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.**

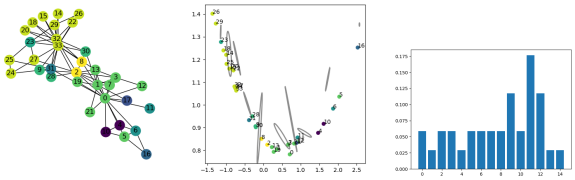
***Index Terms*—graph, embedding, community embedding, ComE, Bayesian, variational inference, Gaussian mixture, expectation maximization**

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].

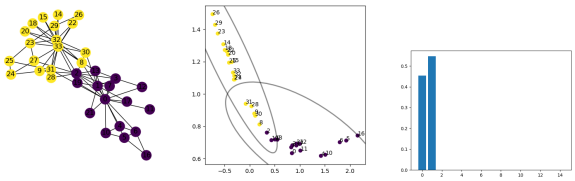
Community Embeddings, in addition to embedding a graph's nodes through first- and second-order proximity, also preserve higher-order proximity by embedding clusters present in the graph data. The graph and community embedding algorithm ComE aims to preserve first-, second- and higher-order proximity 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.



(a) Graph (b) Embeddings (c) Weights

Fig. 1: ComE with GMM, $K = 15$



(a) Graph (b) Embeddings (c) Weights

Fig. 2: ComE with BGMM, $K = 15$

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 source code is written in Python and tested with Python 3.6.10. For node and context embedding, ComE's training_sdg_inner Cython binary is utilized [5, 6].

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, 6].

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

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

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, 7].

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 [8].

During implementation, data transformation, testing, and plotting the following libraries and frameworks have proved invaluable: NumPy, Pandas, Matplotlib, NetworkX, and Jupyter Notebooks [9, 10, 11, 12, 13, 14].

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 [7].

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

1) **Machine learning algorithms on graphs are limited [15]**.Only specific mathematics, statistics, and machine 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 [15].** 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 [15].** 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* [16, 17]. 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[16] 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:

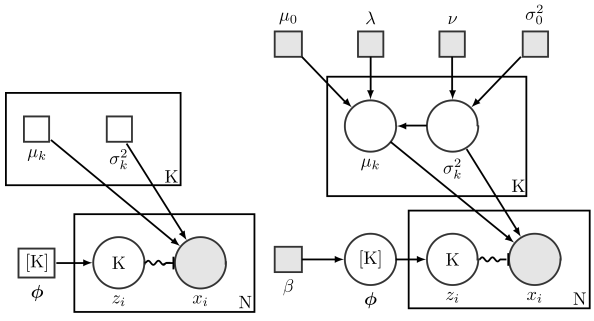
[2]at https://github.com/andompesta/ComE

TABLE I: Hyperparameters used by Cavallari's implementation of ComE [6].

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

The hyperparameters number_walks, walk_length, and representation_size are utilised in the node embedding step Node2Vec to achieve an initial node embedding optimizing for first and second oder proximity. reg_covar, batch_size, window_size, negative, and lr are utilized in the context embedding step. num_workers is a performance parameter which allows multithreading the node embedding or context embedding steps. num_iter, and com_n_init control the number of iterations to run over ComE's closed loop algorithm and how many random initialisations of the community embedding to try out, respectivelly. alpha and beta allow adjusting the weighting between context and community embedding. Communities K is the number of Gaussian generators as pertained to the Gaussian Mixture Model.

C. Non-Bayesian and Bayesian Gaussian mixture models



(a) Non-Bayesian (b) Bayesian

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

Figure 3 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 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*.[18, 19]

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.

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

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

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

IV. Visual Example

In this section community embeddings based on a standard Gaussian mixture model with expectation-maximization (GMM+EM) and community embeddings based on a Bayesian Gaussian mixture model with variational inference (BGMM+VI) are compared visually. Two-dimensional visualizations are computed on a small graph-dataset using multiple values for the hyperparameter number of communities K to show the advantage of BGMM+VI over GMM+EM under specific conditions.

A. Karate Club Dataset

Zachary's Karate Club graph is a popular graph dataset for visualizing graph algorithms, due to its small size [20].

The data was collected from a university karate club by Zachary as part of the 1977 paper *Zachary's Karate Club graph*. The small Karate Club graph consists of 34 nodes, each representing one person as part of the karate club. The presence of an edge represents a social tie among two members of the group. The absence of an edge represents no social tie among the two members of the group. Edges are bidirectional. *Zachary's Karate Club graph* provided a binary version (ZACHE) and a weighted version (ZACHC) of the graph. We will consider the binary version of the karate club graph.

The following hyperparameters are used for computing both ComE with GMM and BGMM results:

TABLE III: Hyperparameters used for the Karate Club dataset.

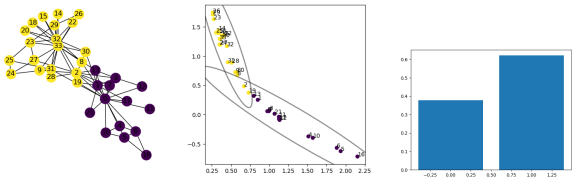
parameter	notation	value
number_walks	γ	10
walk_length	ℓ	80
representation_size	D	2
num_workers		10
num_iter		1
reg_covar		0.00001
batch_size		50
window_size	ζ	10
negative	m	5
lr		0.025
alpha	α	0.1
beta	β	0.1
down_sampling		0.0
communities	K	$[2, 3, 4, 15]$
weight_concentration_prior	Γ	10^{-5}

See Table I for a descriptions of the hyperparameters.

For each $K \in [2, 3, 4, 15]$ embeddings based on ComE with GMM and ComE with BGMM are computed. Each combination of a K and choice of a mixture model for community detection and embedding, three charts are obtained:

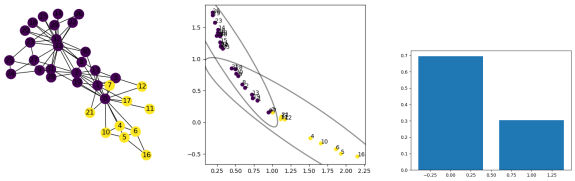
1) Graph: nodes positioned using the Fruchterman-Reingold force-directed (spring) algorithm and colored by community assignment.
2) Embeddings: obtained node and community embeddings. Ellipses represent communities and the underlying Gaussian component. Points colored by the community assignment represent node embeddings.
3) Weights: bar chart of weight for each community.

B. K=2



(a) Graph (b) Embeddings (c) Weights

Fig. 4: ComE with GMM, $K = 2$

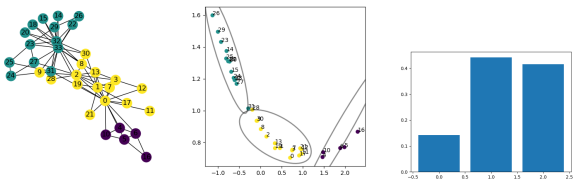


(a) Graph (b) Embeddings (c) Weights

Fig. 5: ComE with BGMM, $K = 2$

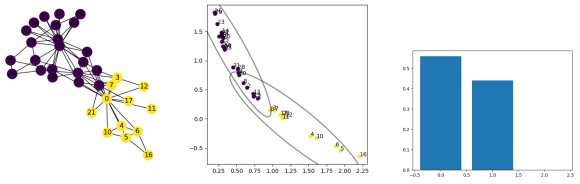
Figure 4 and Figure 5. Two is the intended number of communities. Setting $K = 2$ leads to similar results for both using a GMM and a BGMM for community detection and embedding. Although the labels are inverted, and the separation between both community assignments is slightly different, the general community assignment and structure are the same.

C. K=3



(a) Graph (b) Embeddings (c) Weights

Fig. 6: ComE with GMM, $K = 3$

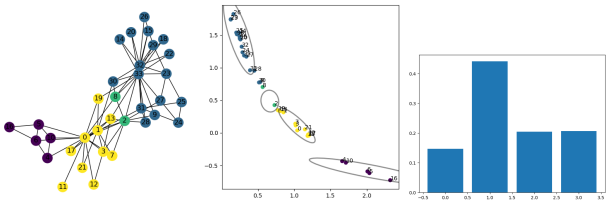


(a) Graph (b) Embeddings (c) Weights

Fig. 7: ComE with BGMM, $K = 3$

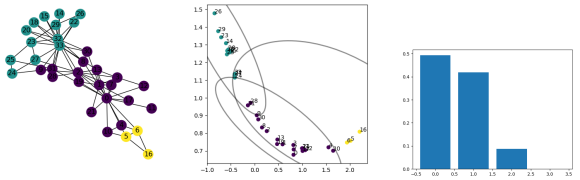
Figure 6 and Figure 7. Choosing $K = 3$ leads to worse results when using a GMM in comparison to the BGMM. The GMM utilizes the three components, while the BGMM drops the unnecessary third component and utilizes only the two necessary components.

D. K=4



(a) Graph (b) Embeddings (c) Weights

Fig. 8: ComE with GMM, $K = 4$



(a) Graph (b) Embeddings (c) Weights

Fig. 9: ComE with BGMM, $K = 4$

Figure 8 and Figure 9. With $K = 4$, the quality of embeddings based on the GMM further declines. The GMM utilizes all four communities. The result obtained by the BGMM also declines for this example, but wins over the GMM. The BGMM utilizes three communities.

E. K=15

Figure 1 and Figure 2. Providing $K = 15$ at initialization further underlines the potential advantage of BGMM over GMM. The GMM utilizes all 15 communities. It is clear that in this dataset of 34 nodes, no 15 communities are necessary for proper representation. The BGMM achieves a visually comparable quality to results at $K = 2$ and $K = 3$ by only utilizing 2 of the maximum 15 communities for representation.

F. Summary

As hypothesized, the community embedding algorithm ComE underperforms when the hyperparameter number of communities K is unknown. A Bayesian approach with variational inference similar to ComE+ is proposed to use the tradeoff parameter weight concentration before dropping unused communities if K is chosen too high.

The small Karate Club dataset, consisting only of 34 nodes, is used to test the hypothesis through a visual comparison experiment. The Karate Club graph is embedded into two-dimensional Euclidean space with ComE and ComE BGMM+VI for multiple values for the hyperparameter number of communities $K = [2, 3, 4, 15]$. The hypothesized advantage of BGMM over GMM for unknown numbers of communities K when embedding with ComE becomes clear. Even with $K = 15$, ComE BGMM+VI identifies the 2 communities, while the non-Bayesian ComE utilizes all 15 communities.

V. Quantitative Evaluation

In their original 2017 paper *Learning Community Embedding with Community Detection and Node Embedding on Graphs*, Cavallari et al. identify three evaluation tasks for experimentation: graph visualization, community detection and node classification [2]. Community detection and node classification can objectively be quantitatively evaluated, while graph visualization is subjectively evaluated.

In this section, community embeddings based on a non-Bayesian Gaussian Mixture Model with Expectation-maximization (GMM+EM), as presented with the ComE algorithm, and community embeddings based on a Bayesian Gaussian Mixture Model with Variational Inference (BGMM+VI), similar to the ComE+ algorithm, are evaluated quantitatively over multiple values for the number of communities K hyperparameter.

VI. DBLP Dataset

To evaluate ComE GMM+EM and ComE BGMM+VI on the tasks graph visualization, community detection, and node classification, the DBLP dataset is considered:

TABLE IV: Dataset used for quantitative evaluation.

graph $G = (V, E)$	#(node) $\lvert V \rvert$	#(edge) $\lvert E \rvert$	#(class) K	#(label)/node
DBLP	13,184	48,018	5	1

A. Hyperparameters

For comparability, the same hyperparameters are used for ComE GMM+EM and for the proposed ComE BGMM+VI, with the addition of the weight concentration prior hyperparameter for ComE BGMM+VI. The following hyperparameters were used for evaluating the DBLP dataset:

TABLE V: Hyperparameters used for the DBLP dataset. See Table I for descriptions of hyperparameters.

parameter	notation	value
number_walks	γ	10
walk_length	ℓ	80
representation_size	D	$[128, 2]$
num_workers		10
num_iter		3
com_n_init		10
reg_covar		0.00001
batch_size		50
window_size	ζ	10
negative	m	5
lr		0.025
alpha	α	0.1
beta	β	0.1
down_sampling		0.0
communities	K	$[2, 5, 10, 20]$
weight_concentration_prior	Γ	10^{-5}

See Table I for a descriptions of the hyperparameters.

In addition to varying the hyper-parameter number of communities K, the dimensionality of the representation (representation_size) is varied. The tasks are evaluated for two-dimensional and 128-dimensional embeddings to determine the effect of embedding size on the quality of results.

B. Community Detection

Community detection references the task of clustering nodes into communities and comparing the detected communities to the true communities as obtained from the true node classifications. The evaluation metrics are conductance and NMI.

1) Conductance: Figure 10 shows the conductance obtained from communities generated with ComE GMM+EM and ComE BGMM+VI on the DBLP graph with two- and 128-dimensional embeddings. Lower conductance is better.

As was expected, the conductance achieved with the true hyperparameter number of communities $K = 5$ is very similar for GMM and BGMM. The conductance achieved for $K = 2$ is similar for two-dimensional embeddings and higher for BGMM for 128-dimensional embeddings.

The conductances for $K = 10$ and $K = 20$ are different depending on the dimensionality of the embedding. For a large representation size of 128 dimensions (see Figure 10b), the conductance for the tested values for K where $K > 5$ increases in variability and decreases in overall quality in comparison to the smaller representation size of 2 dimensions (see Figure 10a), where the conductance stays mostly the same with $K = 10$ and $K = 20$.

Although ComE BGMM+VI produces better conductance than ComE GMM+EM for $K > 5$, a clear advantage of BGMM over GMM for community embeddings can not be identified from this experiment alone.

2) Normalized Mutual Information: Figure 11 shows the NMIs obtained from communities generated by ComE GMM+EM and ComE BGMM+VI with two- and 128-dimensional embeddings on the DBLP graph. Higher NMI is better.

ComE GMM+EM and ComE BGMM+VI perform very similar for the true hyperparameter number of communities $K = 5$ over both two-dimensional and 128-dimensional embeddings, as was to be expected. The controlling experiment $K = 2$ produces similar results for GMM and BGMM as well for both two- and 128-dimensional embeddings.

ComE GMM+EM achieves an NMI of 0.675 and ComE BGMM+VI achieves an NMI of 0.669 for the 128-dimensional embedding. Cavallari et al. state an NMI of over 0.7 for the ComE algorithm [2]. It is not clearly stated how many iterations are run specifically on DBLP in the original ComE paper, but it is stated that ComE converges within two or three iterations.[2] In this experiment three iterations were applied, as stated in Section VI-A.

With two-dimensional embeddings, an around 0.2 units smaller NMI is achieved for $K = K^* = 5$ than for the 128-dimensional embedding. It seems that the two-dimensional embedding is not capable to capture the complexity of the graph as well as the 128-dimensional embedding.

The potential advantage of a smaller embedding becomes clearer for larger values of $K > K^*$. For $K = 20$ the NMI for the 128-dimensional embedding is around 0.3, while the BGMM two-dimensional embedding manages to hold at 0.45.

It seems that for higher $K > 5$, ComE BGMM+VI can maintain higher NMI values for the two-dimensional embedding, while for the 128-dimensional embeddings, ComE BGMM+VI is not able to maintain the NMI score. Section VI-C explores the different behavior of ComE BGMM+VI on low- and higher-dimensional embeddings further in comparison to ComE GMM+EM.

C. Number of Communities and Dimensionality

Figure 12 shows in subfigures 12a, 12c, and 12e, that ComE BGMM+VI is able to identify $K = 5$ communities for the initial hyperparameter settings $K = [5, 10, 20]$ if a two-dimensional embedding is choosen. For the 128-dimensional embeddings in subfigures 12b, 12d, and 12f, ComE BGMM+VI is not able to drop communities to identify the $K = 5$ with initial values of $K = [5, 10, 20]$.

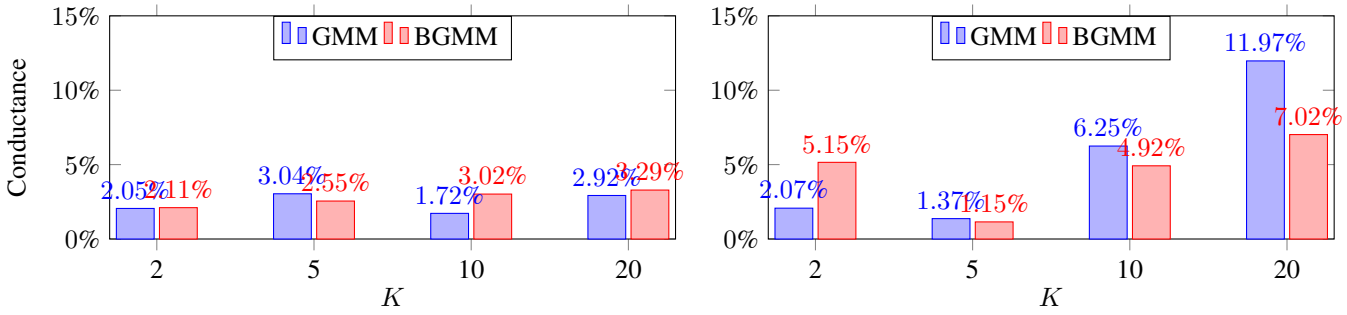
The finding from comparing community weights over the two- and 128-dimensional embedding of ComE BGMM+VI not dropping components for the 128-dimensional embeddings could explain the observation in Section VI-B2 of a better NMI for $D = 2$ when $K = 20$.

It is left for future work to consider different weight concentration priors and further explorations into the ability of BGMM and variational inference to unused drop components dependant of the dimensionality of input space.

VII. Results of Node Classification

Figure 13 shows the results of ComE GMM+EM and ComE BGMM+VI on the task node classification compared over multiple values for the hyperparameter number of communities K. The evaluation metrics micro- and macro-F1 are used.

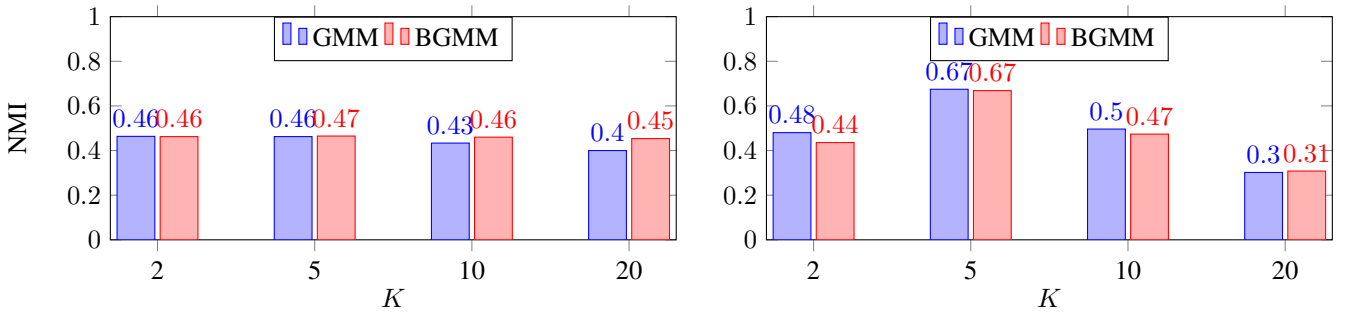
First, embeddings are generated from the whole graph. Then, node classification is done by a support vector machine



(a) two-dimensional embeddings.

(b) 128-dimensional embeddings.

Fig. 10: Conductance on DBLP dataset. Lower conductance is better.



(a) two-dimensional embeddings.

(b) 128-dimensional embeddings.

Fig. 11: NMI on DBLP dataset. Higher NMI is better.

(SVM), which is trained on 90% of the node embeddings and tested on the remaining 10% of node embeddings. The two- or 128-dimensional node embeddings are used as features for the support vector machine.

	#(communities) K	2	5	10	20
Micro-F1 (%)	ComE BGMM+VI	**67.1**	67.2	66.2	**67.0**
	ComE	66.9	**67.4**	**66.6**	66.5
Macro-F1 (%)	ComE BGMM+VI	**50.9**	49.6	48.6	**49.6**
	ComE	50.0	**50.0**	**49.3**	49.0

(a) two-dimensional embeddings.

	#(communities) K	2	5	10	20
Micro-F1 (%)	ComE BGMM+VI	92.6	**92.6**	**93.3**	92.3
	ComE	92.6	92.5	93.2	**92.7**
Macro-F1 (%)	ComE BGMM+VI	**92.7**	**92.4**	**93.0**	91.9
	ComE	92.6	92.3	92.9	**92.7**

(b) 128-dimensional embeddings.

Fig. 13: Micro- and Macro-F1 on the DBLP dataset with two- and 128-dimensional embeddings. Higher F1 values are better.

The Micro- and Macro-F1 scores attained by ComE GMM+EM and ComE BGMM+VI node embeddings through an SVM are very similar. No advantage of one over the other can be determined from the results. The similarity between results for node classification could be due to the specialties of the used DBLP dataset. To gain more clarity, further datasets should be evaluated in future work.

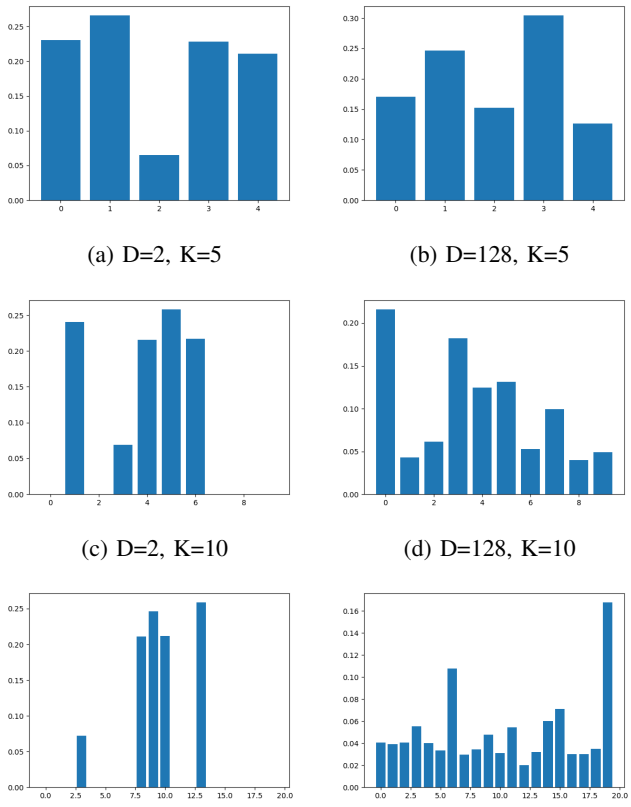
A. Summary

Although a better conductance was achieved for large K on the 128-dimensional embedding, a clear advantage of ComE BGMM+VI over ComE GMM+EM can not be identified for conductance over one experiment.

For NMI, ComE BGMM+VI seems to perform slightly better than ComE when choosing a two-dimensional embedding instead of a 128-dimensional embedding and a greater initial number of communities that the true number of communities: $K > 5$. It is hypothesized that this advantage comes from the ability of BGMM+VI to drop unused components in lower-dimensional space. It appears that BGMM+VI does not drop components as easy in higher-dimensional space.

Examining the weights of components produced by ComE BGMM+VI in two- and 128-dimensional space provides the insight, that ComE BGMM+VI produces the same distributions of 5 components' weights for $K = 5$, $K = 10$, and $K = 20$ in two-dimensional space. In 128-dimensional space, on the other hand, ComE BGMM+VI does not drop components and remains with weight distributions including all the components added at initialization.

In the experiment on the DBLP dataset, no advantage of using ComE BGMM+VI over GMM+EM was identified for node classification by SVM. It is reasonable to note, that node classification with micro- and macro-F1 of over 90% is



(a) D=2, K=5 (b) D=128, K=5

(c) D=2, K=10 (d) D=128, K=10

(e) D=2, K=20 (f) D=128, K=20

Fig. 12: Weights of communties generated by ComE BGMM+VI over hyperparameters representation sizes $D = [2, 128]$ and number of communities $K = [5, 10, 20]$.

achieved, with the 128-dimensional embedding, leaving little room for optimization.

A slight advantage of ComE BGMM+VI over ComE GMM+EM for community detection over the DBLP dataset under specific conditions was identified. No advantage for the task of node classification for ComE BGMM+VI over ComE GMM+EM could be identified for the DBLP dataset.

It was identified, that while the BGMM with variational inference can drop unused components when operating in low dimensional space and achieve the same component weight distributions over multiple initial values for K, in high dimensional space, the BGMM does not drop components.

VIII. Conclusion

ComE BGMM+VI, an extension of the community embedding algorithm ComE is presented in this paper. ComE BGMM+VI utilizes a Bayesian Gaussian mixture model instead of a non-Bayesian Gaussian mixture model for community embedding and detection. The Bayesian community embedding algorithm ComE BGMM+VI is compared to ComE GMM+EM visually on the Karate Club graph dataset and quantitatively evaluated on community detection and node classification the DBLP dataset.

Visually comparing ComE BGMM+VI to ComE GMM+EM clearly shows the advantage of ComE BGMM+VI over ComE GMM+EM for an unknown number of communities K and a low representation size D. ComE BGMM+VI is able to drop unused communities and reduce K to the number of communities present at the resolution captured at the chosen representation size D through the weight concentration prior hyperparameter trade-off parameter.

A slight edge of ComE BGMM+VI over the original ComE as identified for community detection NMI in a low-dimensional embedding for K larger than the true number of communities. More notably, ComE BGMM+VI captures 5 communities in the DBLP dataset at representation size $D = 2$ for all tested values for the number of communities $K > 5$. The true number of communities in the DBLP dataset is 5. ComE with non-Bayesian Gaussian mixture model is not able to drop communities for $K > 5$, as seen in Figure 12.

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] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

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

[7] 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.

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

[9] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane,

J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[10] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.

[11] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[12] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[13] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[14] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87 – 90.

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

[16] 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

[17] 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.

[18] 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.

[19] 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.

[20] S. Pakin. (1977) Zachary's Karate Club graph. [Online]. Available: https://networkx.github.io/documentation/stable/auto_examples/graph/plot_karate_club.html

[21] W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, pp. 452–473, 1977.