Community Embeddings with Bayesian Gaussian Mixture Model and Variational Inference

Anton I. N. Begehr
Graduate School of Business
National Research University Higher School of Economics
Moscow, Russia
a.begehr@fu-berlin.de

Prof. Dr. Petr Panfilov
Graduate School of Business
National Research University Higher School of Economics
Moscow, Russia
ppanfilov@hse.ru

***Abstract*—Graphs, such as social networks, emerge naturally from various real-world situations. Recently, graph embedding methods have gained traction in data science research. The graph and community embedding algorithm ComE aims to preserve first-, second- and higher-order proximity. ComE requires prior knowledge of the number of communities K. In this paper, ComE is extended to utilize a Bayesian Gaussian mixture model with variational inference for learning community embeddings (ComE BGMM+VI), similar to ComE+. ComE BGMM+VI takes K as the maximum number of communities and drops components through the trade-off hyperparameter weight concentration prior. The advantage of ComE BGMM+VI over the non-Bayesian ComE for an unknown number of communities K is shown for the small Karate club dataset and explored for the larger DBLP dataset.**

***Index Terms*—graph, embedding, community embedding, ComE, Bayesian, variational inference, Gaussian mixture, expectation maximization**

I. Introduction

Graphs, such as social networks, knowledge graphs, content-rating graphs, and communication networks, emerge naturally from various real-world situations. Analyzing these graphs leads to findings and understanding of the underlying structures, coherences, and dependencies. Recently, methods for embedding graph's nodes into lower-dimensional Euclidean spaces, called graph embeddings, have gained traction in multiple areas of data science research [1].

Community Embeddings, in addition to embedding a graph's nodes through first- and second-order proximity, also preserve higher-order proximity by embedding clusters present in the graph data. The graph and community embedding algorithm ComE aims to preserve first-, second- and higher-order proximity by embedding a graph's nodes and communities[2]. ComE requires prior knowledge of the number of communities K. In this paper, ComE is extended to utilizing a Bayesian Gaussian mixture model with variational inference for learning community embeddings (ComE BGMM+VI), similar to ComE+ published by Cavallari et al. in 2019 [3]. ComE BGMM+VI takes K as the maximum number of communities and drops components through a trade-off hyperparameter.

The reported study was partially supported by RFBR grant №20-07-00958. The paper was prepared within the framework of the HSE University Project Group Competition 2020-2022.

The recent 2017 graph embeddings algorithm ComE is extended similarily to the 2019 ComE+ by taking a Bayesian approach. The open-source code for the Bayesian approach to ComE's community embedding is published on GitHub[1] and serves as a contribution to community embedding research [4].

The original ComE paper *Learning Community Embedding with Community Detection and Node Embedding on Graphs* by Cavallari et al. and the ComE+ paper *Embedding Both Finite and Infinite Communities on Graphs* by Cavallari et al. in combination with the ComE source code have provided the basis and architecture for the community embeddings utilized in this work [2, 3, 5].

Multiple surveys and articles on graph embeddings were consulted to build a full picture of the current state of graph embedding research. Especially the 2018 survey *Graph Embedding Techniques, Applications, and Performance: A Survey* by Goyal and Ferrara and the 2020 paper *On Proximity and Structural Role-based Embeddings in Networks: Misconceptions, Techniques, and Applications* by Rossi et al. have proven to be primary resources for understanding the current landscape of graph embedding research [1, 6].

On part of comparing the Bayesian Gaussian mixture model with variational inference to the Gaussian mixture model with expectation-maximization, the 2006 book *Pattern Recognition and Machine Learning (Information Science and Statistics)* by Bishop includes essential statistics and information science knowledge and explanations [7].

II. Graph Embedding

A graph embedding is a representation of a graph data structure in lower-dimensional space. Utilizing graph embeddings has recently gained traction in the research community for representing and analyzing graph data [1]. Possible applications of graph embeddings include node and graph classification, anomaly detection, link prediction, recommender systems, graph compression, and visualizations [6].

There are multiple advantages of graph embeddings over the original graph data-structure $G = (V, E)$:

1) **Machine learning algorithms on graphs are limited [8].**Only specific mathematics, statistics, and machine

[1] at https://github.com/abegehr/ComE_BGMM

learning algorithms can be applied to the specific graph data-structure G consisting of nodes V and edges E.

2) **Vector operations are simpler and faster than comparable graph operations [8].** A graph embedding is a representation of a graph in lower-dimensional space of dimension d, therefore nodes are assigned a feature vector of size d, which can then be operated on using vector operations.
3) **Embeddings are compressed representations [8].** A trivial feature vector for a node $v \in V$ of the graph $G = (V, E)$ is of length $|V|$ with an entry for each node $v_i \in V$ valued by the existing or non-existing, binary or weighted edge. When determining this trivial representation for all nodes $v \in V$, the adjacency matrix of size $|V| \times |V|$ is obtained. An embedding of the same graph $G = (V, E)$ is obtained through embedding G into a d-dimensional space. The resulting embedding is of size $|V| \times d$. An embedding is performed into a lower-dimensional space, therefore $d \ll |V|$.

Popular graph embedding algorithms include DeepWalk, introduced by Perozzi et al. in their 2014 paper *DeepWalk*, and Node2Vec, introduced by Grover and Leskovec in their 2016 paper *node2vec: Scalable Feature Learning for Networks* [9, 10]. Both DeepWalk and Node2Vec demonstrate competitive results for the use-case of multi-label classification and are promising options for graph embeddings.

III. Community Embedding

Recently, a graph embedding algorithm was altered and extended to include the concept of communities. Communities refer to groups of nodes that have high inner connectivity and low outer connectivity to other communities [2]. The proximity of nodes hereby is extended to include the concept of higher-order proximity, in addition to first-order and second-order-proximity [2].

Cavallari et al. developed the Community Embedding algorithm ComE in their 2017 paper *Learning Community Embedding with Community Detection and Node Embedding on Graphs* [2].

A. ComE Algorithm

ComE consists of three parts: node embedding, community detection, and community embedding. A closed-loop relationship between node embedding, community detection, and community embedding is indicated by Cavallari et al.. The closed-loop is exploited in an iterative algorithm optimizing the node and community embeddings.

The ComE algorithm is sketched out as follows:

1) Sample the graph by executing random walks.
2) Initialize the node embedding Φ and context embedding Φ' with the node embedding algorithm DeepWalk[9] with random walks.
3) Fit a Gaussian mixture model (GMM) representing communities to (Φ, Φ') using the expectation-maximization (EM) algorithm, while keeping (Φ, Φ') fixed. Yield the parameters of the GMM as the community embedding: mixed community membership Π, Gaussian means Ψ, and Gaussian covariances Σ.
4) Use stochastic decent with first-, second-, and higher-order proximity optimization functions to adjust (Φ, Φ') to the Gaussian mixture model, keeping the GMM's parameters (Π, Ψ, Σ) fixed. Yield the next node embedding and context embedding (Φ, Φ') updated by first-, second-, and higher-order proximity.
5) Repeat steps 3 and 4 at will. Step 3 represents community detection and embedding. Step 4 represents updating the node embedding.

The formulas and concepts used in this chapter are borrowed from the original ComE paper.[2] For a pseudocode implementation including more detail, see algorithm 1 of the original ComE paper [2]. Cavallari published an implementation of the ComE algorithm on GitHub[2].

B. ComE Hyperparameters

ComE requires multiple hyperparameters to be set prior to computation. The following table lists the hyperparameters, a description for each, and the notation used in literature:

TABLE I: Hyperparameters used by Cavallari's implementation of ComE [5].

parameter	notation	description
number_walks	γ	number of random walks for each node
walk_length	ℓ	length of each walk
representation_size	D	dimensionality of the embedding
num_workers		number of threads
num_iter		number of overall iterations
com_n_init		number of initializations to run on the communtiy embedding model (GMM or BGMM)
reg_covar		regularization coefficient for ensuring positive covariance
batch_size		size of the batch
window_size	ζ	windows size used to compute the context embedding
negative	m	number of negative samples
lr		learning rate
alpha	α	Trade-off parameter for context embedding
beta	β	Trade-off parameter for community embedding
down_sampling		perform down sampling of common nodes
communities	K	number of communities

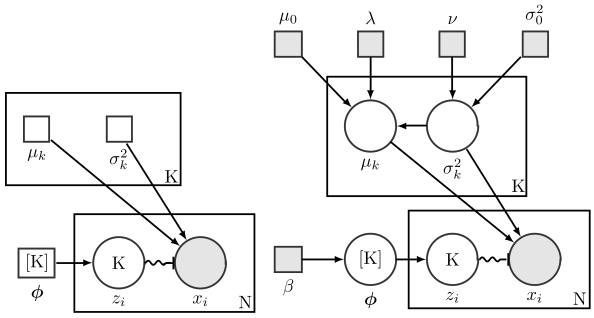
C. Non-Bayesian and Bayesian Gaussian mixture models

Figure 1 compares a non-Bayesian and a Bayesian GMM side-by-side in plate notation: Squares indicate fixed parameters and circles indicate random variables.

The Bayesian Gaussian mixture model assumes all parameters of the Gaussian mixture model (number of communities K, means μ_k, and covariances σ_k) are themselves generated by some distribution. This is represented in the plate diagrams in Figure 1.

[2]at https://github.com/andompesta/ComE

[3]By Benwing - Created using LaTeX, TikZ, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=18934624



(a) Non-Bayesian (b) Bayesian

Fig. 1: Gaussian mixture models in plate notation[3].

A Bayesian Gaussian mixture model (BGMM) with variational inference (VI) offers a solution to the issue of introducing the number of communities K as a fixed parameter, by taking advantage of Bayesian methods.

A BGMM with VI tends to utilize some components heavily while eliminating other components, if the provided dataset suggests so, by applying a tradeoff parameter. Therefore, the parameter K represents a maximum of communities and not the final number of communities like for GMMs.

D. Algorithm ComE BGMM+VI

The algorithm can stay mostly unchanged to the algorithm sketched out in Section III-A. Only steps 3 and 4 need to be adjusted to handle community detection and embedding and node embeddings using a Bayesian Gaussian mixture model (BGMM) with variational inference (VI) instead of a non-Bayesian Gaussian mixture model (GMM) with expectation-maximization (EM).

A Bayesian Gaussian mixture model, just like a non-Bayesian Gaussian mixture model, produces mixed community membership Π, which means Ψ, and covariances Σ for communities based on node embeddings Φ. Therefore, the inputs and outputs of step 3 stay the same.

In their 2019 paper *Embedding Both Finite and Infinite Communities on Graphs*, Cavallari et al. present their version of ComE with an infinite Bayesian-approach to community detection and embedding. As of the time of writing of this paper, no source code for ComE+ was published. While ComE+ and ComE BGMM+VI both take a Bayesian approach to community embedding, ComE+ redefined the optimization function for higher-order proximity based on the Bayesian Gaussian mixture model, while ComE BGMM+VI simply uses a BGMM for community embedding and detection and leaves the node embedding step unchanged.

The implementation of ComE BGMM+VI[4] presented and used in this paper, utilizes sklearn's implementation of the Bayesian Gaussian mixture model and variational inference references at *sklearn.mixture.BayesianGaussianMixture*.[11, 12]

[4]ComE BGMM+VI is published at https://github.com/abegehr/ComE_BGMM [4].

E. Hyperparameters BGMM

Switching from the original ComE's GMM to a Bayesian GMM, will require one additional parameter: the BGMM's weight concentration prior, or simply concentration.

The BGMM's concentration is a trade-off parameter that directly influences the number of components utilized for community embedding. A higher concentration parameter leads to more components active. A lower concentration parameter leads to fewer components active.

ComE with BGMM and VI has the same hyperparameters as ComE with GMM and EM, as presented in Section III-B, and in addition also the following:

TABLE II: Hyperparameter added to ComE for supporting a BGMM for community embedding [4].

parameter	notation	description
weight_concentration_prior	Γ	Bayesian GMM's weight concentration prior

References

[1] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, p. 78–94, 7 2018. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2018.03.022

[2] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ser. CIKM '17. Association for Computing Machinery, 2017, p. 377–386. [Online]. Available: https://doi.org/10.1145/3132847.3132925

[3] S. Cavallari, E. Cambria, H. Cai, K. C. Chang, and V. W. Zheng, "Embedding both finite and infinite communities on graphs [application notes]," *IEEE Computational Intelligence Magazine*, vol. 14, no. 3, pp. 39–50, 2019.

[4] A. Begehr. (2020) abegehr/ComE_BGMM. [Online]. Available: https://github.com/abegehr/ComE_BGMM

[5] S. Cavallari. (2017) andompesta/ComE. [Online]. Available: https://github.com/andompesta/ComE

[6] R. A. Rossi, D. Jin, S. Kim, N. K. Ahmed, D. Koutra, and J. B. Lee, "On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications," in *Transactions on Knowledge Discovery from Data (TKDD)*, 2020, p. 36.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[8] P. Godec. (2018) Graph embeddings — the summary. [Online]. Available: https://towardsdatascience.com/graph-embeddings-the-summary-cc6075aba007

[9] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk," *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining -*

KDD '14, 2014. [Online]. Available: http://dx.doi.org/10.1145/2623330.2623732

[10] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.