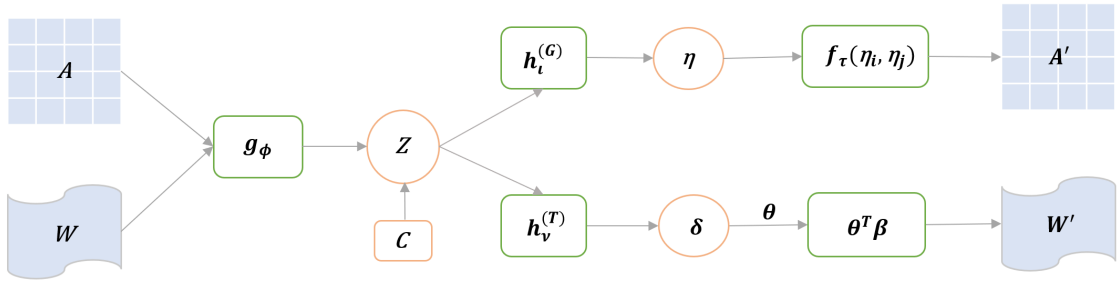


Figure 5.2 – Model architecture of GTEM.

where $\tilde{W}$ is the normalized document-term matrix, obtained via $\tilde{W} = \frac{W}{|W|}$. $\hat{W}^{(\cdot)}$ are learnable weight matrices and $\sigma(\cdot)$ denotes a ReLU activation function. Here g_ϕ can be seen as the *encoder* that transforms two input matrices into latent, joint embeddings. The motivation and details of our proposal for such a structure are described in the following Section 5.3.2.

Finally, a standard assumption is made for variational cluster probabilities

$$q(C) = \prod_{i=1}^{N} \mathcal{M}(c_i; 1, \gamma_i), \quad \text{with} \quad \sum_{k=1}^{K} \gamma_{ik} = 1, \tag{5.14}$$

where γ_{ik} represents the variational probability that node i is in cluster k.

Model architecture. From a deep learning view, the model architecture can be seen in Figure 5.2. GETM takes the graph adjacency matrix A and the document-term matrix W as model inputs. Through the *DS-GCN encoder* g_ϕ, we obtain a combined embedding Z containing information about both graph topology and latent topics. Then, a latent position-based *graph decoder* f_τ is developed to map the joint embedding into a reconstructed graph matrix, and another *document decoder* $\theta^\top \beta$ is used to rebuild the document-term matrix. Additionally, by including latent cluster variables C, we are also able to explicitly optimize and eventually output a matrix $\hat{\gamma}$ that represents the clustering probabilities, and, as a result, achieve end-to-end clustering.

5.3.2 Document similarity-based GCN.

In this section, we detail the motivation for proposing the structure of DS-GCN. Despite having shown proficiency in representation learning tasks, attributed GCNs will inevitably neglect crucial semantic information in text-rich networks when simply considering texts as node attributes.

An introductory example. In order to illustrate the rationale behind our approach, we begin with a straightforward introductory example in which the network has three nodes, each node representing a document. In addition, suppose that the data has five distinct words in the vocabulary, divided into two topics. Such data set will be thus characterized by the following matrices

$$\hat{A} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad W = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix},$$

where $\hat{A}$ shows two connection structures: the first two nodes strongly interact and the third node is not connected. Moreover, there are two types of texts in W : the first and the third node use similar words while words used by the second node completely differs. Therefore, when considering both the link relationships and the text information, it is more natural to conclude that there are three distinct clusters in this instance. Notice if we would have used the convolution operation as in GCN, we would obtain

$$\tilde{A}H^{(0)} = \begin{pmatrix} 0.5 & 0.5 & 0 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0 & 0.5 & 0.5 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}, \qquad \text{with} \quad \hat{D} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

where $\tilde{A} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$, $H^{(0)} = W$ at the first layer. This operation ignores the semantic content of texts and only retains the network structure. As a result, it is unable to distinguish between the first and second nodes into two groups. To overcome this restriction, we develop a new GNN structure that takes into account the document similarities, as described below.

DS-GCN. In contrast to standard GCNs, we here assume a new GNN structure named DS-GCN, with the following per-layer propagation rule

$$H^{(l+1)} = \sigma(\tilde{A} \odot (H^{(l)}{H^{(l)}}^{\top})\hat{W}^{(l)}), \tag{5.15}$$

where $\tilde{A} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$, $\sigma(\cdot)$ denotes the ReLU activation, and $\odot$ is an element-wise multiplication. $H^{(l)}$ encodes the learned representation in layer l, with $H^{(0)} = \tilde{W}$ the normalized document-term matrix, and $\hat{W}^{(l)}$ is the learnable weight matrix of the l-th layer. Returning to the

illustration example, we have in the top layer

$$\tilde{A} \odot (\tilde{W}\tilde{W}^\top) = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

where the dot product $\tilde{W}\tilde{W}^\top$ can be viewed as the similarity between documents, typically used in NLP. The obtained product represents features containing both node connections and document similarities information. As a result, GETM is able to divide the data into three clusters, as expected. More experiments are conducted in the next section to show the validity of DS-GCN.

5.3.3 Optimization

In this part, we focus on maximizing the evidence lower bound (ELBO)

$$\mathcal{L}(q(Z,C);\Theta) = \int_Z \sum_C q(Z,C) \log \frac{p(A,W,Z,C|\Theta)dZ}{q(Z,C)} \tag{5.16}$$

with respect to the model parameters Θ and the variational parameters. Thanks to Equations (5.12)-(5.13)-(5.14), the evidence lower bound (ELBO) denoted by $\mathcal{L}$ can be further developed as

$$\begin{aligned}
\mathcal{L} &= \int_Z \sum_C q(Z,C) \log \frac{p(A,W|Z,\iota,\nu,\tau,\alpha,\rho)p(Z|C,\mu_k,\sigma_k^2)p(C|\pi)dZ}{q(Z,C)} \\
&= \sum_{i\neq j} \mathbb{E}_{q(Z|A,W)} \left[\log p(A|\iota,\tau,\eta_i,\eta_j)\right] + \sum_{i=1}^{N}\sum_{m=1}^{M_i} \mathbb{E}_{q(Z|A,W)} \left[\log p(W_{im}|\nu,\delta_i,\alpha,\rho)\right] \\
&\quad + \mathbb{E}\left[\log \frac{p(Z|C,\mu_k,\sigma_k^2)}{q(Z|A,W)}\right] + \mathbb{E}\left[\log \frac{p(C|\pi)}{q(C)}\right] \\
&= \sum_{i\neq j} \mathbb{E}_{q(Z|A,W)} \left[\log p(A|\iota,\tau,\eta_i,\eta_j)\right] + \sum_{i=1}^{N}\sum_{m=1}^{M_i} \mathbb{E}_{q(Z|A,W)} \left[\log p(W_{im}|\nu,\delta_i,\alpha,\rho)\right] \\
&\quad - \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \mathcal{KL}(\mathcal{N}(\mu_i,\sigma_i^2 I_P)||\mathcal{N}(\mu_k,\sigma_k^2 I_P)) + \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \log(\frac{\pi_k}{\gamma_{ik}}).
\end{aligned} \tag{5.17}$$

The first term of the ELBO calculates the difference between the reconstructed and original graph adjacency matrices. The second term accounts for the reconstruction error between the document-term matrices of the input and output. The third term considers the KL divergence (denoted by $\mathcal{KL}(\cdot)$) between the approximate posterior distribution of node i, obtained with the encoder, and

the prior distribution of component k. Finally, the last term takes into consideration the clustering probabilities.

On the one hand, an *explicit* optimization of the ELBO with respect to the parameters $\gamma_{ik}, \pi_k, \mu_k$ and σ_k can be performed via Proposition 3.

Proposition 3. *The following variational updates can be obtained*:

$$\hat{\gamma}_{ik} = \frac{\pi_k \mathrm{e}^{-\mathcal{KL}_{ik}}}{\sum_{l=1}^{K} \pi_l \mathrm{e}^{-\mathcal{KL}_{il}}}, \tag{5.18}$$

where $\mathcal{KL}_{ik} = \frac{1}{2}\left\{\log \frac{(\sigma_k^2)^P}{(\tilde{\sigma}_\phi^2(\overline{A})_i)^P} - P + \frac{\tilde{\sigma}_\phi^2(\overline{A})_i}{\sigma_k^2} + \frac{1}{\sigma_k^2}||\mu_k - \tilde{\mu}_\phi(\overline{A})_i||^2\right\}$.

Then

$$\hat{\pi}_k = \sum_{i=1}^{N} \gamma_{ik}/N, \tag{5.19}$$

$$\hat{\mu}_k = \sum_{i=1}^{N} \tilde{\mu}_\phi(\tilde{A})_i \gamma_{ik} / \sum_{i=1}^{N} \gamma_{ik}, \tag{5.20}$$

and

$$\hat{\sigma}_k^2 = \frac{\sum_{i=1}^{N} \gamma_{ik}(\sigma_\phi^2(\tilde{A})_i + ||\mu_k - \tilde{\mu}_\phi(\tilde{A})_i||^2)}{P \sum_{i=1}^{N} \gamma_{ik}}. \tag{5.21}$$

Proof.

Detailed derivations are given as follows. Under the equality constraint $\sum_{k=1}^{K} \gamma_{ik} = 1, \forall k$, *we use the method of Lagrange multipliers. Firstly, we introduce a Lagrange multiplier* λ_i

$$\tilde{\mathcal{L}} = \mathcal{L} - \sum_{i=1}^{N} \lambda_i \left(\sum_{k=1}^{K} \gamma_{ik} - 1 \right),$$

then, we derive $\tilde{\mathcal{L}}$ *according to* γ_{ik}

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \gamma_{ik}} = \log \pi_k - \log \gamma_{ik} - \frac{\gamma_{ik}}{\gamma_{ik}} - \mathcal{KL}_{ik} - \lambda_i = 0,$$

thus, we have

$$\begin{aligned} \log \gamma_{ik} &= \log \pi_k - 1 - \mathcal{KL}_{ik} - \lambda_i, \\ \gamma_{ik} &= \mathrm{e}^{\{\log \pi_k - 1 - \mathcal{KL}_{ik} - \lambda_i\}} = \frac{\mathrm{e}^{\{\log \pi_k - \mathcal{KL}_{ik}\}}}{\mathrm{e}^{\{1+\lambda_i\}}}. \end{aligned} \tag{5.22}$$

By using the constraint on $\sum_{k=1}^{K} \gamma_{ik}$*, we can get*

$$\sum_{k=1}^{K} \gamma_{ik} = \frac{\sum_{k=1}^{K} \mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}}}{\mathrm{e}^{\{1+\lambda_i\}}} = 1$$
$$\log \sum_{k=1}^{K} \mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}} = \log \mathrm{e}^{\{1+\lambda_i\}}$$
$$\lambda_i = \log \sum_{k=1}^{K} \mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}} - 1.$$

After putting the value of λ_i *into Eq. 5.22*

$$\gamma_{ik} = \frac{\mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}}}{\mathrm{e}^{\{1+\log \sum_{k=1}^{K} \mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}} - 1\}}} = \frac{\mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}}}{\sum_{k=1}^{K} \mathrm{e}^{\{\log \pi_k - \mathcal{K}\mathcal{L}_{ik}\}}}.$$

Finally, we obtain

$$\hat{\gamma}_{ik} = \frac{\pi_k \mathrm{e}^{-\mathcal{K}\mathcal{L}_{ik}}}{\sum_{l=1}^{K} \pi_l \mathrm{e}^{-\mathcal{K}\mathcal{L}_{il}}}. \tag{5.23}$$

Similarly, since $\sum_{k=1}^{K} \pi_k = 1, \forall k$*, we introduce another Lagrange multiplier, let us say* ζ

$$\tilde{\mathcal{L}} = \mathcal{L} - \zeta \left(\sum_{k=1}^{K} \pi_k - 1 \right),$$

then, we derive $\tilde{\mathcal{L}}$ *according to* π_k

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \pi_k} = \sum_{i=1}^{N} \frac{\gamma_{ik}}{\pi_k} - \zeta = 0,$$

next, we use the equality constraint to find the value of ζ

$$\sum_{k=1}^{K} \sum_{i=1}^{N} \gamma_{ik} = \sum_{k=1}^{K} \pi_k \zeta$$
$$\zeta = N,$$

and finally, we have

$$\hat{\pi}_k = \sum_{i=1}^{N} \gamma_{ik} / N. \tag{5.24}$$

Lastly, we need to calculate the derivatives for μ_k and σ_k^2. We start by deriving $\tilde{\mathcal{L}}$ according to μ_k

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \mu_k} = -\frac{1}{2}\sum_{i=1}^{N}\gamma_{ik}\{\frac{1}{\sigma_k^2}(2\mu_k - 2\tilde{\mu}_\phi(\tilde{A})_i\} = 0,$$

then, we obtain

$$\begin{aligned} \mu_k \sum_{i=1}^{N}\gamma_{ik} &= \sum_{i=1}^{N}\tilde{\mu}_\phi(\tilde{A})_i\gamma_{ik}, \\ \hat{\mu}_k &= \frac{\sum_{i=1}^{N}\tilde{\mu}_\phi(\tilde{A})_i\gamma_{ik}}{\sum_{i=1}^{N}\gamma_{ik}}, \end{aligned} \tag{5.25}$$

and finally for σ_k^2, we have

$$\begin{aligned} \frac{\partial \tilde{\mathcal{L}}}{\partial \sigma_k^2} = -\frac{1}{2}\sum_{i=1}^{N}\gamma_{ik}\{\frac{P}{\sigma_k^2} - \frac{1}{\sigma_k^4}(\sigma_\phi^2(\tilde{A})_i + ||\mu_k - \tilde{\mu}_\phi(\tilde{A})_i||^2)\} &= 0 \\ P\sum_{i=1}^{N}\frac{\gamma_{ik}}{\sigma_k^2} &= \sum_{i=1}^{N}\frac{\gamma_{ik}}{\sigma_k^4}(\sigma_\phi^2(\tilde{A})_i + ||\mu_k - \tilde{\mu}_\phi(\tilde{A})_i||^2) \\ P\sum_{i=1}^{N}\gamma_{ik}\sigma_k^2 &= \sum_{i=1}^{N}\gamma_{ik}(\sigma_\phi^2(\tilde{A})_i + ||\mu_k - \tilde{\mu}_\phi(\tilde{A})_i||^2) \\ \hat{\sigma}_k^2 &= \frac{\sum_{i=1}^{N}\gamma_{ik}(\sigma_\phi^2(\tilde{A})_i + ||\mu_k - \tilde{\mu}_\phi(\tilde{A})_i||^2)}{P\sum_{i=1}^{N}\gamma_{ik}}. \end{aligned} \tag{5.26}$$

□

On the other hand, the *implicit* optimization of the encoder parameter ϕ, two neural networks parameters ι and ν, graph decoder parameter τ as well as document decoder parameters α and ρ, is performed via stochastic gradient descent. In this work, the implicit optimization is implemented using the Adam optimizer [Kingma and Ba, 2014]. We also point out that during the estimation, a reparameterization trick as in Kingma and Welling [2014a] is used for the terms $\mathbb{E}_{q(Z|A,W)}[\log p(A|\iota,\tau,\eta_i,\eta_j)]$ and $\mathbb{E}_{q(Z|A,W)}[\log p(W_{im}|\nu,\delta_i,\alpha,\rho)]$.

5.4 Numerical experiments

This section aims at testing the effectiveness of GETM, including a DS-GCN encoder, a latent position-based graph decoder and a ETM-based document decoder, on three types of synthetic

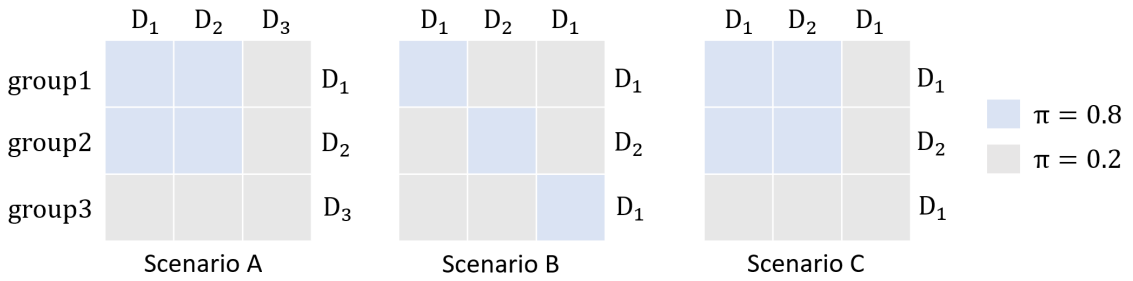


Figure 5.3 – Three scenarios to simulate synthetic networks.

networks, and to demonstrate the validity of the estimation algorithm proposed in the previous section.

5.4.1 Simulation setup

We first generate three different types of synthetic networks, each of which having three groups of nodes. The connections between nodes are obtained by a stochastic block model [SBM, Nowicki and Snijders, 2001]. Each node in the network is then associated with a document, where each word is picked at random from three articles from BBC news, denoted by D_1, D_2 and D_3, respectively. The first document discusses the birth of Princess Charlotte. The second text is about black holes in astrophysics. The last article focuses on UK politics. Three scenarios are described in detail as follows, with an illustration provided in Figure 5.3.

— Scenario A simulates a graph according to SBM, where edges between two nodes are drawn from independent Bernoulli distributions

$$A_{ij}|(Z_{ik}Z_{jl} = 1) \sim \mathcal{B}(\Pi_{kl}),$$

with Z_{ik} being 1 if node i is in cluster k, 0 otherwise. The connection probabilities are defined as

$$\Pi = \begin{pmatrix} 0.8 & 0.8 & 0.2 \\ 0.8 & 0.8 & 0.2 \\ 0.2 & 0.2 & 0.2 \end{pmatrix},$$

where each entry Π_{kl} denotes the probability that a node in group k and a node in group l connect. Thus, the third group has low inter and intra connection probabilities and differs from the other two groups, which are more likely to connect. Additionally, we assign a

document to each node and the words used in the three document groups are randomly extracted from text D_1, D_2 and D_3, respectively (see Figure 5.3). If we solely analyze the network topology structure, the first two groups are likely to be clustered together. However, if we look at the text information that each node in this network is concerned with, the actual number of clusters is three.

— In Scenario B, networks are also simulated according to SBM, with connection probabilities given by

$$\Pi = \begin{pmatrix} 0.8 & 0.2 & 0.2 \\ 0.2 & 0.8 & 0.2 \\ 0.2 & 0.2 & 0.8 \end{pmatrix},$$

corresponding to a clear community structure. It is simple to detect three clusters by just considering the network topology. However, group 1 and group 3 adopt words from D_1, whereas words in group 2 are extracted from D_2 (see Figure 5.3). Therefore, there are two clusters if we only take the document information into account.

— Scenario C has the same connection probabilities as in Scenario A and the texts assignments are the same as in Scenario B. In this situation, focusing solely on the graph topology or the textual data would lead to the discovery of two clusters, whereas the actual group numbers is three when the two types of information are considered simultaneously.

5.4.2 Benchmark study

We now aim at benchmarking the clustering performance of GETM with the following competitors in the three simulated scenarios.

— ETM [Dieng et al., 2020] is a document generative model that combines traditional topic models with word embeddings and is intended only for textual data.

— SBM [Nowicki and Snijders, 2001] is a widely used generative model in network analysis for clustering of nodes and is designed for graph data only.

— VGAE [Kipf and Welling, 2016] encodes the adjacency matrix and the node/document feature matrix by a GCN, and adopts an inner-product decoder for graph reconstruction.

— AM-GCN [Wang et al., 2020b] is a GCN-based method which performs graph convolution both accounting for the network topology and the node features space.

Table 5.1 – Experimental clustering results on 3 simulated scenarios.

	Scenario A	Scenario B	Scenario C
ETM	**1.000**±0.00	0.552±0.02	0.540±0.02
SBM	0.630±0.05	**1.000**±0.00	0.608±0.05
VGAE	0.459±0.00	0.773±0.07	0.460±0.00
AM-GCN	**1.000**±0.00	0.892±0.03	0.961±0.02
GETM	**0.990**±0.01	**1.000**±0.00	**0.998**±0.00

For each scenario, we randomly generated 15 networks with 900 nodes, 3 clusters, and calculated the averaged adjusted rand index [ARI, Hubert and Arabie, 1985] for node clustering comparisons. The results are reported in Table 5.1.

As can be observed, ETM showed excellent results in Scenario A due to the fact that each group is associated with a distinct topic. However, in Scenario B and C, only two types of documents are considered, ETM has a poor ARI since it cannot exploit the connectivity between nodes/documents. SBM only achieved great results in Scenario B, which contains three groups in the graph topological structure. However, SBM failed to detect three clusters in Scenario A and C since it cannot exploit the textual interaction. VGAE treats text data as node attributes to perform clustering. Due to the constraints in fusing the graph topology and the word semantics in GCN, VGAE shows the worst performance in all situations. AM-GCN constructs a k-nearest neighbor graph based on the node feature matrix to capture the underlying document semantics. As a result, AM-GCN was able to identify three clusters in different scenarios. Nevertheless, its performance in Scenario B was unsatisfactory in comparison to Scenario A and C. Finally, GETM consistently displayed strong clustering performance with high ARI values in all situations, which demonstrates its capability in both representation learning and node clustering.

5.4.3 A more detailed example

We now focus on Scenario C, which is the primary emphasis of this work. In the initial configuration, both the graph topology structure and topic subjects are divided into two categories, making it difficult to find the actual number of clusters.

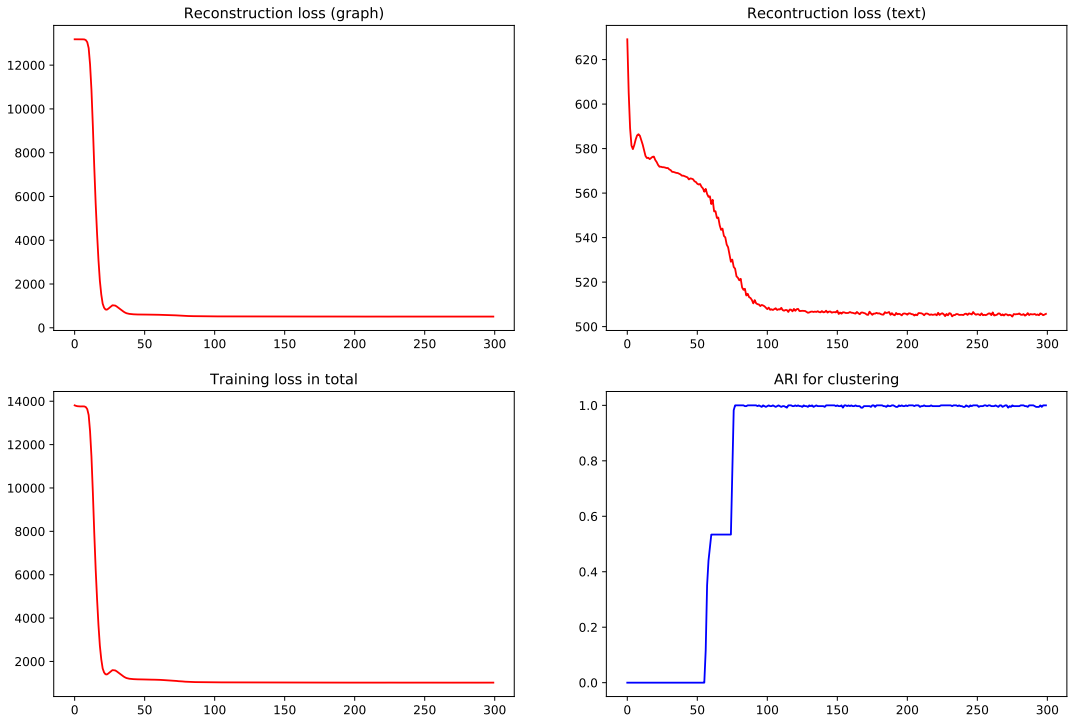


Figure 5.4 – The training loss for graph and texts reconstruction, the overall loss and the evolution of ARI during training.

We first run GETM on a synthetic network with 900 nodes, generated according to Scenario C, and then visualize the learned joint embeddings, graph embeddings and document embeddings, respectively. After training for 600 epochs, we plot the reconstruction loss for graph and text, the total loss (negative ELBO), and display the evolution of the ARI during training (Figure 5.4). All types of loss have converged and the ARI value (equal to 1.0) highlights the clustering capability of GETM as well.

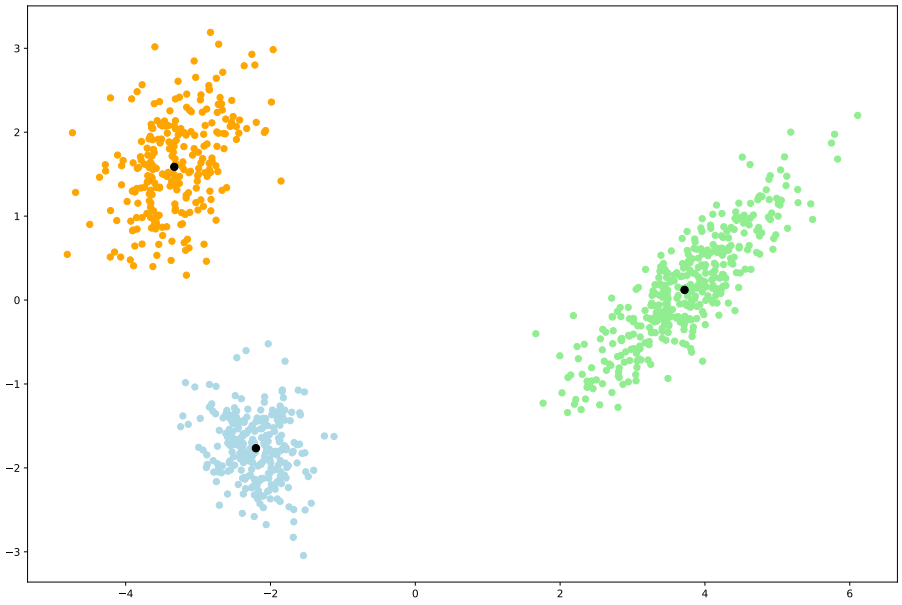


Figure 5.5 – PCA visualisation of the joint embedding Z $(P = 128)$ in Scenario C. Each cluster is represented by a distinct color. Three black points are the estimated cluster centers μ_k.

Then, we visualize the latent embeddings learned by GETM. First, Figure 5.5 displays the combined embeddings Z using PCA. Three groups with distinct colors can be distinguished by GETM, which demonstrates the ability of DS-GCN to exploit the network topology and the textual information. Then, regarding the graph topological embeddings η in Figure 5.6, two groups with different topologies are well preserved. As we can see, there are two clusters with a high probability of intra connections (positions are relatively closed) and one cluster is separated, which is coherent with our initial configuration. In addition, two latent topics are successfully detected based on the document embeddings θ, as shown in Figure 5.7. One document cluster is far away, whereas the other two clusters are very close, recovering the simulation setup in Scenario C.

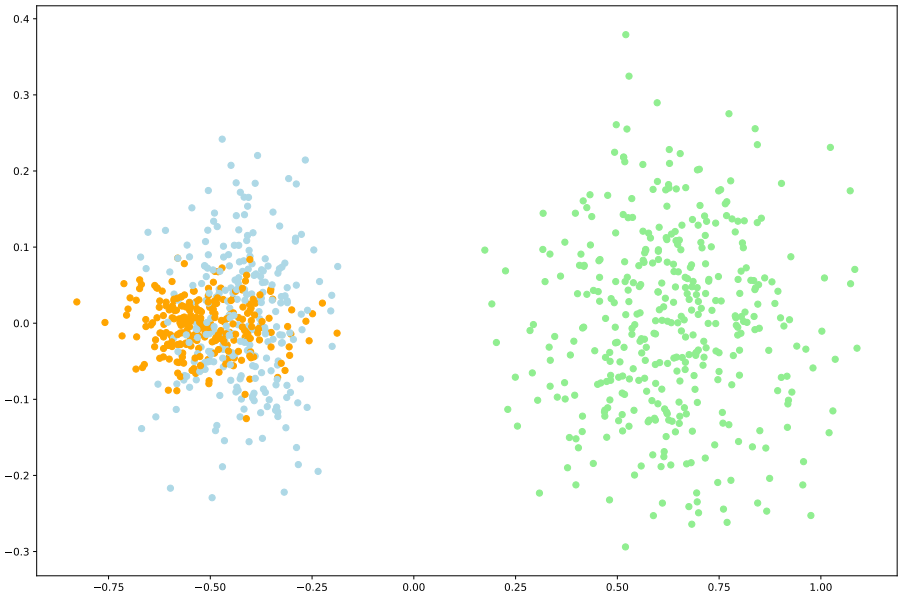


Figure 5.6 – PCA visualisation of η $(F = 128)$ in Scenario C. The green group is far away, whereas the orange and blue groups are relatively close, showing two different topologies.

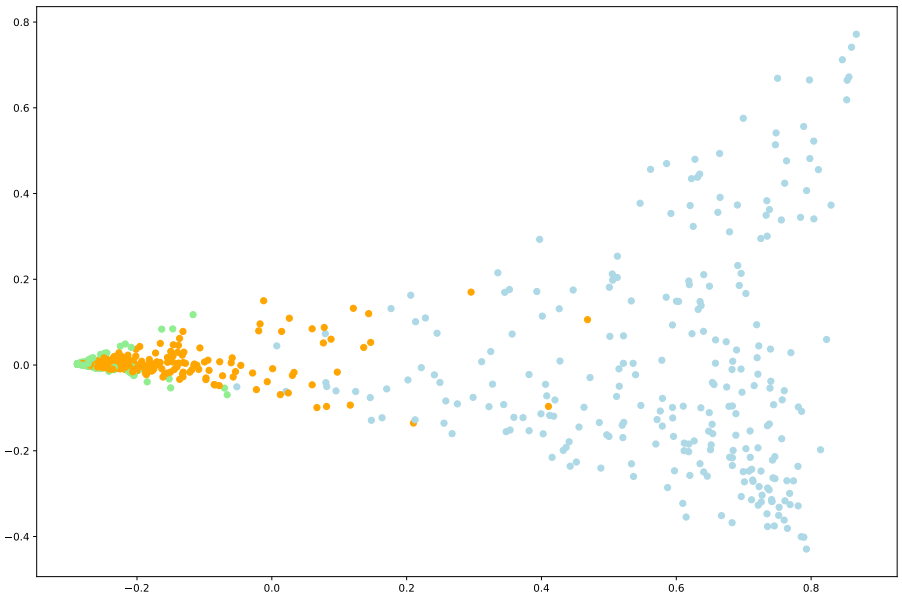


Figure 5.7 – PCA visualisation of θ $(T = 16)$ in Scenario C. The blue group is far away, whereas the green and orange groups are relatively close, showing two different topics.

Moreover, we also illustrate the top-10 words selected from two topics in Table 5.2. As we can see, words associated with the first topic are related to the birth of Princess Charlotte, and vocables

Table 5.2 – Top-10 words obtained from two topics.

Topic 1	"princess"	"charlotte"	"birth"	"queen"	"duchess"
	"duke"	"cambridge"	"granddaughter"	"great"	"palace"
Topic 2	"hole"	"black"	"see"	"gravity"	"light"
	"one"	"event"	"horizon"	"around"	"disc"

in the second topic are about black holes in astrophysics, which recovers the simulation setup for texts.

To conclude, all previous results indicate that GETM is capable of learning representations and performing node clustering in heterogeneous information networks.

5.4.4 Model selection

A key element of an unsupervised learning technique such as GETM is to be able to automatically determine the number of clusters (K). We highlight here the ability of our methodology to auto-penalize the ELBO for selecting the number of groups appropriately, which is made possible by the self-regularization ability of variational auto-encoders, also reported in Kingma et al. [2016]; Dai et al. [2017].

Number of clusters. Letting the number of clusters vary from 2 to 7, Figure 5.8 illustrates how the training loss (negative ELBO) can be used to estimate the number of clusters. In this experiment, for each value of the number of clusters, we generated five synthetic networks from Scenario C and trained GETM with the latent, joint embedding dimension $P = 16$, the graph embedding dimension $F = 16$ as well, and the dimension of document embedding equal to the number of topics $T = 2$. It can be seen that when $K = 3$, the training loss (negative ELBO) is minimal, thus recovering the actual value of K for the simulation setting.

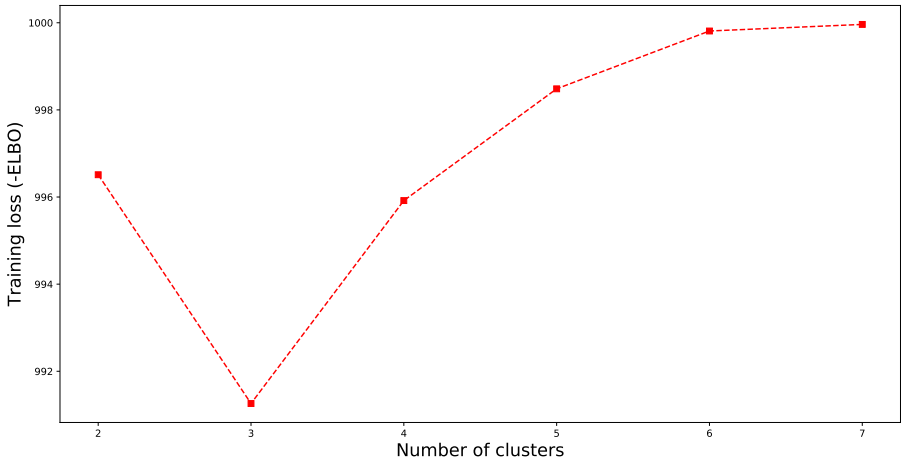


Figure 5.8 – Averaged training loss (negative ELBO) with different number of clusters on 30 synthetic networks in Scenario C. GETM was able to estimate $K = 3$ by displaying a clear minimum of the negative ELBO.

Dimension for latent topics. We further evaluated the model selection ability with different number of latent topics. In this experiment, we generated five synthetic networks from Scenario C and trained GETM with the latent, joint embedding dimension P of 128, the graph embedding dimension F of 128, and the document embedding dimension T of $\{2, 16, 128\}$, respectively. In Table 5.3, we examine the training loss (negative ELBO) for various number of clusters $K \in \{1, \cdots, 6\}$. The capacity of GETM to choose the proper cluster numbers is highlighted by the fact that the minimal training loss (negative ELBO) is discovered when $K = 3$, with varying dimensions of latent topics.

Table 5.3 – Averaged training loss (negative ELBO) with different number of clusters and latent topic dimension in Scenario C.

	K=2	K=3	K=4	K=5	K=6
T=2	1048.11	**1028.74**	1038.37	1037.69	1037.17
T=16	1040.65	**1026.58**	1027.36	1027.66	1028.89
T=128	1027.88	**1026.60**	1028.44	1027.27	1028.14

5.5 Application on real-world network

In this section, GETM is fitted on a text-rich citation network Cora-enrich [*] as an illustration of its practical use. The original Cora [†] dataset contains 2,708 scientific publications classified in seven categories: *case based*, *genetic algorithms*, *neural networks*, *probabilistic methods*, *reinforcement learning*, *rule learning* and *theory*. It consists of 5,429 links and 1,433 vocabulary. Each publication is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from a dictionary. Recently, Ganguly and Pudi [2017] enriched the text information by collecting the titles, abstracts and all sentences from a paper containing citations, which leads to 25,955 vocables in Cora-enrich network. This dataset shares the same papers, categories and citation relationships with Cora.

Most related works [Pan et al., 2018; Mehta et al., 2019; Jin et al., 2021; Yu et al., 2021] assume that the number of clusters is equal to the number of classes used in supervised node classification tasks, whereas we argue that the class labels (thematic categories) might not be in a one-to-one relation with the detected clusters in unsupervised node clustering. Instead, an appropriate cluster number should be obtained through model selection.

5.5.1 Model selection

As the model selection ability of GETM is demonstrated in Section 5.4.4, GETM is also fitted to the Cora-enrich network for different numbers of clusters, ranging between 4 and 10, with fixed dimensions ($P, F, T = 128$) for three latent spaces. The evolution of the training loss (negative ELBO) with various cluster numbers is shown in Figure 5.9. The reported result is the lowest value obtained after running GETM 10 times for each cluster number. Finally, the estimated number of clusters is $K = 6$ by displaying a clear minimum of the negative ELBO.

Confusion matrix. We also plot the confusion matrix between six estimated cluster partitions and seven thematic categories to investigate the fusion or dispersion between multiple classes, as shown in Figure 5.10. As we can see, the majority of publications on reinforcement learning (T3), rule learning (T5) and genetic algorithms (T6) are extracted into clusters $C3$, $C5$ and $C1$, respectively. The two clusters $C2$ and $C6$ constitute the primary division between articles about neural

*. http://zhang18f.myweb.cs.uwindsor.ca/datasets/
†. https://relational.fit.cvut.cz/dataset/CORA

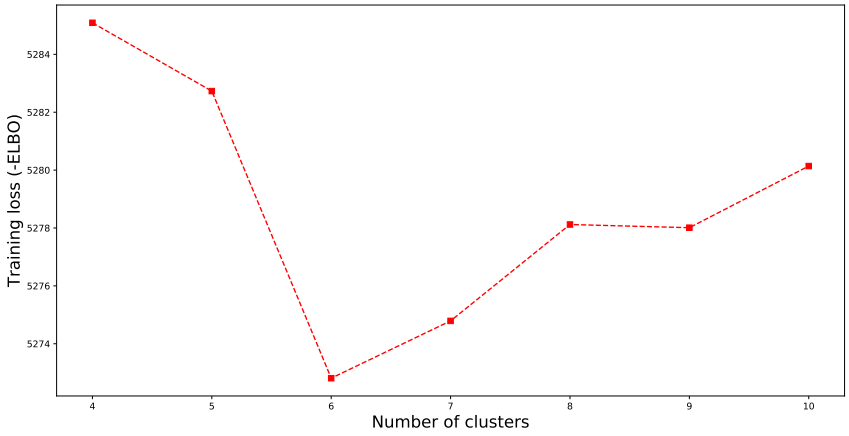


Figure 5.9 – Training loss (negative ELBO) with different number of clusters on Cora-enrich. GETM estimates $K = 6$ by clearly showing the minimum of the negative ELBO.

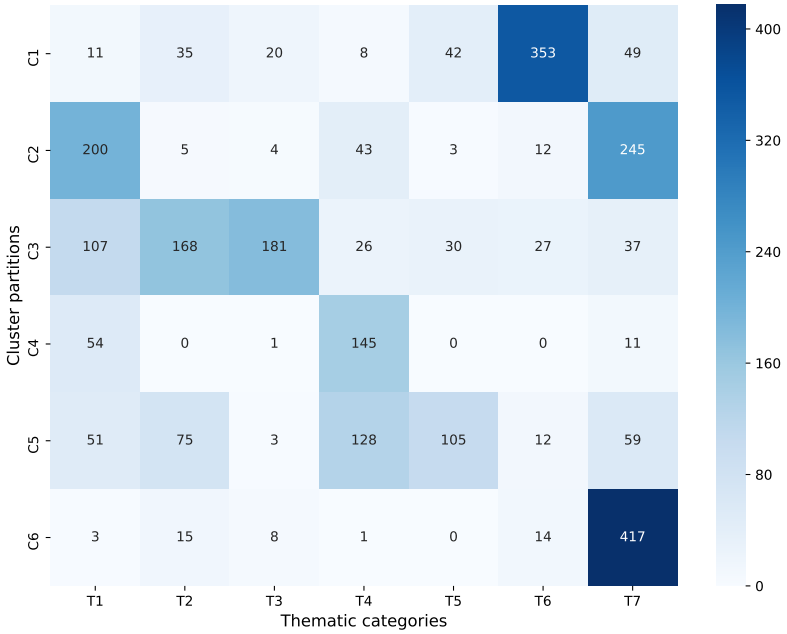


Figure 5.10 – Confusion matrix between six estimated clusters and seven thematic categories (T1: *probabilistic methods*, T2: *case based*, T3: *reinforcement learning*, T4: *theory*, T5: *rule learning*, T6: *genetic algorithms*, T7: *neural networks*).

networks (T7). Theoretical publications (T4) are mainly grouped into $C4$ and $C5$ clusters. Case based papers (T2) are separated into clusters $C3$ and $C5$. The articles that address probabilistic approaches (T1) are divided into the clusters $C2$, $C3$, $C4$, and $C5$.

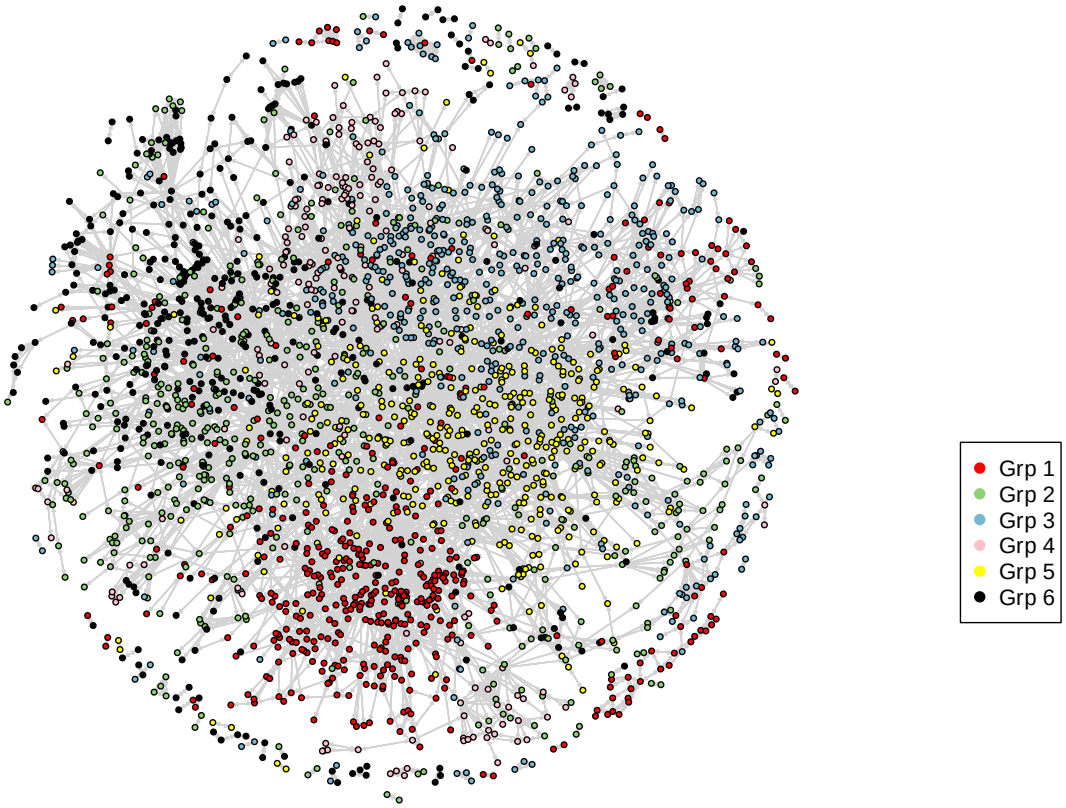


Figure 5.11 – Visualization of Cora-enrich. Six clusters are represented in distinct colors.

Based on these results, we stress that the number of clusters cannot be determined solely based on the number of the thematic classes. Conversely, when selecting the number of clusters via model selection, we are able to discover interesting new similarities between the nodes of a graph.

5.5.2 Visualisation and analysis

We further visualize the latent embeddings discovered by GETM in Figure 5.11. The standard network visualization tool *gplot* within the *sna* library in R is used. Six clusters are represented in distinct colors with a layout using a variant of Fruchterman and Reingold force-directed placement algorithm by default. It can be seen that, GETM was able to detect different communities on Cora-enrich, where nodes from various clusters are gathered. Moreover, Figure 5.12 shows the paper distributions in six groups when seven thematic categories are taken into account.

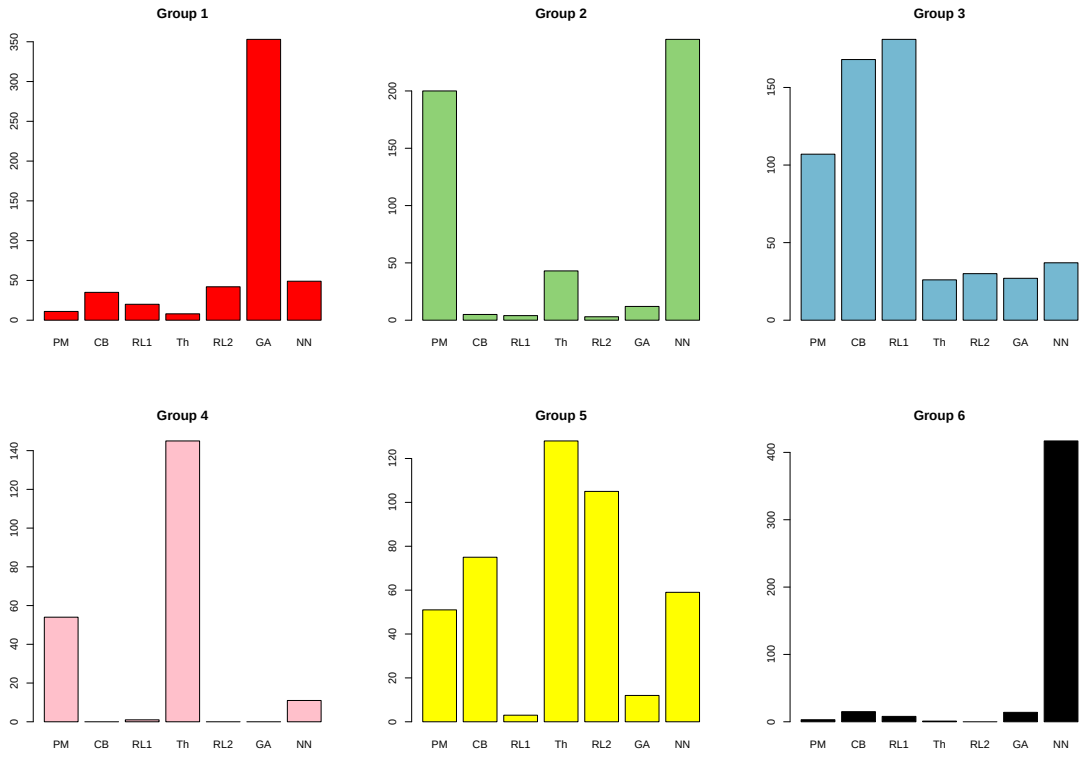


Figure 5.12 – Partitions taking into account thematics in each cluster on Cora-enrich (PM: *probabilistic methods*, CB: *case based*, RL1: *reinforcement learning*, Th: *theory*, RL2: *rule learning*, GA: *genetic algorithms*, NN: *neural networks*).

— The Grp 1 in red, in particular, collects the majority of publications on genetic algorithms. It makes sense because genetic techniques is a specialized topic of research that is distinct from other themes. Since many neural network models are constructed using probabilistic principles and it is common for them to strengthen proposed models based on theory background.

— Grp 2 in green extracts many publications from neural networks, probabilistic approaches, and some theoretical articles. This is expected given that theory papers or probabilistic methods can serve as the foundation for neural network models.

— Because many reinforcement learning models are created for a particular case study and may rely on some probabilistic basis, the Grp 3 (blue) includes practically almost all of the

reinforcement learning publications, a significant number of case-based articles, and some probabilistic methods.

— Similarly, a number of probabilistic or neural network-based models are gathered specifically in the Grp 4 (pink) together with the majority of theory articles.

— Since rule learning can be used in a variety of domains, Grp 5 in yellow contains the majority of rule learning publications along with many other types of papers.

— Finally, the Grp 6 in black captures the other neural network publications that are solely informatics-related.

Additionally, we also visualize the graph topological embedding and analyze the link connections between nodes. Figure 5.13 illustrates the latent embedding η of graph topology. As we can see, there is a small community of nodes in Grp 1 (red) that are exclusively interconnected, these could represent publications on specific genetic algorithms. The relationships for the other groups are not very evident from this figure, so we go further into them by computing the community memberships quantitatively for each cluster (number of links within each cluster). The results are reported in Table 5.4. It is clear that all of the diagonal values are quite large, highlighting the community structure: nodes are more likely to interact within their communities.

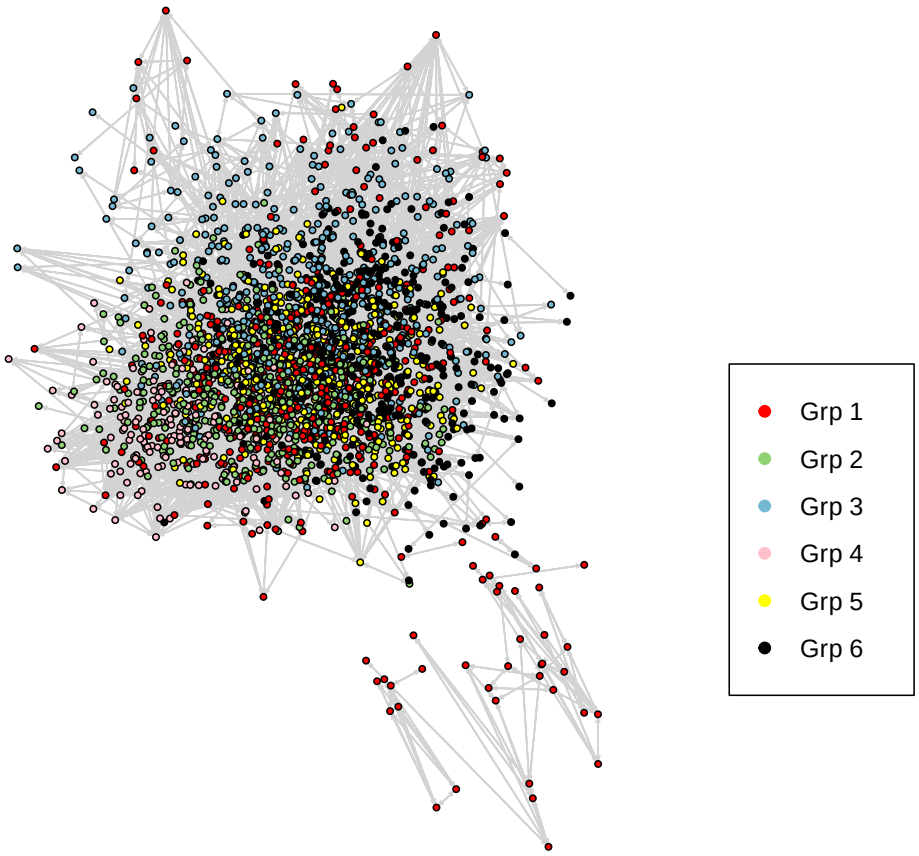


Figure 5.13 – Visualization of the graph topology embedding η.

Table 5.4 – Number of links within each cluster.

Grp \ Grp	1	2	3	4	5	6
1	**1794**	207	393	74	111	218
2	207	**1032**	79	106	117	119
3	393	79	**1480**	7	114	51
4	74	106	7	**568**	46	0
5	111	117	114	46	**1312**	29
6	218	119	51	0	29	**1030**

We close this section emphasizing once more that, in unsupervised problems, we cannot determine the number of clusters solely based on the number of the classes that are used in supervised tasks. These classes are simply assigned based on the article thematics, without taking into account the citation relationships. Conversely, when selecting the number of clusters via model selection (that VAEs seem to perform intrinsically), we are able to discover interesting new similarities between the nodes of a graph by combining the graph topology structure with latent thematics in textual information.

5.6 Conclusion

We propose the document similarity-based graph convolutional network (DS-GCN) to account for both the network topology structure as well as word and topic semantics from the textual information. Then, two different decoding networks are introduced to reconstruct both the graph adjacency matrix and the document-term matrix. In addition, an end-to-end node clustering is performed using the graph embedded topic model (GETM) by estimating the posterior probabilities for cluster memberships. Numerical experiments on simulated scenarios demonstrate that GETM is capable of learning representations in heterogeneous information network. Moreover, the performance of GETM in node clustering is highlighted by the benchmark study with other competitors based on three different simulations. We further conduct a model selection to test the ability of our methodology to auto-penalize the ELBO for choosing the number of clusters appropriately. Finally, an unsupervised network analysis is conducted on the Cora-enrich network to emphasize the model selection ability and the interest to discover hidden patterns behind the thematics.

Here we considered networks that include mixed-type information, but each node and link have the same kind of properties and relationships. For instance, each node is associated with a textual document and links are all undirected and unweighted. For future works, we could deal with more complex real-world networks, including nodes with various characteristics.

CHAPTER 6

Conclusion and Perspectives

6.1 Summary of the contributions

In both statistics and deep learning, dealing with high-dimensional and heterogeneous data is challenging. The performance of learning methods is directly impacted by how mixed-type data is incorporated. Moreover, labeling data is costly and prone to human annotator errors or biases. In this situation, understanding the underlying structure to extract data patterns and then generating new data based on the encoded information becomes a crucial approach, accomplished via deep latent variable models. Chapter 3 demonstrated that the predictability of our proposed deep latent recommender system could benefit from the integration of user ratings and textual reviews on products. Chapter 4 presented a latent position-based generative model to perform node clustering where, apart from the network data, additional covariate information can be incorporated into the model during the decoding phase to reveal more of the hidden patterns. Chapter 5 focused on networks with text data, where a document similarity-based graph convolutional network is developed to combine the graph topological structure with topic modeling and word embeddings. Numerical experiments on simulated scenarios as well as on real-world data demonstrated the validity of these three generative models along with their inference procedures.

These contributions led to the production of several scientific articles, among which one has been published in an international peer-reviewed journal :

— **DeepLTRS: A deep latent recommender system based on user ratings and reviews**, D., Liang, M., Corneli, C., Bouveyron and P., Latouche, *Pattern Recognition Letters*, vol. 152, pp. 267-274 (2021).

One paper was published in the proceedings of an international peer-reviewed conference :

— **Deep latent position model for node clustering in graphs**, D., Liang, M., Corneli, C., Bouveyron and P., Latouche, *Proceedings of the 30th European Symposium on Artificial Neural Networks* (2022).

One preprint is currently under review in a journal :

— **Clustering by Deep Latent Position Model with Graph Convolutional Network**, D., Liang, M., Corneli, C., Bouveyron and P., Latouche, Preprint HAL-03629104 (2022).

The work in Chapter 5, is submitted to a journal :

— **The graph embedded topic model**, D., Liang, M., Corneli, C., Bouveyron and P., Latouche, Preprint HAL-03942487 (2023).

6.2 Perspectives

Eventually, we outline several perspectives on extensions of the work described in this thesis as well as further research directions.

6.2.1 Graph learning-based recommender systems

The deepLTRS model we proposed in Chapter 3 is based on the traditional collaborative filtering approach, where the user-product interactions (ratings, reviews, etc.) are viewed as the encoded matrices. However, with the rapid development of graph learning techniques, graph learning-based recommender systems have emerged as a popular topic in the last couple of years [Wang et al., 2021]. In this context, users and items could be represented as nodes in a graph and the relations between them such as purchases or reviews could be viewed as edges. A demonstration of a graph learning-based recommender system is illustrated in Figure 6.1.

At the beginning, we could initially assess the performance of graph-based recommendations by modifying the model structures proposed in Chapters 4 and 5. Instead of performing node clustering, one possible approach is to adapt our methods to the link prediction task. For instance, we could construct a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consisting of a set of user nodes $u_i \in \mathcal{V}_u$ with $i \in \{1, \cdots, M\}$ and item nodes $v_j \in \mathcal{V}_v$ with $j \in \{1, \cdots, N\}$, such that $\mathcal{V} = \mathcal{V}_u \cup \mathcal{V}_v$. The interactions between them, e.g. purchases, could be represented as edges in E. Based on these information, we could be able to construct a graph adjacency matrix A, where $A_{ij} = 1$ if user i

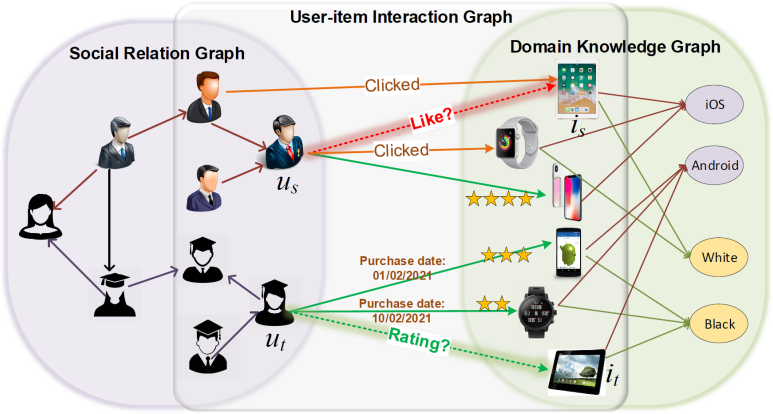


Figure 6.1 – A demonstration of graph learning=based recommender systems†.

purchased item j, 0 otherwise. Each edge could be associated with a weight to indicate the rating value. Moreover, each purchase is frequently accompanied by a text review. Hence, we may convert this textual data into a node features matrix X, where X_i contains all reviews written by the user i, similarly, X_j includes all reviews that the item j received. Once we obtained these two matrices, we could learn the link predictions using either of the graph neural networks discussed in Section 2.3.2. The system could then recommend the products to users if there are links between them. We point out here a number of related work, including the graph convolutional matrix completion [Berg et al., 2017] based on graph auto-encoders, the multi-graph convolutional neural networks [Monti et al., 2017] based on GCN, and the inductive graph-based matrix completion [Zhang and Chen, 2020] based on GraphSAGE.

6.2.2 Generalized graph neural networks

Regarding the generative models developed in Chapters 4 and 5, the encoding process is carried out based on the graph convolutional networks (GCNs). Even though GCNs have demonstrated good representation learning ability, the entire graph is needed as input for training. Indeed, GCNs take into account all of the neighbor nodes information during the aggregation, which could be computationally expensive in large networks. However, as discussed in Section 2.3.2, GraphSAGE learns an aggregation function that generates embeddings by sampling and aggregating features from a local neighborhood of nodes. With the inductive framework of GraphSAGE, it is

†. source from Wang et al. [2021]

not necessary to retrain the entire graph whenever a new node is added because the learnt function could be used to generate new embeddings. Moreover, the idea behind GraphSAGE [Hamilton et al., 2017] would allow us to perform mini-batch operations on huge data sets, which accelerates the training process in real-world applications.

PinSAGE [Ying et al., 2018a], which is incorporated into the recommendation engine of Pinterest, is a direct continuation of GraphSAGE. PinSAGE is applied in a very large graph (3 billion nodes and 18 billion edges) and generates higher-quality recommendations than comparable deep learning and graph-based alternatives.

Due to the high performance of GraphSAGE, we could first change the GCN-based encoder in our models to GraphSAGE to obtain preliminary findings. Moreover, because of the flexible architecture of VAE, the encoding-decoding process can be modified for different purposes. Therefore, in order to enhance our models, one possible way is to randomly choose the encoders from various GNNs (GCN, GraphSAGE, GIN, etc.) according to specific problems, which could bring us a more generalized model structure.

6.2.3 Clustering with heterogeneous graph neural networks

In this thesis, we considered networks that may include mixed-type information, but each node and link have the same kinds of properties and relationships. For instance, each node is associated with a textual document and links are all undirected and unweighted, while in more complex real-

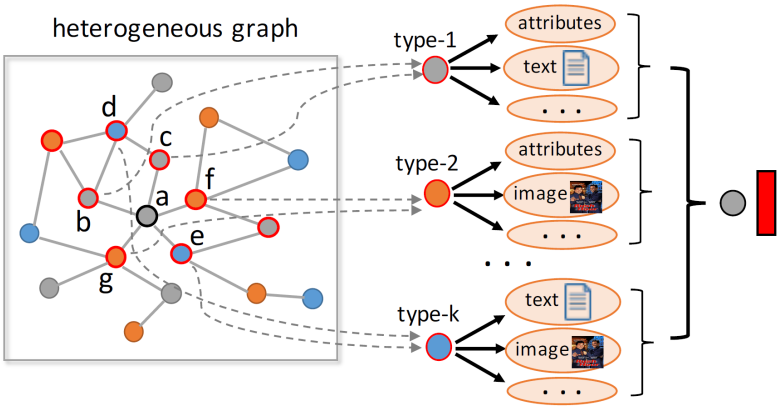


Figure 6.2 – A complex network where neighbors and node contents are heterogeneous†.

†. source from Zhang et al. [2019a]

world networks, we could come across nodes with various characteristics, as seen in Figure 6.2. In this illustration, the network consists of three different types of nodes, where the first type of nodes carries text information, the second type contains image data, etc.

To overcome this problem, recent advancements have been made with heterogeneous graph neural networks (HGNNs) [Zhang et al., 2019a; Wang et al., 2019b]. More precisely, in Zhang et al. [2019a], a heterogeneous graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{O}_V, \mathcal{R}_E)$ with multiple types of nodes in $\mathcal{V}$ and links in $\mathcal{E}$. $\mathcal{O}_V$ and $\mathcal{R}_E$ indicate the set of object types and that of relation types. For different node attributes, the text features are extracted by Par2Vec [Mikolov et al., 2013b], the image features are obtained using CNN, and a pooling operation is used to integrate all types of features. Due to the efficiency and practical interest of HGNNs in carrying out various graph learning tasks, this is another subject that we plan to investigate in the future.

At first sight, one possibility might be changing the aggregation processes between nodes and their neighbors in our proposed models. Assume we have three sorts of nodes: nodes with text data, nodes with image data, and those with both forms of data. We could use Par2vec to extract text features, and use CNN to extract image features. For nodes with two types of content, we might conduct these two procedures and aggregate the resulting features using concatenation or summation. The attention mechanism, as in GAT (see Section 2.3.2), could then be used to model the importance of the neighbor nodes. If two nodes have the same type of attributes, we may give them a large weight, otherwise, we may give them a small weight.

6.2.4 From topic modeling to intelligent document analysis techniques

In this thesis, the bag-of-words representation has been used to model the text information into a document-term matrix without taking into account the ordering, semantics, or relationships between individual words. We are considering using more deep oriented techniques for document analysis, such as doc2vec [Le and Mikolov, 2014], to convert the documents directly into vectors. However, as the length of the texts might vary in doc2vec, the vector size is different for each document. The integration of a document vector with a range of sizes into the graph structure is challenging yet interesting.

Furthermore, in order to improve the modeling ability of text, we might replace LDA-based models by other NLP models, possibly involving RNNs [Agarap and Grafilon, 2018] or BERT [Xu et al., 2020], etc.

Appendix

Appendices

A Appendix for Chapter 4

A.1 Implementation details and computation time

In DeepLPM, the GCN encoder has 64 neurons in the first hidden layer, equipped with a RelU activation function and 16 neurons in the second hidden layer. The decoder is a one-layer neural network which maps the latent embeddings into a reconstructed graph adjacency matrix, following by a sigmoid activation function. Adam optimizer with a learning rate equal to 2e-3 is used to update network parameters. The computation time on the ecclesiastical network with 1,287 nodes and 33,384 edges is about 0.12s/epoch, for a total of 107s, for 800 epochs on a GeForce RTX 2070 GPU. On the same GPU, training on the citation network Cora with 2,708 nodes and 5,429 edges takes about 0.18s/epoch and a total of 444.37s for 2,000 epochs.

In this chapter, LPCM is implemented by *VBLPCM* pakage in R, SBM, VGAE and ARVGA are conducted using the available Python code in github: https://github.com/Remi-Boutin/SBM_package, https://github.com/DaehanKim/vgae_pytorch and https://github.com/GRAND-Lab/ARGA, respectively.

B Appendix for Chapter 5

B.1 Implementation details and computation time

In GETM, the DS-GCN encoder g_ϕ has 512 neurons in the first hidden layer and 128 neurons in the second hidden layer, respectively, equipped with a Relu activation for the first layer. The neural networks $h_\iota^{(G)}$ and $h_\nu^{(T)}$ are one-layer linear networks with 128 and 16 neurons, respectively. The graph decoder f_τ is a one-layer neural network, following with a sigmoid function, which maps the latent graph topology embeddings η into a reconstructed graph. The document decoder $\theta^\top \beta$ maps the topic and word embeddings in dimension $L = 300$ into a reconstructed document-term matrix.

Adam optimizer is used to update network weights. On the simulated networks, the learning rate for the graph part is $5e^{-3}$, and 0.02 for the document part. On the Cora-enrich network, the learning rate for two parts are $5e^{-3}$, and 0.01, respectively.

The computation time on the simulated network with 900 nodes, 558 (Scenario B and C) or 721 words (Scenario A) is about 0.03s/epoch, for a total of 9.18s for 300 epochs on a GeForce

RTX 2070 GPU. On the same GPU, training on the citation network Cora-enrich with 2,708 nodes and 25,955 words takes about 0.10s/epoch and a total of 28.50s for 300 epochs.

For more details, our code is available in: https://github.com/ldggggg/GraphETM. In this paper, SBM is implemented by *sparsebm* package in Python, ETM, VGAE and AM-GCN are conducted using the available Python code in github: https://github.com/lffloyd/embedded-topic-model, https://github.com/DaehanKim/vgae_pytorch and https://github.com/zhumeiqiBUPT/AM-GCN, respectively.

Bibliography

A. F. Agarap and P. Grafilon. Statistical analysis on e-commerce reviews, with sentiment classification using bidirectional recurrent neural network (rnn). *arXiv preprint arXiv:1805.03687*, 2018.

C. Aggarwal and H. Wang. A survey of clustering algorithms for graph data. In *Managing and mining graph data*, pages 275–301. Springer, 2010.

C. Aggarwal, G. He, and P. Zhao. Edge classification in networks. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 1038–1049. IEEE, 2016.

E. M. Airoldi, D. Blei, S. Fienberg, and E. Xing. Mixed membership stochastic blockmodels. *Advances in neural information processing systems*, 21, 2008.

D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner. *Graph partitioning and graph clustering*, volume 588. American Mathematical Society Providence, RI, 2013.

A.-L. Barabasi and Z. N. Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113, 2004.

M. Basbug and B. Engelhardt. Hierarchical compound poisson factorization. In *International Conference on Machine Learning*, pages 1795–1803, 2016.

M. Basbug and B. Engelhardt. Coupled compound poisson factorization. *arXiv preprint arXiv:1701.02058*, 2017.

R. v. d. Berg, T. N. Kipf, and M. Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

C. M. Bishop. Latent variable models. In *Learning in graphical models*, pages 371–403. Springer, 1998.

D. M. Blei. Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and Its Application*, 2014.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, apr 2017.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

L. Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

C. Bouveyron, P. Latouche, and R. Zreik. The stochastic topic block model for the clustering of vertices in networks with textual edges. *Statistics and Computing*, 28(1):11–31, 2018.

C. Bouveyron, G. Celeux, T. B. Murphy, and A. E. Raftery. *Model-based clustering and classification for data science: with applications in R*, volume 50. Cambridge University Press, 2019.

R. Catherine and W. Cohen. Transnets: Learning to transform for recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 288–296, 2017.

L. Chen, G. Chen, and F. Wang. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction*, 25(2):99–154, 2015.

Z. Cheng, Y. Ding, L. Zhu, and M. Kankanhalli. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *Proceedings of the 2018 world wide web conference*, pages 639–648, 2018.

K. Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649, 2020.

M. Corneli, C. Bouveyron, P. Latouche, and F. Rossi. The dynamic stochastic topic block model for dynamic networks with textual edges. *Statistics and Computing*, 29(4):677–695, 2019.

S. Coxe, S. G. West, and L. S. Aiken. The analysis of count data: A gentle introduction to poisson regression and its alternatives. *Journal of personality assessment*, 91(2):121–136, 2009.

I. Csiszár. I-divergence geometry of probability distributions and minimization problems. *The annals of probability*, pages 146–158, 1975.

W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE transactions on visualization and computer graphics*, 14(6):1277–1284, 2008.

B. Dai, Y. Wang, J. Aston, G. Hua, and D. Wipf. Hidden talents of the variational autoencoder. *arXiv preprint arXiv:1706.05148*, 2017.

A. B. Dieng, F. J. Ruiz, and D. M. Blei. Topic modeling in embedding spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453, 2020.

Z. Gan, C. Chen, R. Henao, D. Carlson, and L. Carin. Scalable deep poisson factor analysis for topic modeling. In *International Conference on Machine Learning*, pages 1823–1832. PMLR, 2015.

S. Ganguly and V. Pudi. Paper2vec: Combining graph and text information for scientific paper representation. In *European conference on information retrieval*, pages 383–395. Springer, 2017.

H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang. Context-aware qos prediction with neural collaborative filtering for internet-of-things services. *IEEE Internet of Things Journal*, 7 (5):4532–4542, 2019.

Z. Gilula, R. E. McCulloch, Y. Ritov, and O. Urminsky. A study into mechanisms of attitudinal scale conversion: A randomized stochastic ordering approach. *Quantitative Marketing and Economics*, 17(3):325–357, 2019.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

P. Gopalan, J. M. Hofman, and D. M. Blei. Scalable recommendation with hierarchical poisson factorization. In *UAI*, pages 326–335, 2015.

W. Greene. Functional forms for the negative binomial model for count data. *Economics Letters*, 99(3):585–590, 2008.

J. Grimmer and B. M. Stewart. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political analysis*, 21(3):267–297, 2013.

W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

M. S. Handcock, A. E. Raftery, and J. M. Tantrum. Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354, 2007.

J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

G. E. Hinton and R. R. Salakhutdinov. Replicated softmax: an undirected topic model. *Advances in neural information processing systems*, 22, 2009.

P. Hoff, A. Raftery, and M. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, 2002.

Z. Huang, D. Zeng, and H. Chen. A comparison of collaborative-filtering recommendation algorithms for e-commerce. *IEEE Intelligent Systems*, 22(5):68–78, 2007.

L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

Y. Jernite, P. Latouche, C. Bouveyron, P. Rivera, L. Jegou, and S. Lamassé. The random subgraph model for the analysis of an ecclesiastical network in merovingian gaul. *The Annals of Applied Statistics*, 8(1):377–405, 2014.

Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *International Joint Conference on Artificial Intelligence (IJCAI-2017)*, 2016.

D. Jin, X. Song, Z. Yu, Z. Liu, H. Zhang, Z. Cheng, and J. Han. Bite-gcn: A new gcn architecture via bidirectional convolution of topology and features on text-rich networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 157–165, 2021.

Z. Y. Khan, Z. Niu, S. Sandiwarno, and R. Prince. Deep learning techniques for rating prediction: a survey of the state-of-the-art. *Artificial Intelligence Review*, 54(1):95–135, 2021.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *International conference on Learning Representations*, 2014a.

D. P. Kingma and M. Welling. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations*, volume 19, page 121, 2014b.

D. P. Kingma and M. Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. ISSN 1935-8237.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29:4743–4751, 2016.

T. N. Kipf and M. Welling. Variational graph auto-encoders. In *NeurIPS Workshop on Bayesian Deep Learning (NeurIPS-16 BDL)*, 2016.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations (ICLR-17)*, 2017.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

D. Koutra, A. Parikh, A. Ramdas, and J. Xiang. Algorithms for graph similarity and subgraph matching. In *Proc. Ecol. inference conf*, volume 17. Citeseer, 2011.

N. M. Kriege, F. D. Johansson, and C. Morris. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020.

S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.

A. Kumar, S. S. Singh, K. Singh, and B. Biswas. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, 553:124289, 2020.

P. Latouche, E. Birmelé, and C. Ambroise. Overlapping stochastic block models with application to the french political blogosphere. *The Annals of Applied Statistics*, pages 309–336, 2011.

Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

C. Lee and D. J. Wilkinson. A review of stochastic block models and extensions for graph clustering. *Applied Network Science*, 4(1):1–50, 2019.

D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13, 2000.

J. Li, J. Yu, J. Li, H. Zhang, K. Zhao, Y. Rong, H. Cheng, and J. Huang. Dirichlet graph variational autoencoder. *Advances in Neural Information Processing Systems*, 33:5274–5283, 2020.

R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 243–252, 2010.

C. Liu, T. Jin, S. C. Hoi, P. Zhao, and J. Sun. Collaborative topic regression for online recommender systems: an online and bayesian approach. *Machine Learning*, 106(5):651–670, 2017.

X. Luo, Y. Yuan, M. Zhou, Z. Liu, and M. Shang. Non-negative latent factor model based on β-divergence for recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019a.

X. Luo, M. Zhou, S. Li, D. Wu, Z. Liu, and M. Shang. Algorithms of unconstrained non-negative latent factor analysis for recommender systems. *IEEE Transactions on Big Data*, 2019b.

X. Luo, W. Qin, A. Dong, K. Sedraoui, and M. Zhou. Efficient and high-quality recommendations via momentum-incorporated parallel stochastic gradient descent-based learning. *IEEE/CAA Journal of Automatica Sinica*, 8(2):402–411, 2020.

G. Ma, N. K. Ahmed, T. L. Willke, and P. S. Yu. Deep graph similarity learning: A survey. *Data Mining and Knowledge Discovery*, 35(3):688–725, 2021.

A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *International Conference on Learning Representations (ICLR-16)*, 2016.

F. D. Malliaros and M. Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics reports*, 533(4):95–142, 2013.

M. Mariadassou, S. Robin, and C. Vacher. Uncovering latent structure in valued graphs: a variational approach. *The Annals of Applied Statistics*, 4(2):715–742, 2010.

C. Matias and V. Miele. Statistical clustering of temporal networks through a dynamic stochastic block model. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(4): 1119–1141, 2017.

J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172, 2013.

N. Mehta, L. C. Duke, and P. Rai. Stochastic blockmodels meet graph neural networks. In *International Conference on Machine Learning*, pages 4466–4474. PMLR, 2019.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *International Conference on Learning Representations*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013b.

A. Mnih and R. R. Salakhutdinov. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20, 2007.

F. Monti, M. Bronstein, and X. Bresson. Geometric matrix completion with recurrent multi-graph neural networks. *Advances in neural information processing systems*, 30, 2017.

F. Nie, W. Zhu, and X. Li. Unsupervised large graph embedding. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

K. Nowicki and T. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087, 2001.

A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

G. Palla, A.-L. Barabási, and T. Vicsek. Quantifying social group evolution. *Nature*, 446(7136): 664–667, 2007.

S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang. Adversarially regularized graph autoencoder for graph embedding. In *International Joint Conference on Artificial Intelligence (IJCAI-18)*, pages 2609–2615, 2018.

J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6519–6528, 2019.

A. E. Raftery. Comment: Extending the latent position model for networks. *Journal of the American Statistical Association*, 112(520):1531–1534, 2017.

A. Ramlatchan, M. Yang, Q. Liu, M. Li, J. Wang, and Y. Li. A survey of matrix completion methods for recommendation systems. *Big Data Mining and Analytics*, 1(4):308–323, 2018.

S. Rendle. Factorization machines. In *2010 IEEE International conference on data mining*, pages 995–1000. IEEE, 2010.

D. A. Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and variational inference in deep latent gaussian models. In *International conference on machine learning*, volume 2, page 2. Citeseer, 2014.

J. T. Rolfe. Discrete variational autoencoders. *International Conference on Learning Representations (ICLR-17)*, 2017.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887, 2008.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

S. E. Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

D. K. Sewell and Y. Chen. Latent space approaches to community detection in dynamic networks. *Bayesian analysis*, 12(2):351–377, 2017.

T. A. Snijders. Statistical models for social networks. *Annual review of sociology*, 37:131–153, 2011.

A. Srivastava and C. Sutton. Autoencoding variational inference for topic models. *International Conference on Learning Representations (ICLR-17)*, 2017.

S. Stemler. An overview of content analysis. *Practical assessment, research, and evaluation*, 7 (1):17, 2000.

X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

Y. Teh, D. Newman, and M. Welling. A collapsed variational bayesian inference algorithm for latent Dirichlet allocation. *Advances in neural information processing systems*, 18:1353–1360, 2006.

F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu. Learning deep representations for graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

I. C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, 85(6):317–325, 2003.

S. Vashishth, N. Yadati, and P. Talukdar. Graph-based deep learning in natural language processing. In *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, pages 371–372. 2020.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *International Conference on Learning Representations (ICLR-18)*, 2018.

C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456, 2011.

C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898, 2017.

C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang. Attributed graph clustering: a deep attentional embedding approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3670–3676, 2019a.

F. Wang, P. Cui, J. Pei, Y. Song, and C. Zang. Recent advances on graph analytics and its applications in healthcare. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3545–3546, 2020a.

S. Wang, L. Hu, Y. Wang, X. He, Q. Sheng, M. Orgun, L. Cao, F. Ricci, and P. Yu. Graph learning based recommender systems: A review. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2021.

X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019b.

X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei. Am-gcn: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining*, pages 1243–1253, 2020b.

Y. J. Wang and G. Y. Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19, 1987.

B. Weisfeiler and A. Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

S. Xiao, S. Wang, Y. Dai, and W. Guo. Graph neural networks in node classification: survey and evaluation. *Machine Vision and Applications*, 33(1):1–19, 2022.

A. Xu and X. Zheng. Dynamic social network analysis using latent space model and an integrated clustering algorithm. In *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 620–625. IEEE, 2009.

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR-19)*, 2019.

K. S. Xu and A. O. Hero. Dynamic stochastic blockmodels for time-evolving social networks. *IEEE Journal of Selected Topics in Signal Processing*, 8(4):552–562, 2014.

S. Xu, S. E. Barbosa, and D. Hong. Bert feature based model for predicting the helpfulness scores of online customers reviews. In *Future of Information and Communication Conference*, pages 270–281. Springer, 2020.

R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018a.

Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018b.

Z. Yu, D. Jin, Z. Liu, D. He, X. Wang, H. Tong, and J. Han. As-gcn: Adaptive semantic architecture of graph convolutional networks for text-rich networks. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 837–846. IEEE, 2021.

C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 793–803, 2019a.

D. Zhang, J. Yin, X. Zhu, and C. Zhang. Network representation learning: A survey. *IEEE transactions on Big Data*, 6(1):3–28, 2018.

M. Zhang and Y. Chen. Inductive matrix completion based on graph neural networks. In *International Conference on Learning Representations*, 2020.

X. Zhang, H. Liu, Q. Li, and X.-M. Wu. Attributed graph clustering via adaptive graph convolution. *arXiv preprint arXiv:1906.01210*, 2019b.

Y. Zhang, R. Jin, and Z.-H. Zhou. Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics*, 1(1):43–52, 2010.

L. Zheng, V. Noroozi, and P. S. Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 425–434, 2017.

Modèles probabilistes profonds pour les systèmes de recommandation et le clustering de réseaux

Dingge LIANG