Clustering by Deep Latent Position Model with Graph Convolutional Network

Dingge Liang, Marco Corneli, Charles Bouveyron, Pierre Latouche

Clustering by Deep Latent Position Model with Graph Convolutional Network

Dingge Liang[a,*], Marco Corneli[a,b], Charles Bouveyron[a], Pierre Latouche[c]

[a] *Université Côte d'Azur, INRIA, CNRS, Laboratoire J.A.Dieudonné, Maasai team, Nice, France*
[b] *Center of modeling, Simulation and Interactions (MSI), Nice, France*
[c] *Université Paris Cité, CNRS, Laboratoire MAP5, UMR 8145, Paris, France*

Abstract

With the significant increase of interactions between individuals through numeric means, clustering of vertices in graphs has become a fundamental approach for analyzing large and complex networks. In this work, we propose 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 estimation algorithm is introduced to integrate the explicit optimization of the posterior clustering probabilities via variational inference and the implicit optimization using stochastic gradient descent for graph reconstruction. 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 a citation network Cora to illustrate the practical interest in exploring large and complex real-world networks.

Keywords: Network Analysis, Graph Clustering, Unsupervised Deep Learning, Graph Neural Networks

*Corresponding author
Email address: dingge.liang@inria.fr (Dingge Liang)

1. Introduction and related work

Networks are used in many 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.

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 for analyzing large-scale networks and they have shown their abilities for representation learning purpose on data with complex structures (Hamilton et al., 2017; Zhang et al., 2018). We hereafter split the existing approaches for vertex clustering in networks into two categories and briefly review them.

Statistical models for clustering. On the one hand, the stochastic block model (SBM, Wang and Wong, 1987; Nowicki and Snijders, 2001) is widely used to detect communities or more general clusters of nodes (Lee and Wilkinson, 2019). It assumes that nodes are spread into different latent clusters and that the connection probability between each pair of nodes depends exclusively on their group memberships. Based on SBM, many extensions looking for overlapping clusters have been proposed. For instance, the mixed-membership stochastic blockmodel (MMSBM, Airoldi et al., 2008) introduces a mixing weight vector π_i drawn from the Dirichlet distribution for each vertex, while the overlapping stochastic blockmodel (OSBM, Latouche et al., 2011) assumes each node to be characterized by a binary latent vector sampled from a product of Bernoulli distributions, allowing each node to belong to multiple clusters. Other variants consider the processing of valued graphs, such as networks with discrete edges (Mariadassou et al., 2010), categorical edges (Jernite et al., 2014) or text edges (Bouveyron et al., 2018). Moreover, some extensions (Xu and Hero, 2014; Matias and Miele, 2017; Corneli et al., 2019) allow to deal with time-evolving networks through dynamic network modeling. On the other hand, a different approach to model network data relies on the potential position of nodes. Originally proposed by Hoff et al. (2002), the latent position

model (LPM) supposes that each node has an unknown position in a latent space and that the probability of a specific link between two nodes is modelled by some function of their positions. 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). Further developments of LPMs exist and the reader is referred to Raftery (2017) for an extensive review. Nevertheless, these statistical models face a challenging inference procedure that primarily relies on MCMC or variational approximations and do not scale easily to large and complex networks. A more general overview of statistical models for clustering network data can be explored in Bouveyron et al. (2019, Chapter 10).

Deep learning models for clustering. From another aspect, deep neural network (DNN)-based techniques have recently shown to be effective for feature representation learning and have been actively explored in clustering (Aljalbout et al., 2018; Zhang et al., 2020). Two widely used approaches are deep embedded clustering (DEC, Xie et al., 2016) and variational deep embedding (VaDE, Jiang et al., 2016). VaDE models the data generative procedure by combining GMM prior distributions with variational auto-encoding (VAE, Kingma and Welling, 2014), while DEC learns a mapping function and imposes a soft assignment constraint on the latent features. However, neither VaDE nor DEC are designed for graph-structured data. For the purpose of performing vertex clustering on networks, new models were introduced based on graph neural networks (GNNs). In this line of methods, the variational graph auto-encoder (VGAE, Kipf and Welling, 2016b) adopts a graph convolutional network (GCN, Kipf and Welling, 2016a) encoder to produce nodes embeddings in the latent space and a simple inner product decoder for graph reconstruction. As an extension, Wang et al. (MGAE, 2017) proposed a marginalization process that allows node content to interact with network features. By introducing adversarial learning into the generation process, Pan et al. (ARVGA, 2018) enforced the latent representation to match a prior distribution. Lately, Mehta et al. (DGLFRM, 2019) combined OSBM with GCN by positing each node of the graph to have an embedding modelled by a Beta-Bernoulli process. All the above mentioned approaches employ inner-product-based decoders, whereas we argue that a different

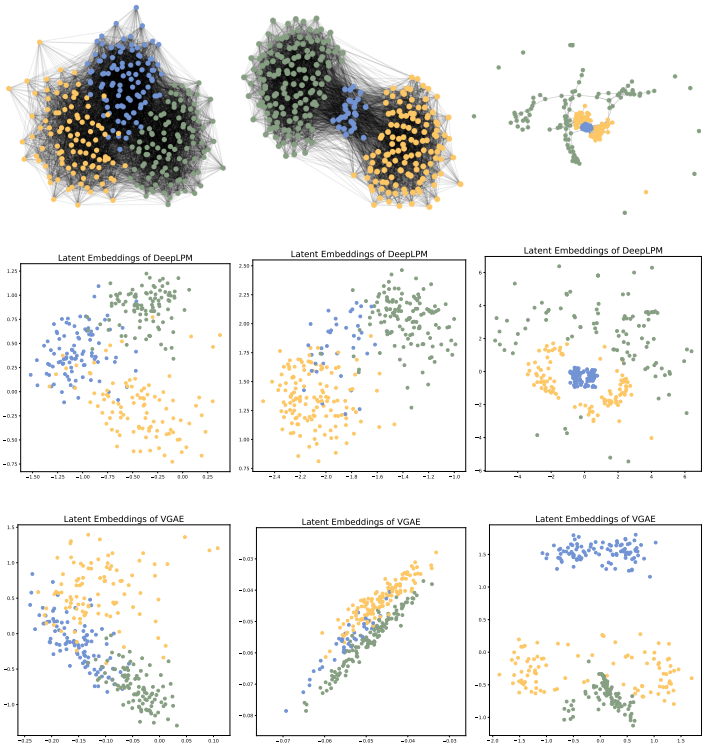

Figure 1: Simulated networks and learned embeddings in three scenarios. From top to bottom: the original simulated graphs, the latent embeddings learned by DeepLPM and latent embeddings learned by VGAE. To facilitate the visualization, the latent dimension is set to 2.

solution, accounting for the Euclidean distance between nodes in the latent space might be more suited. Additionally, these and other existing approaches (Tian et al., 2014; Nie et al., 2017; Zhang et al., 2019) 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.

Main contributions. In order to overcome the limitations of the methods listed above, 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. 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 statistical model considered. The 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 the latent space. Compared with a standard inner-product-based decoder, this choice better preserves the network topology in different scenarios (see Figure 1);
- DeepLPM performs an *end-to-end* clustering of the nodes by estimating the posterior probabilities for cluster memberships. Thus, it can automatically assign each node to its group without using any additional algorithms;
- 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.

Organization of the paper. In Section 2, we introduce the generative model behind DeepLPM. The variational inference and the original optimization algorithm are discussed in Section 3. Numerical experiments are provided in Section 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 5 and an analysis of the citation network Cora is described in Section 6. Section 7 finally concludes with a summary of this work.

2. Deep latent position model

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

Table 1: List of all model parameters

Notation	Description	Notation	Description
A	Adjacency matrix in $[0,1]^{N\times N}$	Y	Edge feature matrix
N	Number of nodes	K	Number of clusters
P	Latent space dimension	D	Edge features dimension
π	Prior cluster probability vector	C	Cluster memberships
Z	Latent node embeddings in $\mathbb{R}^P$	$f_{\alpha,\beta}$	Decoder parametrized by α, β
g_ϕ	GCN encoder parametrized by ϕ	γ_{ik}	Posterior clustering probability that node i is in cluster k

2.1. Notations

In this work, networks are modelled 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 $Y \in \mathbb{R}^{|E|\times D}$. The generic entry of Y, denoted y_{ij}, is a D-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. Necessary notations are summarized in Table 1.

2.2. Generative model

As in LPM (Hoff et al., 2002), 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 modelled as a function of the distance between the two nodes, in the latent space. The generative process is as follows.

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

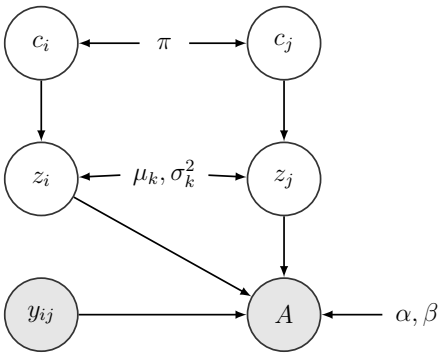

Figure 2: Graphical representation of DeepLPM (variational parameters are not included).

membership

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

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

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

independently for each node, where μ_k and σ_k^2 denote the mean and variance of each cluster, I_P denotes the 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 modelled through a Bernoulli random variable related to the distance between the corresponding latent positions

$$A_{ij} | z_i, z_j \sim \mathcal{B}(f_{\alpha,\beta}(z_i, z_j)), \tag{3}$$

with

$$f_{\alpha,\beta}(z_i, z_j) = \sigma(\alpha + \beta^T y_{ij} - ||z_i - z_j||^2), \tag{4}$$

where $f_{\alpha,\beta}$ can be seen as a *decoding*, one-layer, neural network parametrized by α and β. Moreover, σ 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 2.

2.3. Links with related models

At this point, DeepLPM can be linked with the following models:

- In LPCM, a specific prior distribution for the parameters $\beta = (\beta_0^T, \beta_1)^T$ 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 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 and integrates the cluster memberships to achieve an end-to-end clustering.

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.

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)

$$\log p(A|\Theta) = \log \int_Z \sum_C p(A, Z, C|\Theta) dZ. \tag{5}$$

Unfortunately, Eq. (5) is not tractable and we have to rely on a variational approach to approximate it

$$\log p(A|\Theta) = \mathcal{L}(q(Z, C); \Theta) + D_{KL}(q(Z, C)||p(Z, C|A, \Theta)), \tag{6}$$

where D_{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

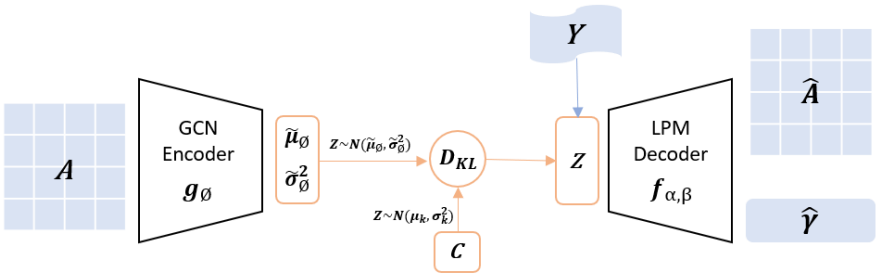

Figure 3: A deep-learning-like model view of DeepLPM.

to deal with a tractable family of distributions, $q(Z, C)$ is assumed to fully factorize (*mean-field* assumption)

$$q(Z, C) = q(Z)q(C) = \prod_{i=1}^{N} q(z_i)q(c_i). \tag{7}$$

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

$$q(z_i) = \mathcal{N}(z_i; \tilde{\mu}_\phi(\overline{A})_i, \tilde{\sigma}^2_\phi(\overline{A})_i I_P), \tag{8}$$

where $\tilde{\mu}_\phi(\cdot) : \mathbb{R}^{N\times N} \mapsto \mathbb{R}^{N\times P}$ (respectively $\tilde{\sigma}^2_\phi(\cdot) : \mathbb{R}^{N\times N} \mapsto \mathbb{R}^{+N}$) is the function mapping the normalized adjacency matrix $\overline{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ (Kipf and Welling, 2016a) into the matrix of the variational means (vector of the standard deviations). In the above equation, $\tilde{\mu}_\phi(\overline{A})_i$ denotes the i-th row of $\tilde{\mu}_\phi(\overline{A})$, corresponding to the variational mean for the latent position z_i (similarly for $\tilde{\sigma}^2_\phi(\overline{A})_i$). The functions $\tilde{\mu}_\phi(\cdot)$ and $\tilde{\sigma}^2_\phi(\cdot)$ are parametrized by the GCN *encoder* g_ϕ.

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

$$q(C) = \prod_{i=1}^{N} \mathcal{M}(c_i; 1, \gamma_i), \tag{9}$$

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 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; 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 $\hat{A}$ and obtain the cluster probability matrix $\hat{\gamma}$.

3.2. Optimization

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

$$\mathcal{L}(A|\Theta) = \int_Z \sum_C q(Z,C) \log \frac{p(A,Z,C|\Theta)dZ}{q(Z,C)} \tag{10}$$

with respect to the model parameters Θ and the variational parameters ϕ. Thanks to Equations (7)-(8)-(9), Eq. (10) can be further developed as

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

where $\eta_{ij} = \sigma(\alpha + \beta^T y_{ij} - ||z_i - z_j||^2)$, $D_{KL}(\cdot)$ 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 to obtain the following

updates:

$$\hat{\gamma}_{ik} = \frac{\pi_k \mathrm{e}^{-D_{KL}^{ik}}}{\sum_{l=1}^{K} \pi_l \mathrm{e}^{-D_{KL}^{il}}}, \tag{11}$$

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

Then

$$\hat{\pi}_k = \sum_{i=1}^{N} \gamma_{ik}/N, \tag{12}$$

$$\hat{\mu}_k = \sum_{i=1}^{N} \tilde{\mu}_\phi(\overline{A})_i \gamma_{ik} / \sum_{i=1}^{N} \gamma_{ik}, \tag{13}$$

and

$$\hat{\sigma}_k^2 = \frac{\sum_{i=1}^{N} \gamma_{ik}(\sigma_\phi^2(\overline{A})_i + ||\mu_k - \tilde{\mu}_\phi(\overline{A})_i||^2)}{P \sum_{i=1}^{N} \gamma_{ik}}. \tag{14}$$

Detailed derivations are given in the appendix.

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 optimiser.

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 $\tilde{\mu}_\phi(\overline{A})_i$ and variance $\tilde{\sigma}_\phi^2(\overline{A})_i$ 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. 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 1.

Algorithm 1 Estimation of DeepLPM

Input: adjacency matrix A, edge features Y

pretrain_model = pretrain(A, 50 epochs) ▷ pre-training to save initial weights of encoder/decoder

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

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

explicit optimization:

update $\hat{\gamma}_{ik}$ by Equation (11)

update $\hat{\pi}_k, \hat{\mu}_k, \hat{\sigma}^2_k$ by Equations (12)-(13)-(14)

calculate loss $-\mathcal{L}$

implicit optimization:

update encoder parameter ϕ and decoder parameters α, β

3.3. 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 becomes completely different. Therefore, choosing appropriate values for P and K can be considered as a model selection task.

Regarding the model selection, 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 4.3, we conduct several experiments to show that DeepLPM does 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 to select the intrinsic dimension P of the latent space and the number K of clusters.

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.

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.

4.2. 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, 2016b) and ARVGA (Pan et al., 2018) on simuated datasets in three scenarios. To facilitate the experiments, we do not consider the covariate information Y in simulated data, thus β in Eq. (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 10 different networks and calculated the averaged adjusted rand index (ARI). Experimental results of clustering are shown in Table 2.

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

Table 2: 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

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 clustering value of ARI.

Lastly, on the circular-structured data, all deep learning-based methods perform better than the ones based on statistical models. ARVGA presents the highest clustering ARI compared to the other deep models. DeepLPM and VGAE have a slightly lower ARI.

Robustness. To further demonstrate the robustness of DeepLPM compared to other competitors, Figures 4 and 5 illustrate the evolution of the clustering ARI in scenario A and scenario B. For all models, we varied the parameter δ from 0.2 to 0.95 and $\delta^{'}$ from 0.4 to 1 to compare the clustering performances. We can see that DeepLPM has the highest ARI with a small variance in all situations. Moreover, Figure 6 shows the embeddings learned by ARVGA, VGAE and DeepLPM with latent dimension equal to 2 in scenario C. It can be seen that DeepLPM better preserves the network topology.

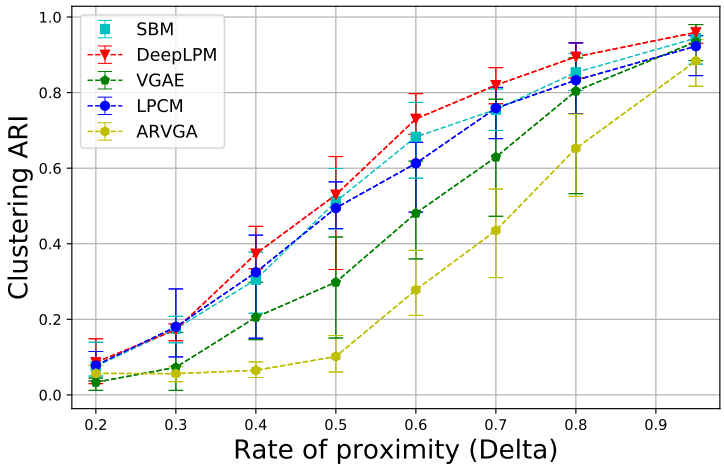

Figure 4: Clustering ARI with different proximity rate δ in Sc.A.

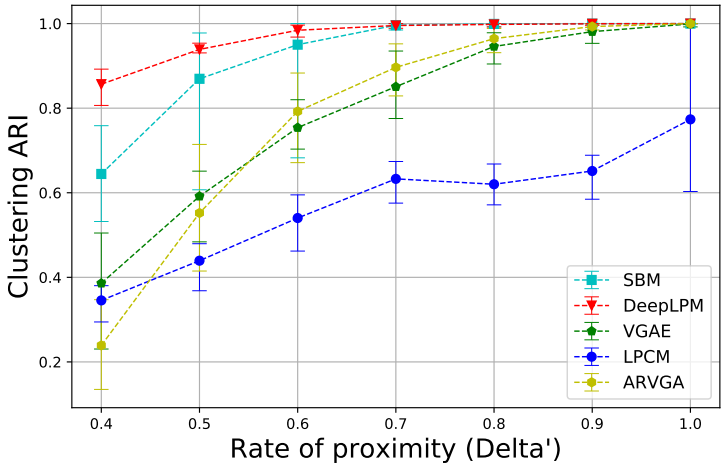

Figure 5: Clustering ARI with different proximity rates $\delta^{'}$ in Sc.B.

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

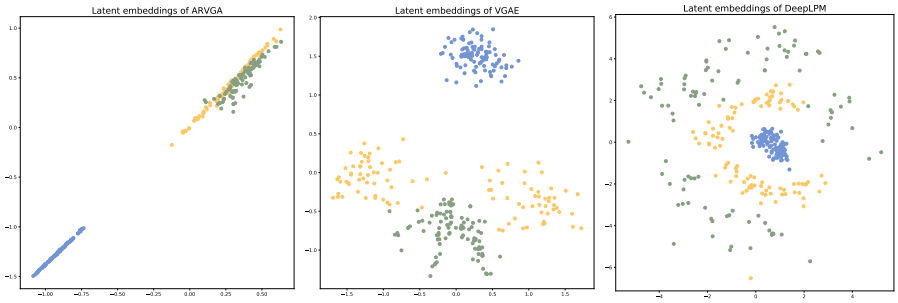

Figure 6: From left to right: embeddings learned by ARVGA, VGAE and DeepLPM with latent dimension equal to 2 in Sc.C.

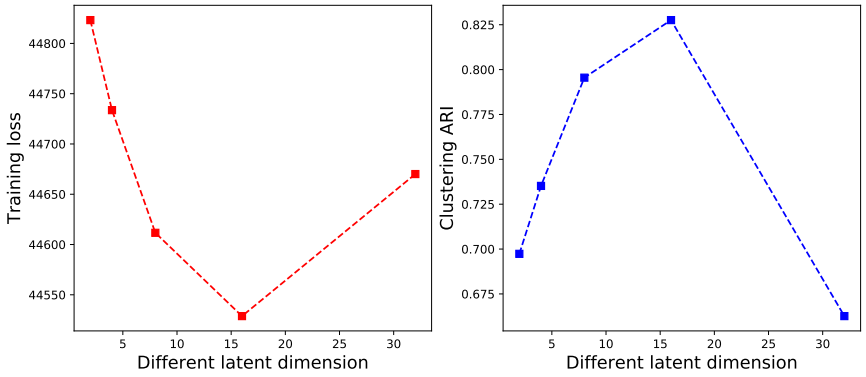

Figure 7: Averaged training loss (-ELBO) and clustering ARI with different latent dimensions on 50 networks based on scenario B.

selecting both the intrinsic dimension of the latent space and the number of groups appropriately.

Figure 7 shows the clustering ARI and training loss (-ELBO) 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 ELBO when $P = 16$, which is also associated with the highest ARI.

Similarly, by varying the number of clusters from 2 to 6, Figure 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,

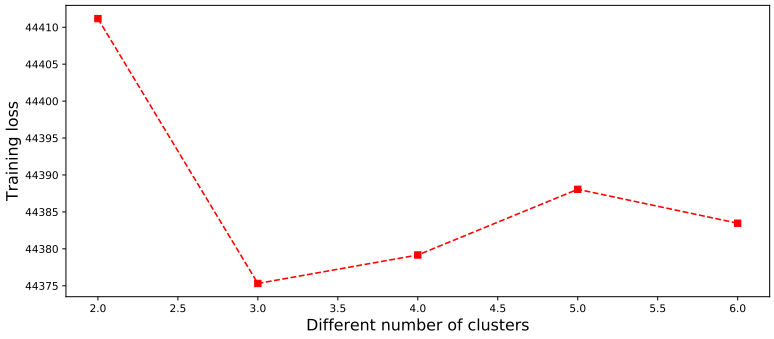

Figure 8: Averaged training loss (-ELBO) with different number of clusters on 50 synthetic data in scenario B.

thus recovering the actual value of K for the simulation setting.

5. Analysis of a medieval network

5.1. Dataset

As an illustration of the practical use of DeepLPM, it is fit here on a real-world dataset coming from historical science. Thus, 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 9 shows a visualisation of the network highlighting the importance of the temporality in the relationships.

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,

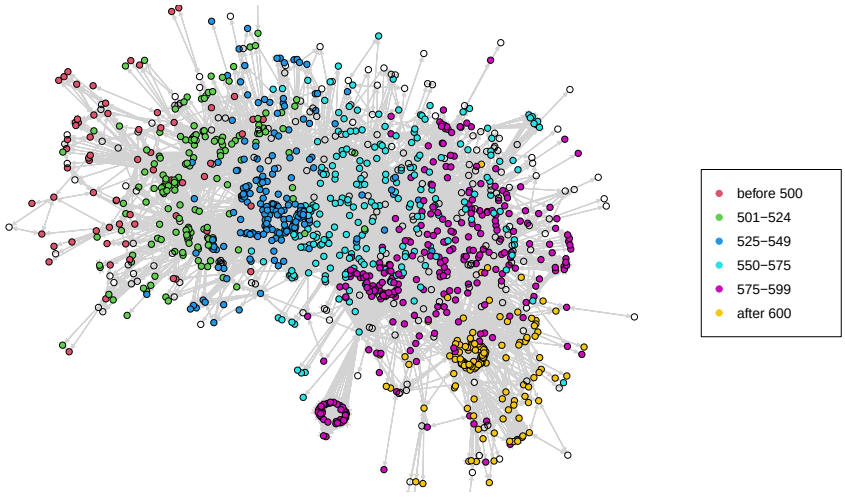

Figure 9: Visualisation of the ecclesiastical network, highlighting the temporality of the relationships.

-1 otherwise; $Y_{ij}^{(3)} = 1$ if i and j held a similar position (noble, ecclesiastical or other), -1 otherwise.

5.2. Results without covariates

We first analyze hereafter the clustering results without taking into account the covariate information Y. DeepLPM was run on this network for different number of groups, ranging between 2 and 10, and a fixed number of dimension ($P = 16$) for the latent space. When we ignore the covariate information, the evolution of the training loss shows a clear minimum at $K = 9$ according to the number of groups. The visualisation of cluster partitions of 9 groups by DeepLPM without covariates is shown in Figure 10. We also provide the results considering the personal positions in each group in Figure 11. 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, in Figure 10, the groups #3 and #5 gather people who lived at different and not overlapping time periods.

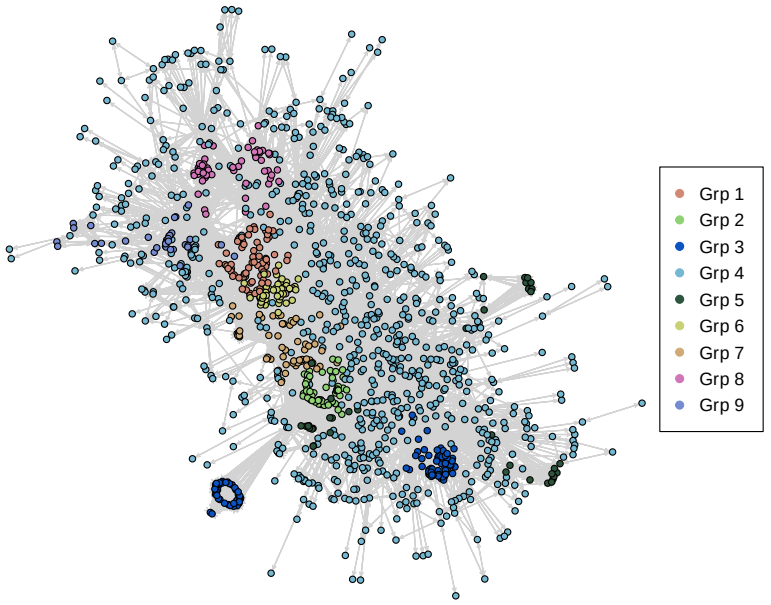

Figure 10: Cluster partitions without the covariate information on medieval data.

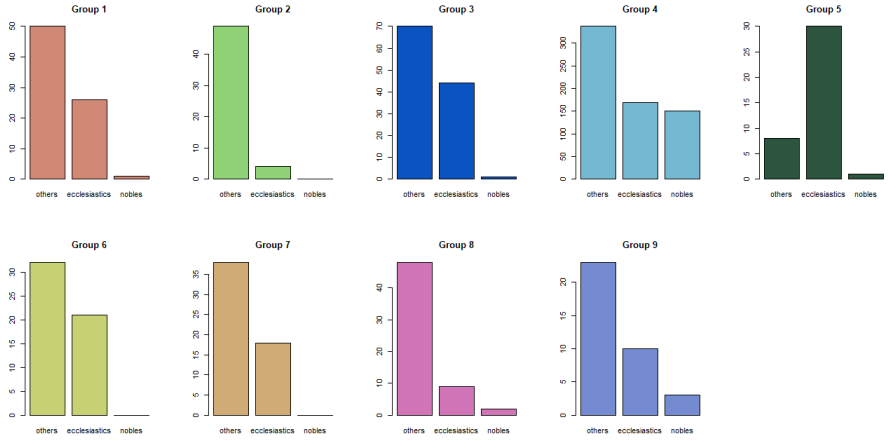

Figure 11: Partitions without covariates taking into account types in each group on medieval data.

5.3. Results with covariates

To show the effectiveness of the covariates, we integrated the 3-dimensional Y into DeepLPM and perform clustering on this network. The number of groups also varies

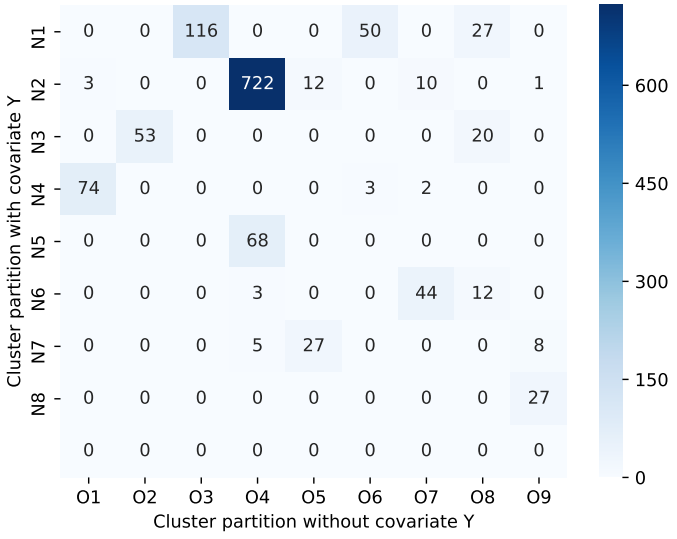

Figure 12: Confusion matrix between cluster partitions with and without covariate.

between 2 and 10, with a same intrinsic dimension equals to 16. When edge features are added to the model, the number of groups is estimated to be $K = 8$ with minimal loss. Thus, with this additional covariate information, 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 12. One can see that, by introducing the covariate, the cluster $N1$ gathered people from clusters $O3$, $O6$ and $O8$ together; then, it separates the individuals from $O8$ into $N1$, $N3$ and $N6$; and all the people from $N5$ were coming from $O4$. Therefore, the exploitation of the covariate information allows DeepLPM to focus on patterns that are not explained by the covariates alone.

Visualisation and analysis. Figures 13 shows the clustering visualisation obtained by DeepLPM for 8 groups with covariates. We can also see that DeepLPM was able to detect communities that played a similar role in the network at different periods. Indeed, the groups #1 (black), #3 (green), #5 (cyan) and #7 (yellow) gather people who

lived at different and not overlapping time periods. The red group #2 is particularly representative of this since it covers the whole period (480-614 of our era). Figure 14 shows that group #2 gathers individuals from all strata of society, who do not play a central role in the network.

We analyze the results by comparing the partitions without and with covariates in Figures 10 and 13. Firstly, we can see that the group #1 (black) in Figure 13 extracted some individuals from groups #6 and #8 in Figure 10 and kept people in the group #3. Combining with Figures 11 and 14, the group #1 in Figure 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. Moreover, the group #5 (cyan) in Figure 13 is specially separated from the big group #4 in Figure 10, which is made of a relatively significant proportion of nobles, in particular kings and queens, which suggests that this group was discussing political or nobility matters.

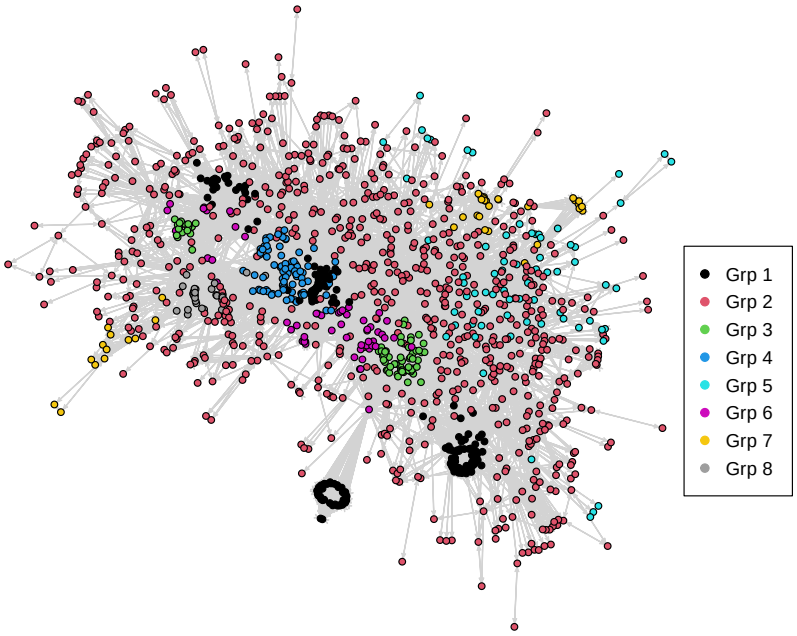

Figure 13: Visualisation of cluster partitions with covariates on medieval data.

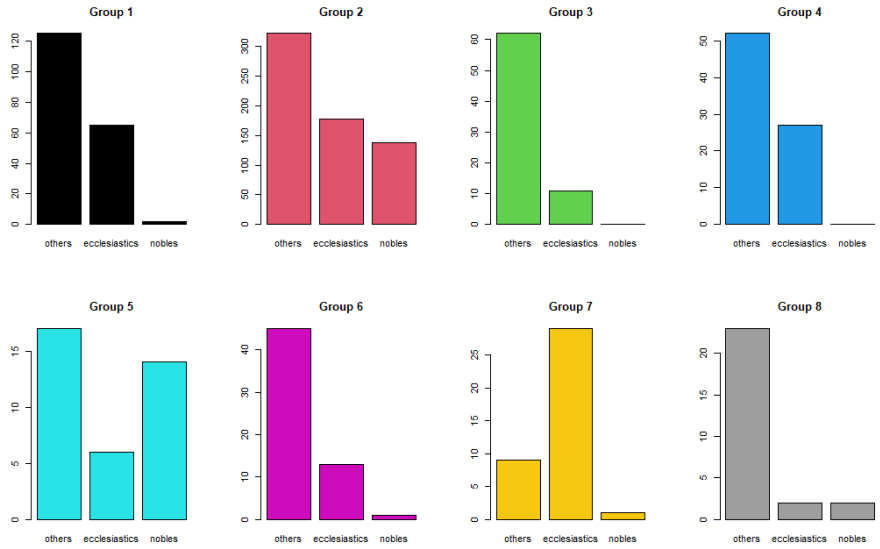

Figure 14: Partitions with covariates taking into account types in each group on medieval data.

6. Cora citation network

6.1. Dataset

The Cora dataset has been analysed 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.

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 classification. 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.

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 visualisation of the latent embeddings learned by DeepLPM is shown in Figure 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 16. In particular, groups #4 and #5 focus on subjects related to *theory* and *probabilistic methods*, groups #2 and #9 are mainly based on *neural networks* and *reinforcement learning*, respectively.

Without considering covariates, it is clear that most clusters, such as #1, #4, #6, #7, #8 and #9 contain mainly one category of papers, which coincides with the known supervised information. However we also see groups 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.

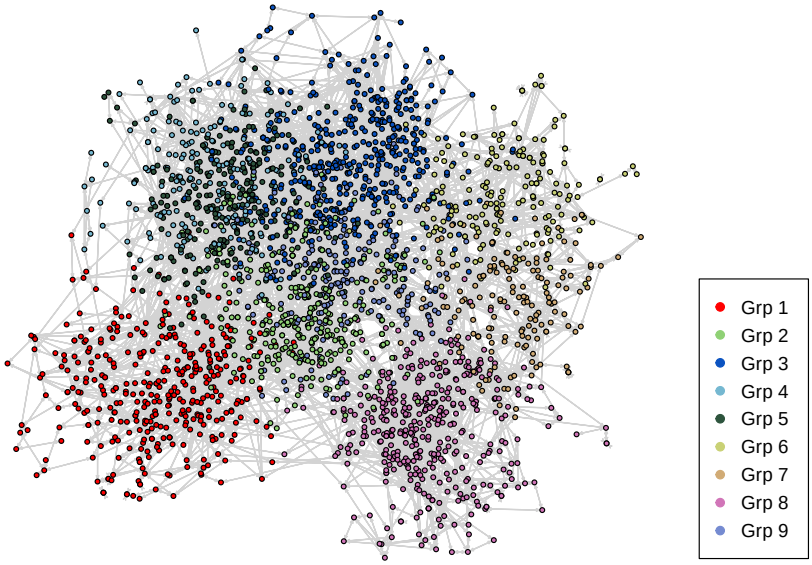

Figure 15: Visualisation of the clustered embeddings without covariates on Cora.

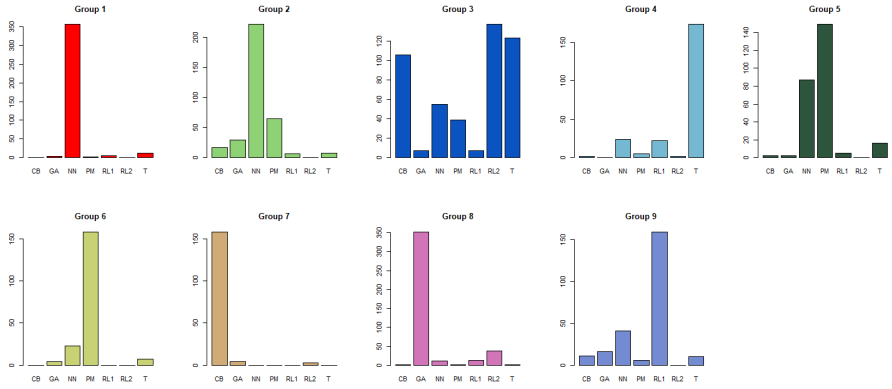

Figure 16: Partitions without covariates taking into account the classes in each group on Cora. Here CB: Case_Based, GA: Genetic_Algorithms, NN: Neural_Networks, PM: Probabilistic_Methods, RL1: Reinforcement_Learning,RL2: Rule_Learning, T: Theory.

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 17. As we can see, the cluster $N1$ extracted papers from 9 different clusters $O1$ 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 fusion facts here confirm that the addition of covariates helps to reveal hidden patterns behind supervised class information when performing clustering tasks.

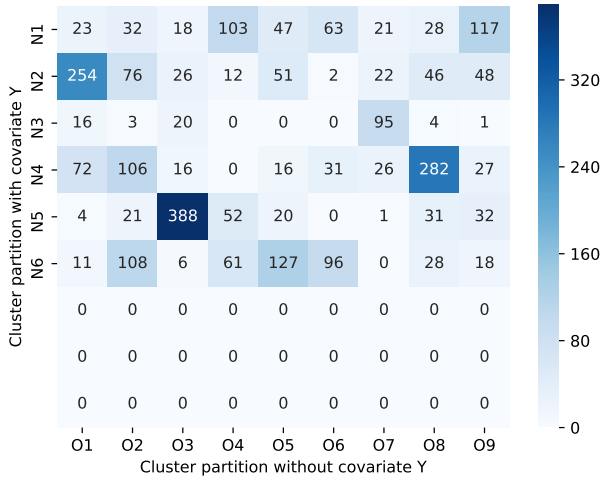

Figure 17: Confusion matrix between the estimated clusters with and without covariates.

Visualisation and analysis. The visualisation of the latent embeddings learned by DeepLPM is shown in Figure 18. Figure 19 shows the paper distributions when considering the class labels for six groups. In contrast to Figure 16 where each group contains

principally one or two different classes, it is clear that, due to the introduction of paper
380 labels as covariates new similarities between papers in different categories emerge.

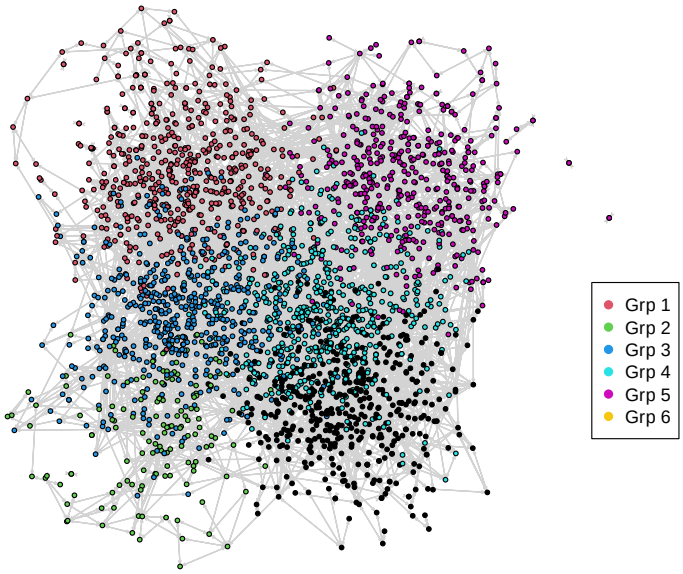

Figure 18: Visualisation of the clustered embeddings with covariates on Cora.

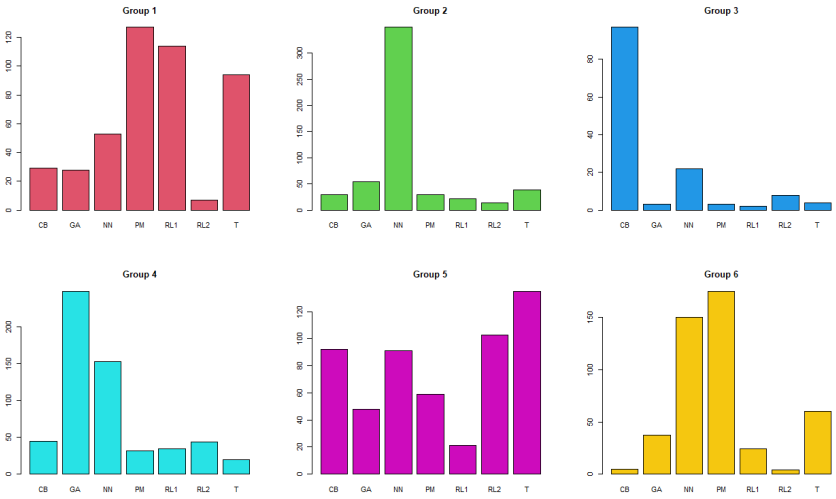

Figure 19: Partitions with covariates taking into account classes in each group on Cora. Here CB: Case_Based, GA: Genetic_Algorithms, NN: Neural_Networks, PM: Probabilistic_Methods, RL1: Reinforcement_Learning,RL2: Rule_Learning, T: Theory.

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 in Figure 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 3. According to paper titles, it can first be seen that group #1 (red) focuses on dynamic or temporal learning algorithms using probabilistic methods or reinforcement learning; group #2 (green) then discusses different aspects of neural networks, such as self-organization, adjusting or rules; in group #3 (blue), the papers are largely based on the analysis and development of case studies; next, group #4 (cyan) contains articles on applications of genetic algorithms and neural networks; while in group #5 (purple), papers consist of rule learning and inductive methods; finally, group #6 (yellow) typically involves statistical and machine learning models.

Interestingly, when looking at Figure 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.

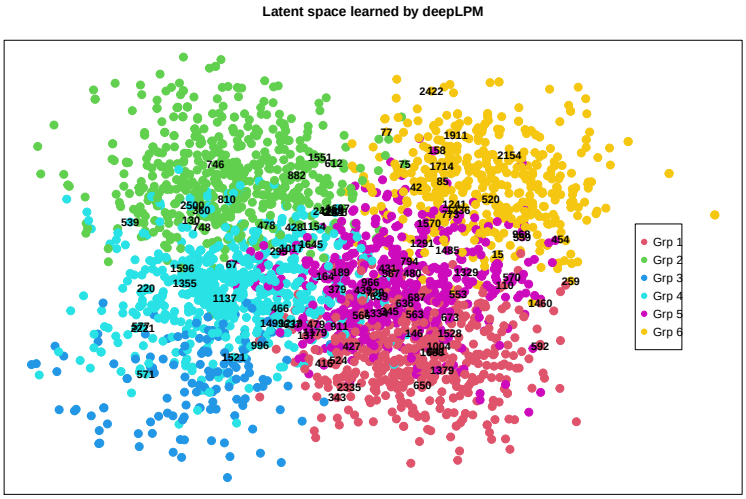

Figure 20: Learned hidden space (PCA compression), highlighting the nodes with degrees higher than 10.

Table 3: 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

Groups	Node ID	Paper title	Degree
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 bases 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.

7. Conclusion

We introduced DeepLPM to perform node clustering on network data in an end-to-end manner. 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. For future work, we are interested into analyzing textual edges by incorporating topic modeling.

Acknowledgements

This work has been supported by the French government, through the 3IA Côte d'Azur Investment in the Future Project managed by the National Research Agency (ANR) with the reference numbers ANR-19-P3IA-0002.

References

Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. (2008). Mixed membership stochastic blockmodels. *Journal of machine learning research*.

Aljalbout, E., Golkov, V., Siddiqui, Y., Strobel, M., and Cremers, D. (2018). Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*.

Bouveyron, C., Celeux, G., Murphy, T. B., and Raftery, A. E. (2019). *Model-based clustering and classification for data science: with applications in R*, volume 50. Cambridge University Press.

Bouveyron, C., Latouche, P., and Zreik, R. (2018). The stochastic topic block model for the clustering of vertices in networks with textual edges. *Statistics and Computing*, 28(1):11–31.

Corneli, M., Bouveyron, C., Latouche, P., and Rossi, F. (2019). The dynamic stochastic topic block model for dynamic networks with textual edges. *Statistics and Computing*, 29(4):677–695.

Dai, B., Wang, Y., Aston, J., Hua, G., and Wipf, D. (2017). Hidden talents of the variational autoencoder. *arXiv preprint arXiv:1706.05148*.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.

Handcock, M. S., Raftery, A. E., and Tantrum, J. M. (2007). Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354.

Hoff, P. D., Raftery, A. E., and Handcock, M. S. (2002). Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098.

Jernite, Y., Latouche, P., Bouveyron, C., Rivera, P., Jegou, L., and Lamassé, S. (2014). The random subgraph model for the analysis of an ecclesiastical network in merovingian gaul. *The Annals of Applied Statistics*, 8(1):377–405.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. (2016). Variational deep embedding: An unsupervised and generative approach to clustering. In *International Joint Conference on Artificial Intelligence (IJCAI-2017)*.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29:4743–4751.

Kingma, D. P. and Welling, M. (2014). Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, volume 19, page 121.

Kipf, T. N. and Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations (ICLR-17)*.

Kipf, T. N. and Welling, M. (2016b). Variational graph auto-encoders. In *NeurIPS Workshop on Bayesian Deep Learning (NeurIPS-16 BDL)*.

Latouche, P., Birmelé, E., and Ambroise, C. (2011). Overlapping stochastic block models with application to the french political blogosphere. *The Annals of Applied Statistics*, pages 309–336.

Lee, C. and Wilkinson, D. J. (2019). A review of stochastic block models and extensions for graph clustering. *Applied Network Science*, 4(1):1–50.

Mariadassou, M., Robin, S., and Vacher, C. (2010). Uncovering latent structure in valued graphs: a variational approach. *The Annals of Applied Statistics*, 4(2):715–742.

Matias, C. and Miele, V. (2017). 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.

Mehta, N., Duke, L. C., and Rai, P. (2019). Stochastic blockmodels meet graph neural networks. In *International Conference on Machine Learning*, pages 4466–4474. PMLR.

Nie, F., Zhu, W., and Li, X. (2017). Unsupervised large graph embedding. In *Thirty-first AAAI conference on artificial intelligence*.

Nowicki, K. and Snijders, T. A. B. (2001). Estimation and prediction for stochastic blockstructures. *Journal of the American statistical association*, 96(455):1077–1087.

Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., and Zhang, C. (2018). Adversarially regularized graph autoencoder for graph embedding. In *International Joint Conference on Artificial Intelligence (IJCAI-18)*, pages 2609–2615.

Raftery, A. E. (2017). Comment: Extending the latent position model for networks. *Journal of the American Statistical Association*, 112(520):1531–1534.

Schaeffer, S. E. (2007). Graph clustering. *Computer science review*, 1(1):27–64.

Snijders, T. A. (2011). Statistical models for social networks. *Annual review of sociology*, 37:131–153.

Tian, F., Gao, B., Cui, Q., Chen, E., and Liu, T.-Y. (2014). Learning deep representations for graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Wang, C., Pan, S., Long, G., Zhu, X., and Jiang, J. (2017). Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898.

Wang, Y. J. and Wong, G. Y. (1987). Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19.

Xie, J., Girshick, R., and Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR.

Xu, K. S. and Hero, A. O. (2014). Dynamic stochastic blockmodels for time-evolving social networks. *IEEE Journal of Selected Topics in Signal Processing*, 8(4):552–562.

Zhang, D., Yin, J., Zhu, X., and Zhang, C. (2018). Network representation learning: A survey. *IEEE transactions on Big Data*, 6(1):3–28.

Zhang, X., Liu, H., Li, Q., and Wu, X.-M. (2019). Attributed graph clustering via adaptive graph convolution. *arXiv preprint arXiv:1906.01210*.

Zhang, Z., Cui, P., and Zhu, W. (2020). Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*.