Deep latent position model for node clustering in graphs

Dingge Liang[1], Marco Corneli[1], Charles Bouveyron[1], and Pierre Latouche[2]

[1] Université Côte d'Azur, Inria, CNRS, Laboratoire J.A.Dieudonné, Maasai team, Nice, France
[2] Université Paris Cité, CNRS, Laboratoire MAP5, UMR 8145, Pairs, France

Abstract. With the significant increase of interactions between individuals through numeric means, the clustering of vertex in graphs has become a fundamental approach for analysing large and complex networks. We propose here the deep latent position model (DeepLPM), an end-to-end clustering approach which combines the widely used latent position model (LPM) for network analysis with a graph convolutional network (GCN) encoding strategy. Thus, DeepLPM can automatically assign each node to its group without using any additional algorithms and better preserves the network topology. Numerical experiments on simulated data and an application on the Cora citation network are conducted to demonstrate its effectiveness and interest in performing unsupervised clustering tasks.

Keywords: Network analysis · Graph clustering · Unsupervised learning.

1 Introduction and related work

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.

On the one hand, a long series of statistical methods have been developed to discover the underlying features in networks. The stochastic block model (SBM) [9] is widely used to detect communities or more general clusters of nodes. Based on SBM, many extensions looking for overlapping clusters have been proposed, such as MMSBM [1] and OSBM [7]. On the other hand, a different approach to model network data relies on latent position models (LPMs) [3]. Afterwards, LPCM [2] was developed to incorporate a clustering structure into LPM. Nevertheless, these models have a challenging inference procedure that primarily relies on MCMC and do not scale easily to large and complex networks.

From another aspect, deep learning based techniques have been intensively investigated in clustering. In this line of methods, VGAE [6] adopts a graph convolutional network (GCN) [5] encoder to produce nodes embeddings in the

latent space. By introducing adversarial learning into the generation process, ARVGA [10] enforced the latent representation to match a prior distribution. Lately, DGLFRM [8] combined OSBM with GCN by positing each node of the graph to have an embedding modeled by a Beta-Bernoulli process. All of the aforementioned methods employ inner-product-based decoders, whereas we argue that a different solution, accounting for the Euclidean distance between nodes in the latent space might be more suited. Additionally, these approaches 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.

In order to overcome the limitations of the methods listed above, while exploring their benefits, we introduce the deep latent position model (DeepLPM), 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.

2 Deep latent position model

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 $Y \in \mathbb{R}^{|E| \times D}$. 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.

Generative model As in LPM [3], we assume that each node $i = \{1, \cdots, N\}$ has an unknown position $z_i \in \mathbb{R}^P$ in a latent space. The probability of a link between two individuals is modeled as a function of the distance between their latent positions. The generative process is as follows. First, each node is assigned to a cluster via a random variable c_i encoding its cluster 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, a latent embedding 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. Finally, the probability of a connection between nodes i and j is modeled through a Bernoulli random variable related to the

distance between 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 and σ is the logistic sigmoid function and y_{ij} is the covariate of the edge connecting i with j.

3 Model inference

By denoting $\Theta = \{\pi, \mu_k, \sigma_k^2, \alpha, \beta\}$ the set of the model parameters introduced so far, we rely on a variational approach to approximate the intractable integrated log-likelihood

$$\log p(A|\Theta) = \mathcal{L}(q(Z, C); \Theta) + D_{KL}(q(Z, C)||p(Z, C|A, \Theta)), \tag{5}$$

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

$$q(Z, C) = q(Z)q(C) = \prod_{i=1}^{N} q(z_i)q(c_i). \tag{6}$$

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

$$q(z_i) = \mathcal{N}(z_i; \tilde{\mu}_\phi(\overline{A})_i, \tilde{\sigma}_\phi^2(\overline{A})_i I_P), \tag{7}$$

where $\tilde{\mu}_\phi(\cdot) : \mathbb{R}^{N\times N} \mapsto \mathbb{R}^{N\times P}$ (respectively $\tilde{\sigma}_\phi^2(\cdot) : \mathbb{R}^{N\times N} \mapsto \mathbb{R}^{+N}$) is the function mapping the normalized adjacency $\overline{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ into the matrix of the variational means (vector of the standard deviations), parametrized by the two-layer GCN *encoder* g_ϕ.

Finally, a standard assumption is made for variational cluster probabilities

$$q(C) = \prod_{i=1}^{N} \mathcal{M}(c_i; 1, \gamma_i), \quad \text{with} \quad \sum_{k=1}^{K} \gamma_{ik} = 1, \tag{8}$$

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

Thanks to Equations (6)-(7)-(8), the evidence lower bound (ELBO) can be further developed as

$$\begin{aligned}
\mathcal{L} &= \int_Z \sum_C q(Z,C) \log \frac{p(A|Z,\alpha,\beta)p(Z|C,\mu_k,\sigma_k^2)p(C|\pi)dZ}{q(Z,C)} \\
&= \mathbb{E}\left[\log p(A|Z,\alpha,\beta)\right] + \mathbb{E}\left[\log p(Z|C,\mu_k,\sigma_k^2\right] + \mathbb{E}\left[\log p(C|\pi)\right] - \mathbb{E}\left[\log q(Z|A)\right] - \mathbb{E}\left[\log q(C)\right] \\
&= \mathbb{E}\left[\log p(A|Z,\alpha,\beta)\right] + \mathbb{E}\left[\log \frac{p(Z|C,\mu_k,\sigma_k^2)}{q(Z)}\right] + \mathbb{E}\left[\log \frac{p(C|\pi)}{q(C)}\right] \\
&= \mathbb{E}\left[\sum_{i\neq j} A_{ij}\log\eta_{ij} + (1-A_{ij})\log(1-\eta_{ij})\right] - \sum_{i=1}^{N}\sum_{k=1}^{K}\gamma_{ik} D_{KL}(\mathcal{N}(\tilde{\mu}_\phi(\overline{A})_i, \tilde{\sigma}_\phi^2(\overline{A})_i I_P)||\mathcal{N}(\mu_k,\sigma_k^2 I_P)) \\
&\quad + \sum_{i=1}^{N}\sum_{k=1}^{K}\gamma_{ik}\log(\frac{\pi_k}{\gamma_{ik}}),
\end{aligned} \tag{9}$$

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 $q(\cdot)$. The pseudo code of the optimization process is 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
while $\mathcal{L}$ increases **do**
  $\tilde{\mu}_\phi, \tilde{\sigma}_\phi^2 = \text{GCN}(A)$
  update γ_{ik}, π_k, μ_k and σ_k^2 by calculating derivations ▷ explicit optimization
  calculate the training loss (negative ELBO) $-\mathcal{L}$
  update neural net parameters ϕ, α and β via SGD ▷ implicit optimization
Output: reconstructed graph $\hat{A}$, cluster probability matrix $\hat{\gamma}$

4 Numerical experiments

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 LPCM, SBM and from circle data. By varying the values of parameters δ and δ' 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.

Benchmark study We benchmark DeepLPM with SBM [9], LPCM [2], VGAE [6] and ARVGA [10] 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) is set to 0. For each situation, we generated ten different networks and calculated the averaged adjusted rand index (ARI) [4].

First, focusing on scenario A demonstrated in Figure 2, we can see that although the networks are simulated according to the LPCM, it does not exhibit the best performance. 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 the highest ARI and a small variance in all situations. Second, considering scenario B in Figure 3, 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. 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 and outperforms SBM when the value of $\delta^{'}$ is less than 0.7. 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 shown in Table 1. However, Figure 3 shows the embeddings learned by ARVGA, VGAE and DeepLPM with latent dimension equal to 2 in scenario C. It can be seen that only DeepLPM better preserves the network topology.

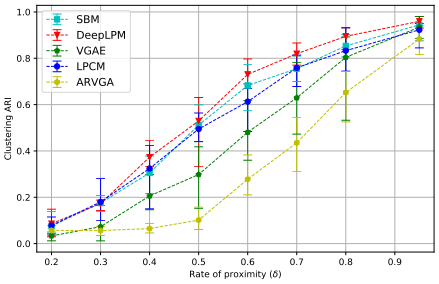

Fig. 2. Clustering ARI with different proximity rates in Sc.A.

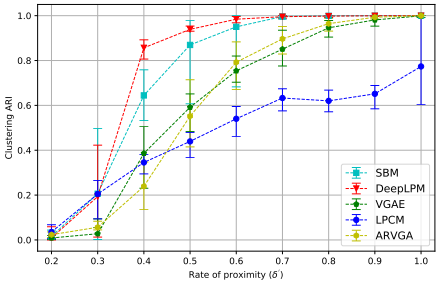

Fig. 3. Clustering ARI with different proximity rates in Sc.B.

Table 1. Clustering ARI with different proximity rates in Sc.C.

Method	SBM	LPCM	VGAE	ARVGA	DeepLPM
Sc.C	0.443±0.00	0.415±0.20	0.610±0.03	0.631±0.04	0.625±0.03

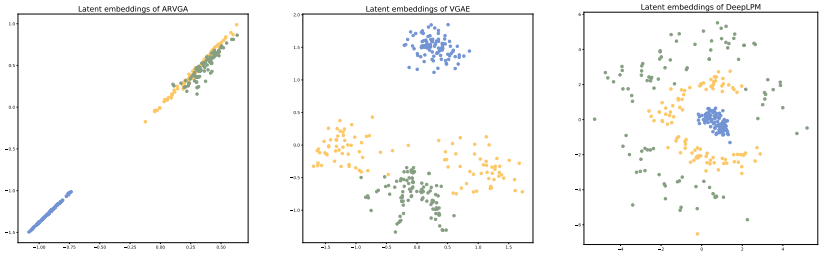

Fig. 3. Embeddings learned by ARVGA, VGAE and DeepLPM in Sc.C.

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 5 shows the averaged training loss (-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 6 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.

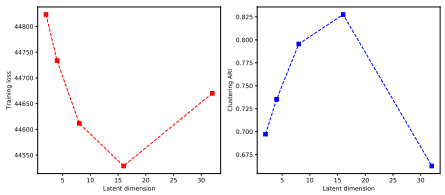

Fig. 5. Averaged training loss (-ELBO) and ARI with different latent dimensions on 50 networks based on scenario B.

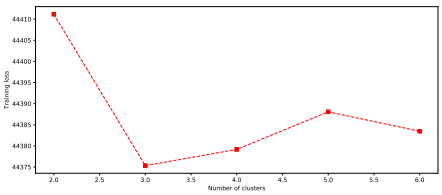

Fig. 6. Averaged training loss (-ELBO) with different number of clusters on 50 synthetic data in scenario B.

5 Analysis of Cora network

The Cora dataset contains 2,708 scientific publications classified in seven classes and consists of 5,429 citation links. Most related works 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 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.

The model selection was 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 training loss, the number of groups was estimated to be $K = 6$ with a clear minimum. Figure 6 shows the paper distributions when considering the class labels for six groups. In contrast to the fact that each group contains only

one defined class, it is clear that new similarities between papers in different categories emerge as a result of the addition of paper labels as covariates.

Next, to better understand the clustering results, we plotted the latent positions learned by DeepLPM using PCA in Figure 7, highlighting nodes with degrees higher than 10. Those papers are more often cited by other papers and can be more representative. Interestingly, when looking at this figure 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.

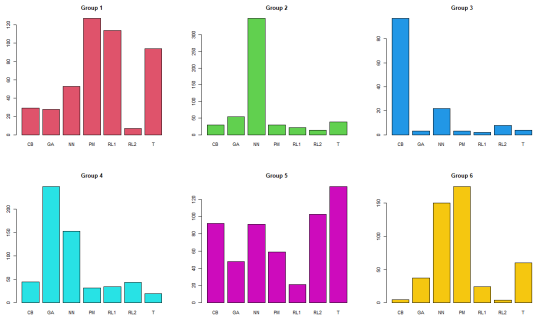

Fig. 6. Partitions with covariates taking into account classes in each group on Cora. Each group now contains a variety of categories that represent the hidden patterns discovered through the addition of covariates.

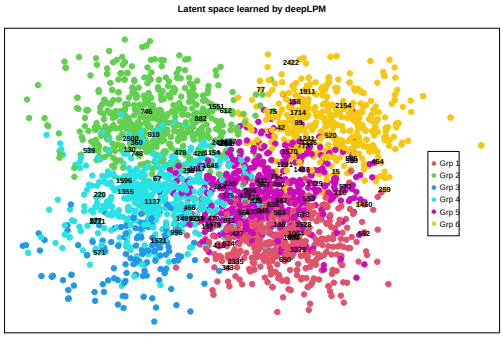

Fig. 7. Learned hidden space (PCA compression on the first two principal components), highlighting the nodes with degrees higher than 10.

Furthermore, based on the publications ID, we selected several articles with relatively large degree from each group and analyzed the information. For instance, according to paper titles, we find that group #1 (red) focuses on dynamic or temporal learning algorithms using probabilistic methods or reinforcement learning; in group #3 (blue), papers are largely based on the analysis and development of case studies; then, group #4 (cyan) contains articles on applications of genetic algorithms and neural networks; group #6 (yellow) typically involves statistical and machine learning models, etc.

Finally, we emphasize that, unlike most related works that consider supervised class labels as clusters in unsupervised learning, we encode this information into edge features and estimate the number of clusters via model selection, which aids in the discovery of new node similarities hidden behind the supervised information, as demonstrated by the results.

6 Conclusion

We introduced DeepLPM to perform node clustering in an end-to-end manner by integrating the GCN encoder with the LPM-based decoder. Numerical experiments show that DeepLPM outperforms state-of-the-art methods. Moreover, real-world application on a scientific citation network was also proposed to illustrate the interest of the methodology for unsupervised analysis.

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 number ANR-19-P3IA-0002.

References

1. Airoldi, E.M., Blei, D.M., Fienberg, S.E., Xing, E.P.: Mixed membership stochastic blockmodels. Journal of machine learning research (2008)
2. Handcock, M.S., Raftery, A.E., Tantrum, J.M.: Model-based clustering for social networks. Journal of the Royal Statistical Society: Series A (Statistics in Society) **170**(2), 301–354 (2007)
3. Hoff, P.D., Raftery, A.E., Handcock, M.S.: Latent space approaches to social network analysis. Journal of the american Statistical association **97**(460), 1090–1098 (2002)
4. Hubert, L., Arabie, P.: Comparing partitions. Journal of classification **2**(1), 193–218 (1985)
5. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations (ICLR-17) (2016)
6. Kipf, T.N., Welling, M.: Variational graph auto-encoders. In: NeurIPS Workshop on Bayesian Deep Learning (NeurIPS-16 BDL) (2016)
7. Latouche, P., Birmelé, E., Ambroise, C.: Overlapping stochastic block models with application to the french political blogosphere. The Annals of Applied Statistics pp. 309–336 (2011)
8. Mehta, N., Duke, L.C., Rai, P.: Stochastic blockmodels meet graph neural networks. In: International Conference on Machine Learning. pp. 4466–4474. PMLR (2019)
9. Nowicki, K., Snijders, T.A.B.: Estimation and prediction for stochastic blockstructures. Journal of the American statistical association **96**(455), 1077–1087 (2001)
10. Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., Zhang, C.: Adversarially regularized graph autoencoder for graph embedding. In: International Joint Conference on Artificial Intelligence (IJCAI-18). pp. 2609–2615 (2018)