Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 173 (2020) 372–381

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on *Smart Sustainable Intelligent Computing and Applications* under ICITETM2020

Community detection in Networks using Graph Embedding

Rimjhim Agrawal, Md. Arquam, Anurag Singh

Department of Computer Science & Engineering,
National Institute of Technology Delhi,
New Delhi-110040, India

Abstract

Graphs are used to depict real-world scenarios as they have a wide variety such as online social networks, data and communication networks, word co-occurrence networks, biological networks, transport networks etc. Rigorous analysis of graphs produces the insight of graphs and yields the more profound knowledge of the social structure, language, and different communication patterns through data analysis of each type and provide the interaction between them. Many applications like node classification, link prediction, clustering can be inferred by using the graph. Several proposals are formulated to perform graph analysis. Recently, the most popular methods used by researchers are the presentation of graph nodes in a vector space by transforming the graph information into a low-dimensional manifold with maximum preservation of graph properties. In this course of the research, researchers apply the embedding technique to classify nodes of a graph based on the intrinsic property of nodes. However, most graph analytic methods require excessive space and computation cost. Therefore, a new method of embedding is needed for graph analysis. In this paper, various existing embedding techniques are simulated and comparatively discussed with different datasets. Based on this embedding, communities are created on the real-world network data of the karate club. At last, some future direction of research is mentioned in terms of application scenarios.

Keywords: Node Embedding; Complex Network; Community Detection

1. Introduction

Nowadays, graph theory is used to explain the diverse range of real-world phenomena, e.g., diffusion of information in social networks, user interest networks in e-commerce, information graph, etc. By analysis of such type of graphs,

* Corresponding author. Tel.: +91-8173087476
E-mail address: rimjhimagrawal167@gmail.com

10.1016/j.procs.2020.06.044

researchers can extract the hidden information from the graph, and therefore, the study of such graphs has drawn notable attentiveness in the recent time. Rigorous and deep analysis of graph will provide loads of applications, like node categorization [1], clustering of nodes[2], prediction of links [3], node revival/proposal [4], etc. For instance, after examining the user interplay in an exceedingly social network (e.g. tweets, likes and comments on Twitter and Facebook), users can be categorized, discover new communities, friends, and forecast the friendship between two users.

Though researchers have smartly applied graph analytics techniques for extracting information from the graph, those methods face high computational and space cost, plenty of analysis is dedicated to conduct graph analytics expeditiously. Examples embrace the distributed graph method (e.g., GraphX [5], GraphLab [6]),and so on. Traditionally, in graph embedding, individual nodes are focused upon, which produce a vector as an output, for every node within the graph, in such a way that nodes closer in the graph are represented with the same vector representation in a low-dimensional manifold. Such embedding is shown with success in conserving the structure of the network and remarkably improves a variety of applications, inclusive of classification of nodes.

To analyze the online social networks, similar types of nodes based on attributes can be grouped in a community while the previous researches focused only on structural properties to find community in a network. The work includes a comparison of different embedding algorithm and their analysis on different datasets. Finally, the community is formed based on the node embedding approach by using the real-world dataset of the Karate Club. In this paper, first, the previous related work by node embedding is shown in section 2. Then the technical details of the community detection method using node embedding are presented in the next section, 3. Section 4 shows the simulation and comparison of different types of node embedding techniques with community detection. Finally, in section 5, the summary of the paper along with the future scope of node embedding is given.

2. Literature Survey

Graph analytics problem can be solved by the use of graph embedding in an efficient way that explores the hidden properties of nodes. The idea is to transform a graph into a less dimensional space, using graph embedding and maintaining the information of the graph. An efficient graph embedding algorithm is required to convert a graph in low dimensional vectors. Graphs can be of different types, such as homogeneous graphs, heterogeneous graphs, attribute graphs, etc. Therefore, graph embedding gives different outputs in different scenarios. Embedding graph gives a less-dimensional vector as output that represents the entire graph or a part of the graph. In the 2000s, graph embedding methods were devised, with the primary aim of reducing the high mobility of non-relational data by assuming that the data lies in low dimensional space. A similarity graph depends on coupled attribute similarity for non-relational high dimensional data attributes. Thereafter, each node of the graph is taken and embedded in a less-dimensional manifold where the disconnected nodes are kept far.

The problem of graph embedding is based on the intersection of two research problems, namely, graph analytics [7] and representation learning [8]. The primary aim of graph embedding is the representation of the graph in a low dimension. This embedding is done by preserving the structure of the graph. On the other side, analytics on the graph helps in mining the valuable information. Whereas representation learning derives data representation, which makes it easy to gather valuable information for constructing a classifier or predictor. [9].

Embedding into less dimensional manifold is not an easy task. Based on the problem, different kinds of embedding input leads to different graph embedding challenges. These inputs contain different information that has to be retained in the embedded space. For example, structural embedding requires preserving connections within the nodes. However, embedding a graph with a node label maintains the graph property based on the attribute along with other related information, and this should be considered during embedding. The embedding output is task-oriented. For example, node embedding, which is generally used, indicates that the corresponding similar nodes are close. This type of embedding provides functions related to a node such as clustering, node classification, etc. Moreover, sometimes, functions may be similar to the apathy of the graph, e.g., pairs of nodes, subgraphs, entire graphs. Therefore, the main objective for embedding is to find an embedding type of output which suits our area of interest.

3. Technical Detail of Various Node Embedding Technique

This section starts with the definition and some basic concepts of the graph embedding problem.

Definition 1: Let us consider a graph $G(V, E)$, where, V is the set of vertices or nodes and $E = \{e_{i,j}\}_{i \neq j}^{n}$ edges or links.

Definition 2: Given a graph G, a graph embedding can be defined as a function $f : v_i \rightarrow y_i \in R^d \ \forall i \in [n]$ such that $d \leq |V|$. The proxomity measures defined on the graph G is preserved by the given function.

The idea of graph embedding technique is to transform a graph G, in a d dimensional manifold, by considering the preservation of graph properties as much as possible. The graph property is defined by quantifying the proximity measures of the first order to high order proximity. For representing a graph in d dimensional space, each graph is reduced in vectors or set of vectors where each vector represents the embedded part of the graph. The part may be any vertex, edge or substructure[10].

3.1. Locally Linear Embedding(LLE)

Locally Linear Embedding (LLE) [11] assumes each vertex to be a linear combination in the embedding space. The combination is formed by the adjacent vertices. Let $W_{i,j}$ be the weighted adjacency matrix of the graph G, where, $W_{i,j}$ is used to represent the weight of the node j, then LLE is defined as,

$$Y_i \approx \sum_j (W_{i,j} Y_j) \ \ \forall i \in V \tag{1}$$

Hence, the embedding $Y^{N \times d}$, can be obtained by minimizing,

$$y^* = \sum_i |(Y_i - \sum_j W_{i,j} Y_j)|^2 \tag{2}$$

The degenerated solutions are removed to constrain the embedding variance, as $\frac{1}{N} Y^T Y = I$. Further, the translational invariance is removed. The embedding obtained, sums to zero: $\sum_i Y_i = 0$. Now, it is transformed into an eigenvalue so that the above optimization problem can be solved. The last eigenvectors $(d + 1)$, obtained by the sparse matrix $(I - W)^T (I - W)$ are taken and disposing of the ones which correspond to the lowest eigenvalue.

3.2. Graph Laplacian Eigenmaps

In Laplacian Eigenmaps [12], the two nodes embedding, are kept closer when the weight $W_{i,j}$ between respective nodes is high. Mainly, the following function is minimized to obtain the Laplacian Eigenmaps,

$$y^* = \frac{1}{2} \sum_{i \neq j} |y_i - y_j|^2 W_{i,j} = tr \ \ (y^T L y) \tag{3}$$

where $W_{i,j}$ is the defined weight between node v_i and v_j, $L = D - W$ is the graph Laplacian. D is the diagonal matrix where $D_{i,i} = \sum_{i \neq j} W_{i,j}$. The larger the value of $D_{i,i}$, the more valuable is y_i[13]. A constraint $y^T D y = I$ is usually applied on Eq. 3 in the embedding to remove scaling factors, if any.

3.3. Graph Factorization

As per our knowledge and given litrature, the first method of graph embedding is Graph Factorization(GF) [14] with complexity $O(|E|)$. In this method of the embedding, the graph's adjacency matrix is factorized, and then the following function is minimized,

$$f(Y, \lambda) = \frac{1}{2} \sum_{(i,j) \in E} (W_{ij} - < Y_i, Y_j >)^2 + \frac{\lambda}{2} \sum_i \|Y_i\|^2 \tag{4}$$

Where λ is called the regularization coefficient. From the above optimization, the sum is done only on observed edges. This approximation is used in the interest of scalability by introducing the solution with some noise. As it is known that the adjacency matrix can sometimes be non-positive semidefinite, the minimum value of the loss function can be greater than 0 even though the embedding dimensionality is $|V|$.

3.4. Hope

HOPE [15] was proposed by minimizing $\|S - Y_s Y_t^T\|_F^2$ with preserving higher-order proximity, where S represents the similarity matrix. Various similarity measures were experimented and investigated by applying, together with Katz similarity, Adamic-Adar similarity score, Common Neighbors score, and Rooted Page Rank. Each similarity measure is represented by $S = M_g^{-1} M_l$ by considering M_g and M_l as sparse. Sparseness enhances HOPE to utilize the Singular Value Decomposition (SVD) [16] to get the embedding in an effective way.

3.5. Community Detection

Due to the different variety of information, depending upon the topology of the network and node characteristics, it creates challenges to classify nodes in meaningful communities. Many researchers proposed the different types of models that combine the structural and attributed information[17] [18]. However, in the case of unweighted networks, all proposed methodology uses original network topology directly. Due to this, inherent community structures are ignored by assigning each connection with a similar value. Nodes are densely connected within a community, therefore, multiple densely-connected subgraphs exist in attributed graphs. Instinctively, edges that form densely connected subgraphs are much more similar to create a corresponding community than edges that connect separate subgraphs. So, direct use of original network topology directly causes indiscriminately penalizing node pairs, whether in densely-connected structures or not [19]. While nodes in a community are densely-connected, they should also share homogenous node attributes. By using the above methods, a model SCI combines original topology of the network and attributes of the node, to find the community with a unified objective function. However, SCI does not factorize the sparse node attributes matrix but treats it as the basis to fit the community membership [18], which could degrade the effectiveness of SCI due to the redundancy and noise in node attribute matrix.

A community structure embedding method to quantify the structural closeness of nodes is explained, according to their potential community membership similarities. Then, based on the similarity measurement, skip-gram network with negative-sampling(SGNS) [20] is used, to explore the network structure and depict the underlying community structures. Finally, the community structure embedding matrix is obtained, that encodes the inherent community structures. Now, start with a concise and reasonable observation that each linked pair of nodes should have a certain tendency to fall into the same community because nodes are densely connected in a community. Consider two nodes v_i and v_j in graph G and the product of community memberships $(U_{i:} U_{j:}^T) \geq 0$, our similarity measurement function $f(i, j) \in [0, 1]$ is designed, using the sigmoid function to measure the similarity of their community memberships [21] as follows,

$$f(i,j) = 2 * \sigma(U_{i:} U_{j:}^T) - 1 = 2 \times \left(\frac{1}{1+e^{-U_{i:} U_{j:}^T}}\right) - 1 \tag{5}$$

The choice of the sigmoid function is a bounded differentiable real function for all real input values. The community structure embedding is used to encode densely-connected subgraphs and also to explore inherent community structures. Also, it can be easily increased to all kinds of networks, e.g., undirected networks, directed networks, weighted networks and unweighted networks, which shows the broad applications of community structure embedding method. Community Structure Embedding Matrix (M) [22]. In this community structure embedding method tries to maximize,

$$\sigma(U_{i:} U_{j:}^T) \tag{6}$$

for connected nodes v_i and v_j , meanwhile minimizing $\sigma(U_{i:} U_{j:}^T)$ for a set of two randomly chosen nodes v_i and v_j. In real-world life, most of the large networks are very sparse, and a pair of randomly selected nodes are likely to be connected with a low probability. Thus, based on the skip-gram network with negative-sampling (SGNS)[20], a community structure embedding matrix $M^{n \times n} \in R^{n \times n}$ is applied with the given formulation [1],

$$M_{ij} = max\ (U_{i:} U_{j:}, 0) \tag{7}$$

In the attributed graph G, the objective of detecting community is to find K communities such that nodes in communities are highly connected and have similar attribute values. By including the objective functions proposed by [23]

for community structures embedding and node attributes respectively as 6 and 7, the unified objective function can be used for community detection embedding model as,

$$\min_{U\geq 0, C\geq 0} L(U,C) = \|T - UC\|_f^2 + \alpha \sum_i \|C_{:i}\|_1^2 + \beta \|M - UU^T\|_f^2 \quad (8)$$

Where β is used to balance the node attributes matrix T, and community structure embedding matrix M and β must be positive. It helps in the factorization of community structure embedding matrix M for determining communities.

4. Simulation and Result Analysis

The principal part of the evaluation includes a detailed comparison of Node Embedding on various datasets. Further it is tried to detect communities on the output obtained by the node embedding algorithms. The various node embedding method by using different datasets are simulated and the community on karate club data is found, by using a node embedding approach.

The simulation is run on the system having with the configuration shown in Table 1. Configuration detail is necessary because embedding methods are compared based on embedding time.

Table 1. System Specifications

Hardware Detail	Software Detail
Processor: I-7 7^{th} generation 2.4GHZ System Memory: 16GB RAM HDD: 1TB	OS: Ubuntu 18.04 Programming Language : Python 3.5

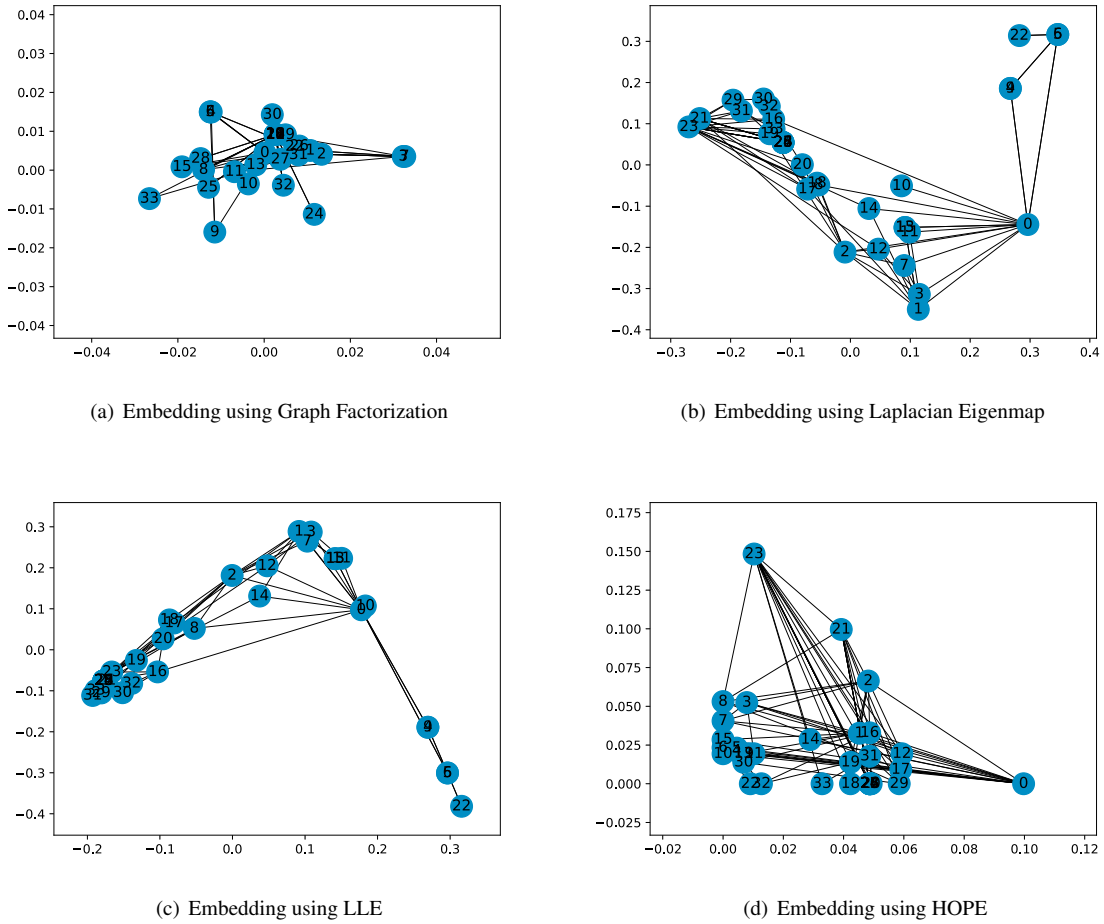
There are 4 types of dataset used in the simulation are mentioned in Table 2. Each dataset has different number of nodes and edges. Karate Club data has 34 nodes and 78 edges, while, MANN dataset has 45 nodes and 918 edges. Hamming Dataset has 64 nodes and 704 edges and 4227 nodes and 39484 edges are in BIO HS CL dataset.

Table 2. Statistics of the Datasets used

Dataset	No of Nodes	No of Edges
Karate Club	34	78
MANN	45	918
Hamming	64	704
BIO_HS_CL	4227	39484

Various graph embedding strategies of Zacharys karate club is shown in a 2D space in Fig.1(a)-(d). Similar data as input is provided to various strategies for node embedding. For every node, 10 sample paths are used. A negative sampling size is used and window size is set as 5. In HOPE, for each node, both proximity as first, as well as second-order proximity embedding, is considered. In Fig.1(a), GF is plotted to show the embedding of communities and keep leaf nodes distant away from other nodes. The embedding time of GF strategy is 26.345584 sec. LE and LLE are plotted in Fig.1(b)&(c) by grouping the nodes with high intracluster edges, while the community structure is preserved. Time taken to run the LE and LLE embedding strategy on Zacharys karate club is 0.204045 sec and 0.246332 sec respectively. In Fig.1(d), HOPE is plotted, which shows that embedding takes place only those nodes, whose Katz similarity is very low in the original graph and distant apart from each other. Embedding time of HOPE strategy is 0.157046 sec.

Similar graph embedding strategies is applied on MANN dataset shown in Fig.2(a)&(d). In Fig.2(a), GF is applied to show the embedding of communities by keeping leaf nodes distant away from other nodes. Embedding time of GF on MANN dataset is 13.817421 sec. Plot of LE and LLE are depicted in Fig.2(b)&(c) by grouping the nodes with high intracluster edges by conserving the community structure. Time taken by LE and LLE strategies are 0.020729 sec and 0.079954 sec. In Fig.2(d), HOPE is plotted, which shows that embedding takes place only those nodes,



(a) Embedding using Graph Factorization

(b) Embedding using Laplacian Eigenmap

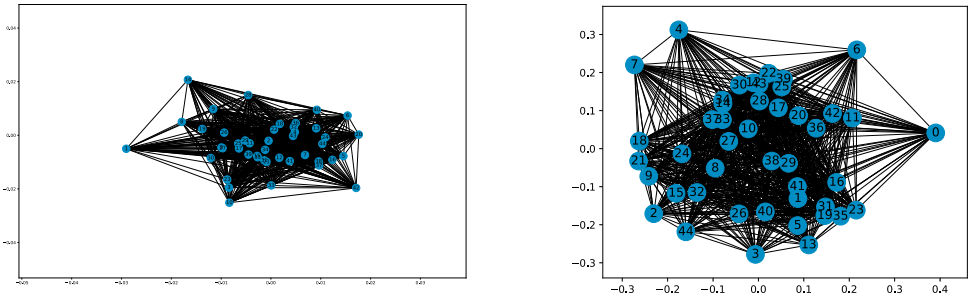
(c) Embedding using LLE

(d) Embedding using HOPE

Fig. 1. Embedding a graph into 2D space using Karate Club Dataset

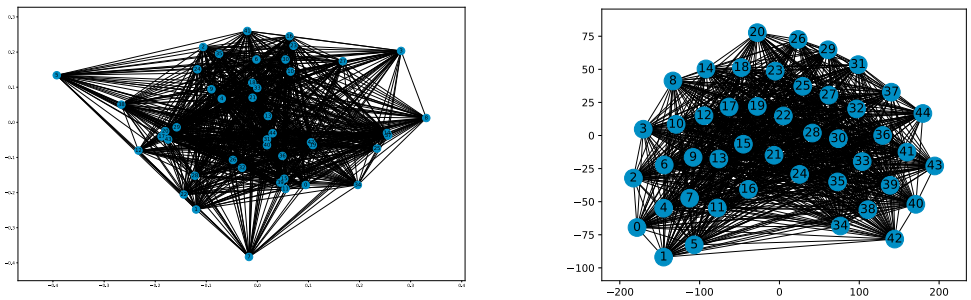
whose Katz similarity is very low in the original graph and distant apart from each other, while embedding time of HOPE is 0.017999 sec. All four graph embedding strategies mentioned above is applied on HAMMING dataset is shown in Fig.3(a)&(d). GF is plotted to show the embedding of communities by keeping leaf nodes distant away from other nodes in Fig.3(a). Increment in number of nodes increases the embedding time. Embedding time of GF on HAMMING dataset is 26.345584 sec. LE and LLE are plotted in Fig.3(b)&(c) by assembling the nodes with high intracluster edges, while the community structure is conserved. Embedding time of LE and LLE strategies on HAMMING dataset are 0.204045 sec and 0.246332 sec respectively In Fig.3(d), The HOPE is plotted, indicating that the embedding takes only those nodes whose Katz similarity is much lower in the original graph and distant from each other. The embedding time of HOPE is 0.157046 sec.

In this course, same set of graph embedding strategies are applied on BIO HS LC dataset to see the effect of embedding as shown in Fig.4(a)&(d). In Fig.4(a), GF is applied for embedding on BIO HS LC dataset to show the embedding of communities and keep leaf nodes distant away from other nodes. As number of nodes in BIO HS LC is much more than other dataset used in this work. Therefore, embedding time is also higher for GF strategy on BIO HS LC dataset as 26390.1268 sec. LE and LLE are also plotted in Fig.4(b)&(c) by teaming up the nodes with high intracluster edges, while the community structure is preserved. But embedding time taken by LE strategy is much lower than LLE strategy. Reason behind this is that LE strategy uses $d + 1$ lowest eigenvectors, while LLE uses first d eigenvectors of the graph Laplacian. Time of embedding of LE strategy is 1.334812 sec, while LLE strategy takes 22.189020 sec. HOPE



(a) Embedding using Graph Factorization

(b) Embedding using Laplacian Eigenmap



(c) Embedding using LLE

(d) Embedding using HOPE

Fig. 2. Embedding a graph into 2D space using MANN Dataset

strategy is applied on BIO HS LC dataset by considering the nodes having lower value of katz similarity and nodes with distant apart from each other. It is is plotted in Fig.4(d). Time of embedding of HOPE strategy is 6.390745 sec. Based on the number of nodes and network structure, embedding efficiency varies according to the embedding algorithm. Time taken by each algorithm with the various dataset is shown in Table 3.

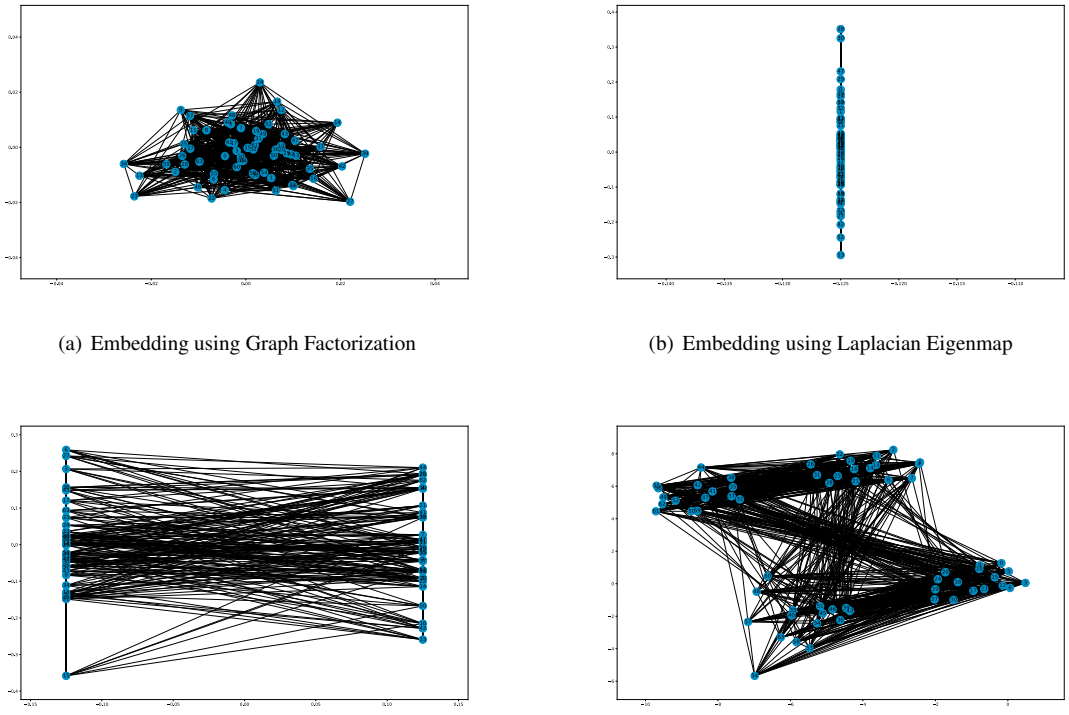
For all the embedding methods other datasets are used and result is plotted in Fig.2, 3, 4.

Based on the number of nodes and network structure, embedding efficiency varies according to the embedding algorithm. Time taken by each algorithm with the various dataset is shown in Table3.

Table 3. Embedding Time (in secs)

Dataset	Graph Factorization	HOPE	Laplacian	LLE
Karate Club	26.345584	0.157046	0.204045	0.246332
MANN	13.817421	0.017999	0.020729	0.079954
Hamming	26.345584	0.157046	0.204045	0.246332
BIO_HS_CL	26390.126897	6.390745	1.334812	22.189020

Table3 represents the performance of various graph embedding algorithms on various datasets in terms of embedding time(in secs). It is seen that the HOPE algorithm takes minimum time for node classification for the three dataset named Karate Club, MANN, Hamming, whereas LE takes the minimum time for BIO HS CL dataset. Graph Factorization takes the maximum time for all dataset.



(a) Embedding using Graph Factorization

(b) Embedding using Laplacian Eigenmap

(c) Embedding using LLE

(d) Embedding using HOPE

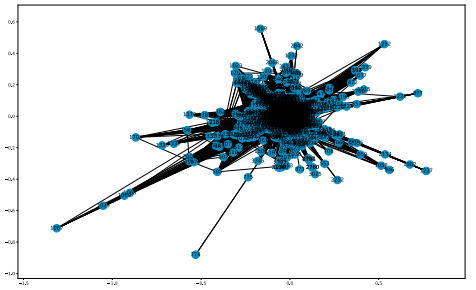
Fig. 3. Embedding a graph into 2D space using HAMMING Dataset

Also, the detected community based on embeddings is plotted, which yields two community, as shown in Fig.5. This community is created based on the node attribute. In this method, it is observed that some nodes overlap in both communities.

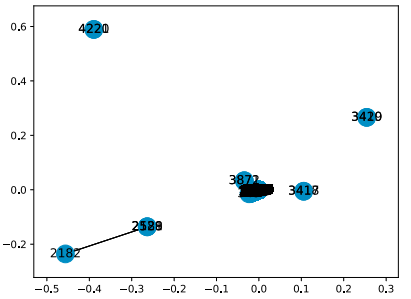
5. Conclusion and Future Scope

The paper describes four approaches of graph embedding, namely Graph Factorization, Locally Linear Embedding, Laplacian Eigen maps and HOPE. The various embedding methods were studied on different datasets. The main aim was to preserve structural and logical properties of the network. After the analysis, the performance of HOPE was found to be better. The performance largely depends on the structure of the network. If we increase the intracluster in the network then LE performs better than HOPE. Further, based on HOPE, community is found by applying node attribute on Karate Club dataset. Then, the embedding efficiency of the studied algorithm is analyzed, and it was found that HOPE performs better than others for the same datasets. There is two main research direction in the area of graph embedding, and one is the use of non-linear models, second is the use of evolving networks. Future work will be to explore these areas in graph embedding.

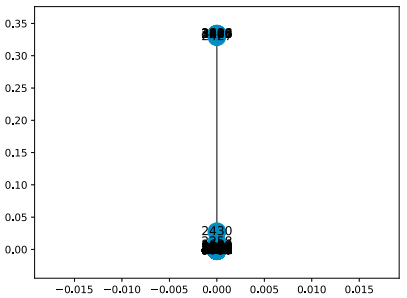
Apart from this, graph embedding can be applied to analyze the epidemic spreading[24][25] and rumor spreading[26] on the network by considering nodes intrinsic properties.



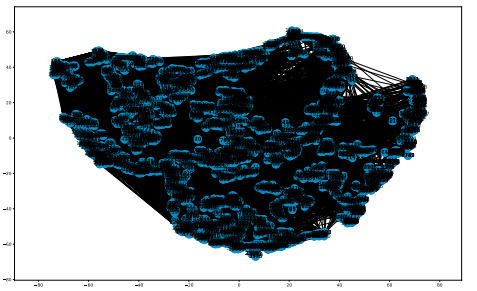
(a) Embedding using Graph Factorization



(b) Embedding using Laplacian Eigenmap



(c) Embedding using LLE



(d) Embedding using HOPE

Fig. 4. Embedding a graph into 2D space using *BIO_HS_LC* Dataset

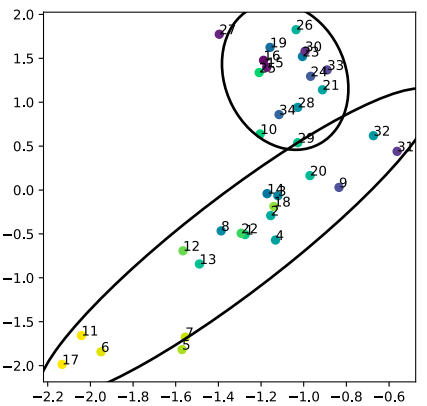


Fig. 5. Community Detection in Karate Club Dataset

References

[1] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[2] Feiping Nie, Wei Zhu, and Xuelong Li. Unsupervised large graph embedding. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[3] Xiaokai Wei, Linchuan Xu, Bokai Cao, and Philip S Yu. Cross view link prediction by learning noise-resilient representation consensus. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1611–1619. International World Wide Web Conferences Steering Committee, 2017.

[4] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. Scalable graph embedding for asymmetric proximity. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[5] Joseph E Gonzalez, Reynold S Xin, Ankur Dave, Daniel Crankshaw, Michael J Franklin, and Ion Stoica. Graphx: Graph processing in a distributed dataflow framework. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 599–613, 2014.

[6] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, 2012.

[7] Nadathur Satish, Narayanan Sundaram, Md Mostofa Ali Patwary, Jiwon Seo, Jongsoo Park, M Amber Hassaan, Shubho Sengupta, Zhaoming Yin, and Pradeep Dubey. Navigating the maze of graph analytics frameworks using massive graph datasets. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 979–990. ACM, 2014.

[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[9] Arif Mahmood, Michael Small, Somaya Ali Al-Maadeed, and Nasir Rajpoot. Using geodesic space density gradients for network community detection. *IEEE Transactions on Knowledge and Data Engineering*, 29(4):921–935, 2016.

[10] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[11] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[12] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[13] Xiaofei He and Partha Niyogi. Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160, 2004.

[14] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48. ACM, 2013.

[15] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114. ACM, 2016.

[16] Charles F Van Loan. Generalizing the singular value decomposition. *SIAM Journal on Numerical Analysis*, 13(1):76–83, 1976.

[17] Martin Atzmueller, Stephan Doerfel, and Folke Mitzlaff. Description-oriented community detection using exhaustive subgroup discovery. *Information Sciences*, 329:965–984, 2016.

[18] Xiao Wang, Di Jin, Xiaochun Cao, Liang Yang, and Weixiong Zhang. Semantic community identification in large attribute networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[19] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Homophily, structure, and content augmented network representation learning. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 609–618. IEEE, 2016.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[21] Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik 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*, pages 377–386. ACM, 2017.

[22] Hongyun Cai, Vincent W Zheng, Fanwei Zhu, Kevin Chen-Chuan Chang, and Zi Huang. From community detection to community profiling. *Proceedings of the VLDB Endowment*, 10(7):817–828, 2017.

[23] Ye Li, Chaofeng Sha, Xin Huang, and Yanchun Zhang. Community detection in attributed graphs: an embedding approach. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[24] Md Arquam, Anurag Singh, and Rajesh Sharma. Modelling and analysis of delayed sir model on complex network. In *International Conference on Complex Networks and their Applications*, pages 418–430. Springer, 2018.

[25] Md Arquam, Anurag Singh, and Hocine Cherifi. Integrating environmental temperature conditions into the sir model for vector-borne diseases. In *International Conference on Complex Networks and Their Applications*, pages 412–424. Springer, 2019.

[26] Anurag Singh and Yatindra Nath Singh. Nonlinear spread of rumor and inoculation strategies in the nodes with degree dependent tie strength in complex networks. *arXiv preprint arXiv:1208.6063*, 2012.