A Comparative Study of Network Embedding Based on Matrix Factorization

Xin Liu(✉) and Kyoung-Sook Kim

Artificial Intelligence Research Center,
National Institute of Advanced Industrial Science and Technology,
AIST Waterfront ANNEX, 2-4-7 Aomi, Koto-ku, Tokyo 135-0064, Japan
{xin.liu,ks.kim}@aist.go.jp

Abstract. In the era of big data, the study of networks has received an enormous amount of attention. Of recent interest is network embedding—learning representations of the nodes of a network in a low dimensional vector space, so that the network structural information and properties are maximally preserved. In this paper, we present a review of the latest developments on this topic. We compare modern methods based on matrix factorization, including GraRep [5], HOPE [22], DeepWalk [23], and node2vec [12], in a collection of 12 real-world networks. We find that the performance of methods depends on the applications and the specific characteristics of the networks. There is no clear winner for all of the applications and in all of the networks. In particular, node2vec exhibits relatively reliable performance in the multi-label classification application, while HOPE demonstrates success in the link prediction application. Moreover, we provide suggestions on how to choose a method for practical purposes in terms of accuracy, speed, stability, and prior knowledge requirement.

1 Introduction

We live in a complex world of interconnected entities [24,27]. In all fields of human endeavor, from biology to medicine, economics, and climate science, we are flooded with large-scale datasets. These datasets describe intricate real-world systems, with entities being modeled as nodes and their connections as edges, comprising various networks [24]. Indeed, we are surrounded by networks, including the internet, neural networks, cellular networks, food webs, electrical grids, communication networks, transportation networks, trade networks, and social networks [2,21]. Effective analysis of these networks can provide a solid understanding of the structure and dynamics of the real-world systems and can improve many useful applications [10].

Recently, a growing number of researchers have shown interests in learning representations of nodes of a network in a low dimensional vector space, or *network embedding* [4,11,13]. One important reason is that we can use the learned embeddings as feature inputs for downstream machine learning algorithms, and this technology is beneficial for many network analysis tasks, such as community detection [20], node classification [3], link prediction [17], recommendation [16], and visualization [28].

Many attempts to tackle the network embedding problem have been proposed [6,9,26,33,35]. The proposed methods are extremely varied, and are based on a range of different ideas. According to [4], the methods can be roughly classified into five categories: matrix factorization based methods, deep learning based methods, edge reconstruction based methods, graph kernel based methods, and generative model based methods. Among them, the matrix factorization based methods such as HOPE [22] have the advantage of high accuracy while preserving efficiency, and thus are widely used. Additionally, both the deep learning based methods, such as DeepWalk [23] and node2vec [12], and the edge reconstruction based methods, such as LINE [29] and PTE [28], are closely related to matrix factorization [15,25,34]. Therefore, in this paper we give a comparative study of these methods that are closely related to matrix factorization. Specifically, we make a comprehensive comparison in terms of accuracy, speed, stability, and prior knowledge requirement in two embedding enabled applications in a collection of 12 real-world networks. Then, we provide suggestions on how to choose a method for practical purposes.

To this end, this paper is organized as follows. Section 2 reviews the technique of matrix factorization and the methods to be compared. Section 3 specifies the experiment settings and datasets. Section 4 shows the comparison results. Based on these results, Sect. 5 gives our suggestion on how to choose a network embedding method for practical purposes. Finally, Sect. 6 presents our conclusion.

2 The Matrix Factorization Based Methods

We begin with the symbols and definitions that will be used. For simplicity and clarity, we limit our vision to an unweighted and undirected network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i \mid i = 1, \cdots, n\}$ is the node set, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set. The aim of network embedding is to learn a mapping ϕ: $v_i \mapsto \boldsymbol{e_i} \in \mathbb{R}^d$ for $\forall i = 1, \ldots, n$. $d \ll n$ is the embedding dimension. The sense of a good mapping is that the embeddings preserve the proximity structure between nodes. In other words, the (dis)similarity of embeddings in the $\mathbb{R}^d$ space should, to some extent, reflect the (dis)similarity of nodes in the original network.

Let $\mathbf{E} = (\boldsymbol{e_1}, \boldsymbol{e_2}, \cdots, \boldsymbol{e_n})^\top$ denote the embedding matrix, where the i-th row represents the embedding of v_i. The matrix factorization based methods have the unified form:

$$\mathbf{E} = \arg\min_{\mathbf{E}_l} \|\mathbf{S} - \mathbf{E}_l \mathbf{E}_r^\top\|_F^2, \tag{1}$$

where $\|\cdot\|_F$ denotes the matrix Frobenius norm, $\mathbf{S} \in \mathbb{R}^{n \times n}$ is some matrix defined on the network topology (different methods have different definitions), and $\mathbf{E}_l, \mathbf{E}_r \in \mathbb{R}^{n \times d}$ are matrices that factorize $\mathbf{S}$.

For a given $\mathbf{S}$, Eq. (1) is often solved approximately by Singular Value Decomposition (SVD) on $\mathbf{S}$ [5,15,22,25]. Suppose $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ denote the singular values of $\mathbf{S}$, and $\boldsymbol{u_i}, \boldsymbol{v_i}$ denote the corresponding left and right singular vectors of σ_i. Let $\mathbf{\Sigma_d} = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_d)$ be the diagonal matrix formed from the top d singular values, and let $\mathbf{U_d} = (\boldsymbol{u_1}, \boldsymbol{u_2}, \cdots, \boldsymbol{u_d})$ and $\mathbf{V_d} = (\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_d})$ be the matrices produced by selecting the corresponding left and right singular vectors. The truncated SVD of $\mathbf{S}$ can be expressed as

$$\mathbf{S_d} = \mathbf{U_d}\mathbf{\Sigma_d}(\mathbf{V_d})^\top \approx \mathbf{S}, \tag{2}$$

where $\mathbf{S_d}$ is the best rank-d matrix that approximates $\mathbf{S}$. Then, the embedding solution is:

$$\mathbf{E} = \mathbf{U_d}(\mathbf{\Sigma_d})^{\frac{1}{2}}. \tag{3}$$

Note that the matrix $\mathbf{S}$ to be factorized is based on user's definition. Previous research have shown the success of $\mathbf{S}$ based on different definitions, such as the adjacency matrix [1], modularity matrix [30], graph Laplacians [25,31], and Katz similarity matrix [22]. In addition, there are random walk and Skip-Gram model based methods such as DeepWalk [23] and node2vec [12] that factorize some matrices $\mathbf{S}$ implicitly.

In this paper, we limit our comparison to four methods that cover the state-of-the-art techniques, *i.e.* GraRep [5], HOPE [22], DeepWalk [23], and Node2vec [12], all of which are related to matrix factorization of the above form. We exclude other matrix factorization approaches such as LINE [29] and SocDim [30,31], because they have already been shown to be inferior to the methods considered here [12,23].

A brief introduction of the four methods follows.

- GraRep defines a loss function by integrating the transition probabilities. Minimizing this loss function has proven to be equivalent to factorizing a matrix that is related to the k-step transition probability matrix. For each k the factorization produces a sub-embedding. Then GraRep concatenates sub-embeddings on different k as the final embedding solution.
- HOPE learns embeddings by factorizing a similarity matrix defined on the Katz Index. The authors also developed a generalized SVD algorithm that can efficiently factorize a matrix in the form of $\mathbf{S} = \mathbf{M}_g^{-1}\mathbf{M}_l$, where $\mathbf{M}_g^{-1}$ and $\mathbf{M}_l$ are sparse matrices.
- DeepWalk first transforms a network into a collection of linear sequences of nodes using multiple random walks. It then learns embeddings by applying the Skip-Gram model [18,19], originating from natural language processing, to the sequences of nodes. DeepWalk implicitly factorizes a matrix, which is a low-rank transformation of the networks normalized Laplacian matrix [25].
- Node2vec is a variant of DeepWalk. Similar to DeepWalk, node2vec samples sequences of nodes and feed them to the Skip-Gram model. Instead of copying DeepWalk's random search sampling strategy, node2vec introduces two hyper-parameters to use 2nd-order random walks in order to bias the walks

towards a particular search strategy. Node2vec factorizes a matrix related to the stationary distribution and transition probability tensor of the 2nd-order random walk [25].

3 Experiment Settings

We evaluate these methods based on two embedding enabled applications: multi-label classification [23] and link prediction [12]. In the multi-label classification settings, every node is associated with one or more labels from a finite set $\mathcal{L}$. The task is executed according to the following procedure. First, we randomly sample a portion of the labeled nodes for training, with the rest for testing. Then, we use the learned embeddings (normalized by L2-norm) and the corresponding labels of the training nodes to train a one-vs-all logistic regression (LR) classifier (with L2 regularization). Finally, feeding the embeddings of the testing nodes to the classifier we predict their labels, which will be compared to the true labels for evaluation. We repeat this procedure 10 times and evaluate the performance in terms of the average *F1-Macro* and *F1-Micro* scores.

It is worth noting that previous approaches for training the LR classifier often ignores tuning the regularization strength parameter and accepts the default parameter value for granted [23]. We found that this parameter sometimes have a significant influence on the result, especially when the classification problem is highly imbalanced. Therefore, we carefully tune this parameter based on 10-fold cross-validation of the training nodes. Also, we note that at the prediction stage previous approaches often employes information that is typically unknown. Precisely, they use the actual number of labels m each testing node has [23,25]. They consider a label as a positive if it is among the top m labels in terms of prediction probability by the LR classifier, regardless of its real probability value. However, in real-world situations it is fairly uncommon to have such prior knowledge of m. We eliminate the use of the prior knowledge in a similar way as proposed in [8]. Instead of ranking the prediction probabilities and taking the labels corresponding to the top m, we label a testing node based on the probability directly, *i.e.*, if the probability of a label l is greater than 0.5 we consider l as positive.

In the link prediction task, we are given a network $\mathcal{G}'$ with 50% of edges removed from the original network $\mathcal{G}$. We predict the missing edges (*i.e.* the 50% removed edges) according to the procedures in [12]. First, based on the node embeddings learned from $\mathcal{G}'$, we generate edge embeddings for pairs of nodes using the element-wise operators listed below. We label an edge embedding as positive if the corresponding edge exists in $\mathcal{G}'$ and negative otherwise. Then, we train a binary LR classifier (with L1 regularization) using all of the edge embeddings that have positive labels and the same amount of randomly sampled edge embeddings that have negative labels. After that, feeding an edge embedding to the LR classifier we can calculate the existence probability of the corresponding edge. Finally, we evaluate the performance based on the probabilities of the missing edges and non-existent edges (*i.e.* the edges that do not exist in $\mathcal{G}$) in terms of the *Area Under the Curve (AUC)* score.

The element-wise operators for generating edge embeddings are:

- Average: $[\boldsymbol{e}_{ij}]_t = \big([\boldsymbol{e}_i]_t + [\boldsymbol{e}_j]_t\big)/2$,
- Hadamard: $[\boldsymbol{e}_{ij}]_t = [\boldsymbol{e}_i]_t \cdot [\boldsymbol{e}_j]_t$,
- Weighted L1: $[\boldsymbol{e}_{ij}]_t = |[\boldsymbol{e}_i]_t - [\boldsymbol{e}_j]_t|$,
- Weighted L2: $[\boldsymbol{e}_{ij}]_t = |[\boldsymbol{e}_i]_t - [\boldsymbol{e}_j]_t|^2$,

where $t \in 1, \cdots, d$ denotes the subscript of the t-th element of an embedding.

Table 1. Statistics of the datasets.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{L}\rvert$
Kaggle3059	157	2,474	15
Kaggle4406	399	3,412	28
BrazilAir	131	1,003	4
EuropeAir	399	5,993	4
USAir	1,190	13,599	4
Cora	2,708	5,278	7
Citeseer	3,264	4,551	6
DBLP	13,184	47,937	5
WikiPage	2,363	11,596	17
WikiWord	4,777	92,295	40
PPI	3,860	37,845	50
BlogCatalog	10,312	333,983	39

We use a variety of real-world network datasets from various domains. A brief description of them follows.

- Kaggle3059, Kaggle4406 [7][1]: The friendship networks of Facebook users. The labels represent the social circles of the users.
- BrazilAir [26][2], EuropeAir [26][3], USAir [26][4]: The air-traffic networks of Brazil, Europe, and the USA, respectively. The nodes indicate airports and the edges denote the existence of commercial flights. The labels represent the capacity levels of the airports.
- Cora [35][5], Citeseer [14](See footnote 5), DBLP [28][6]: Paper citation networks. The labels represent the topics of the papers.

[1] https://www.kaggle.com/c/learning-social-circles/data.
[2] http://www.anac.gov.br/.
[3] http://ec.europa.eu/.
[4] https://transtats.bts.gov/.
[5] https://linqs.soe.ucsc.edu/data/.
[6] https://aminer.org/billboard/citation/.

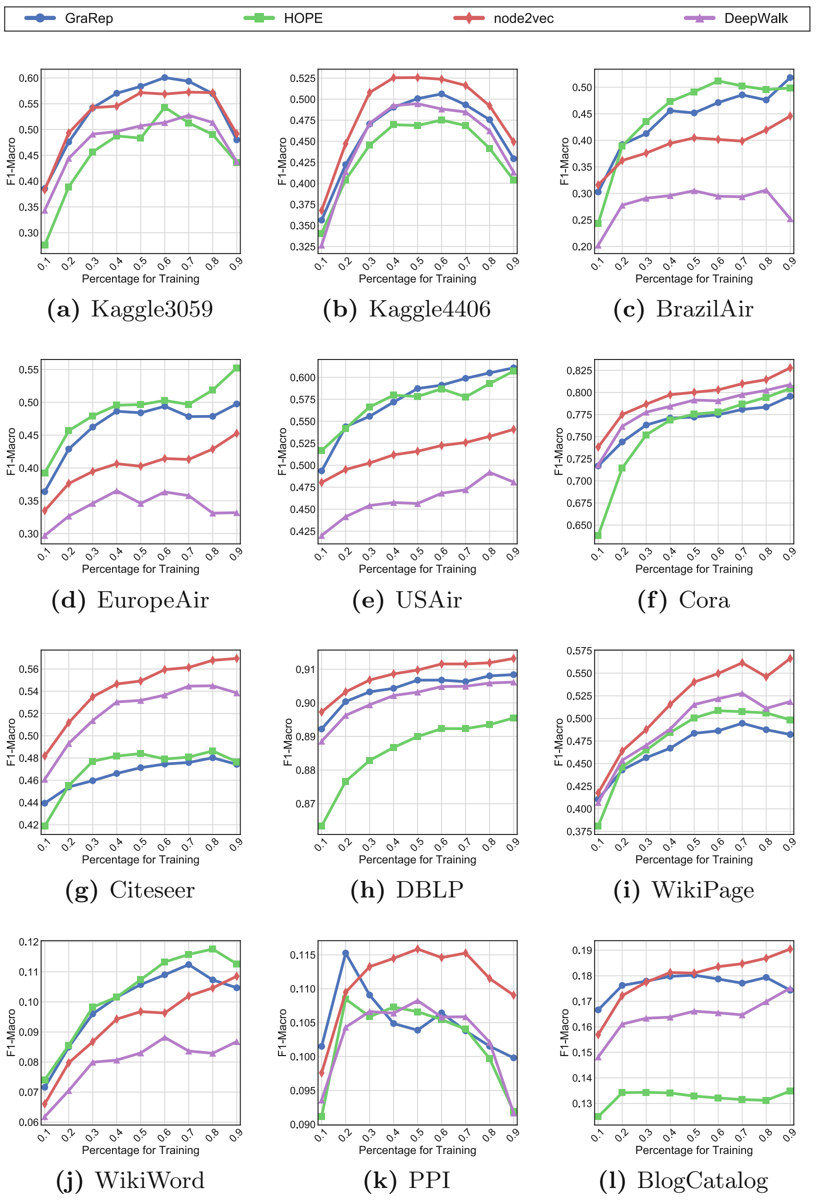

Fig. 1. F1-Macro of multi-label classification on the 12 networks.

– WikiPage [32][7]: A network of webpages in Wikipedia, with edges indicating hyperlinks. The labels represent the topic categories of the webpages.

[7] https://github.com/thunlp/MMDW/tree/master/data/.

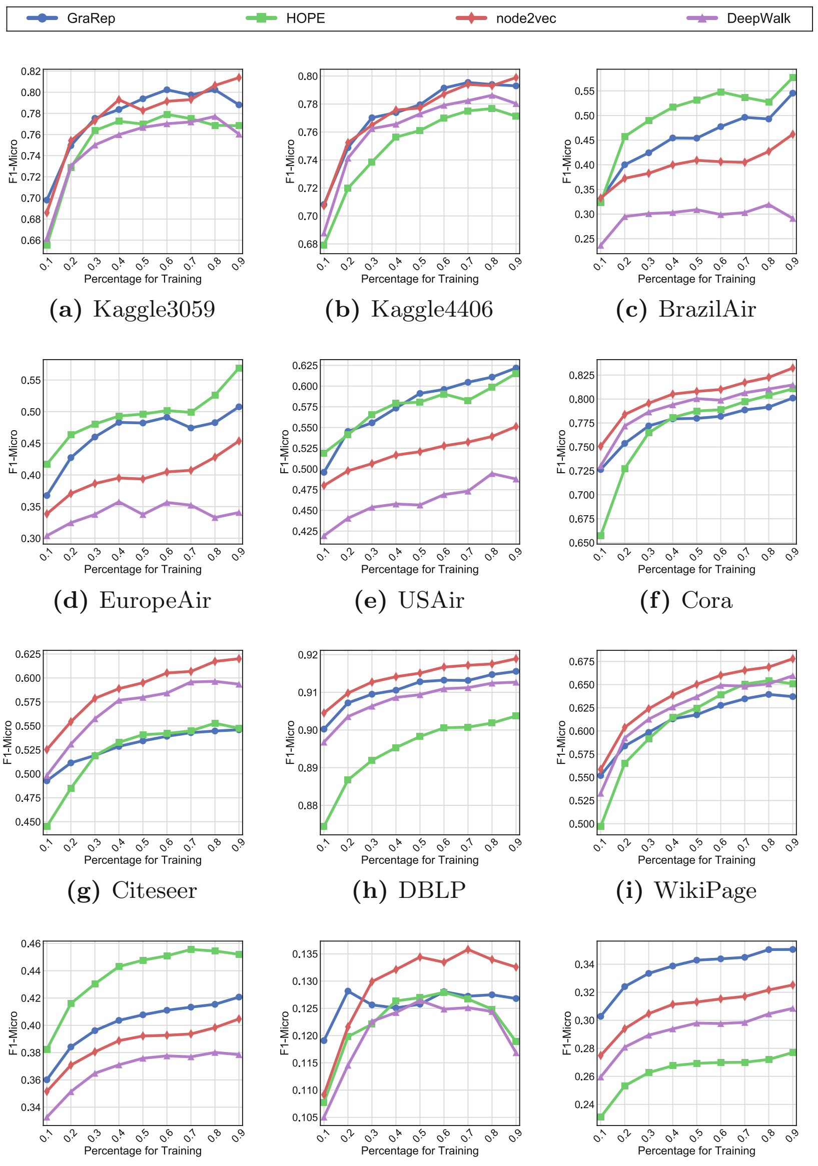

Fig. 2. F1-Micro of multi-label classification on the 12 networks.

– WikiWord [12][8]: A co-occurrence network of the words appearing in Wikipedia. The labels represent the part-of-speech tags inferred using the Stanford POS-Tagger.

[8] http://snap.stanford.edu/node2vec/#datasets/.

- PPI [12](See footnote 8): A subgraph of the protein-protein interactions network for Homo Sapiens. The labels represent the biological states.
- BlogCatalog [30][9]: A network of social relationships of the bloggers listed on the BlogCatalog website. The labels represent topic categories provided by the bloggers.

We remove self-loop edges and transform bi-directional edges to undirected edges. The datasets after pre-processing are summarized in Table 1.

We uniformly set the embedding dimension as 120 for all methods. The parameter settings for each method are in line with the typical ways. That is, for GraRep, we set the maximum matrix transition step as 4. For HOPE, we set the decay rate as 0.95 divided by the spectral radius of the adjacency matrix. For DeepWalk and node2vec, we set the window size as 10, the walk length as 80, the number of walks per node as 10. Lastly, for node2vec, we learn the best in-out and return hyperparameters using a grid search over {0.25, 0.50, 1, 2, 4}.

4 Results

Figures 1 and 2 depict the F1-Macro and F1-Micro scores of multi-label classification on different networks. Overall, F1-Macro and F1-Micro scores show the similar trend, although there are many differences in the details. We find that the performance is network dependent. Node2vec achieves remarkable performance in five out of the 12 networks (Cora, Citeseer, DBLP, WikiPage, PPI). However, both node2vec and DeepWalk, which employ random walk with Skip-Gram model strategy, show less success in the air-traffic networks (BrazilAir, EuropeAir, and USAir). One reason is that the labels in these networks are, to some extent, an indication of the structural identity. However, random walk is not able to find such identify that implies the symmetry structure. Node2vec is always better than DeepWalk, because the former uses additional ground-truth labels for learning the in-out and return hyperparameters and guide the 2nd-order random walkers towards a particular search strategy. Moreover, HOPE outperforms the other methods in three networks (BrazilAir, EuropeAir, WikiWord) but loses to the competition in another three networks (Kaggle4406, DBLP, and BlogCatalog). GraRep obtains good result in three networks (Kaggle3059, Kaggle4406, BlogCatalog), where there is no clear winner. Also, GraRep demonstrates acceptable performance in other networks, except in Cora, Citeseer and WikiPage. Comparatively speaking, node2vec, as an improved version of DeepWalk, exhibits relatively reliable performance in the multi-label classification task.

Table 2 shows the link prediction results. Again, the performance of different methods is network dependent. HOPE performs the best in seven out of the 12 networks. Impressively, in BrazilAir network the performance gain over DeepWalk is as high as 24.27%. Note that the similarity matrix factorized by HOPE is defined on the Katz index and preserves higher order proximity between nodes.

[9] http://socialcomputing.asu.edu/datasets/BlogCatalog3/.

This implies that preserving higher order proximity is conducive to predicting unobserved links. However, in the DBLP network HOPE obtains the lowest score, which is significantly lower than the others. A reason is that DBLP network is very sparse.[10] So, the network $\mathcal{G}'$ that are obtained by removing 50% of the edges of the original network $\mathcal{G}$ contains many disconnected components. However, Katz index is insufficient to measure the proximity for pairs of nodes that come from disconnected components, consequently resulting in the less attractive performance. One the other hand, GraRep and node2vec only outperform the others in three and two networks, respectively. Like the situation for muti-label calssification, DeepWalk is always inferior to node2vec. Comparatively speaking, the performance of HOPE is more consistent, as it obtains high scores in almost all of the networks.

Table 2. AUC scores of link prediction on the 12 datasets. The scores are based on the best results of choosing different operators for edge embedding.

Dataset	Method			
	GraRep	HOPE	DeepWalk	node2vec
Kaggle3059	0.9386	**0.9389**	0.8888	0.9144
Kaggle4406	0.9619	**0.9633**	0.9588	0.9613
BrazilAir	0.8903	**0.8951**	0.7203	0.7582
EuropeAir	**0.9083**	0.8992	0.8226	0.8383
USAir	0.9434	**0.9501**	0.8650	0.8934
Cora	0.6947	0.7018	0.7324	**0.7381**
Citeseer	**0.6939**	0.6673	0.6315	0.6535
WikiPage	**0.8863**	0.8839	0.8763	0.8827
WikiWord	0.9015	**0.9097**	0.8551	0.8656
PPI	0.8630	**0.8700**	0.8033	0.8097
DBLP	0.9216	0.9078	0.9224	**0.9231**
BlogCatalog	0.9307	**0.9387**	0.8774	0.8878

5 Choosing a Method

The results in the previous section indicate the accuracy of the methods in multi-label classification and link prediction tasks. However, we have to take other factors into account when choosing a method for practical purposes. In some cases, a compromise must be reached between accuracy and running time, especially for large networks. In other cases, we may consider whether a method is stable, or whether we have some prior knowledge about the network. Table 3 summarizes the characteristics of different methods from five aspects: accuracy

[10] The similar reason also applies to the Cora and Citeseer networks.

for multi-label classification, accuracy for link prediction, speed, stability, and prior knowledge requirement. To clarify this further, we give the following suggestions for choosing a suitable method.

In one example, we want to embed a relatively small network that contains several hundred nodes. Since the network is small, the speed of a method should pose no restriction, and we are free to choose the most accurate method. In this case, GraRep and node2vec would be an appropriate choice. When we deal with large networks, it becomes intractable with methods such as GraRep. For example, it may take more than ten days to embed a network with the number of nodes in the order of 10^5 on a current desktop PC. In this case, HOPE and node2vec would be better candidates.

Table 3. Characteristics of different methods.

Characteristic	Method			
	GraRep	HOPE	DeepWalk	node2vec
Accuracy (Clas.)	High	Medium	Medium	X-High
Accuracy (Pred.)	High	X-High	Medium	High
Speed	Medium	X-Fast	Fast	Fast
Stability	Stable	Stable	Unstable	Unstable
Prior Know. Reqt.	No	No	No	Yes

Let us consider a network that evolves with time. We want to embed the network at multiple time slices and find out how the embeddings change with time. In this case, we would not choose unstable methods such as DeepWalk or node2vec. These two methods introduce some random factors such as random walks, random initialization of the neural network weights, and random selection for the mini-batch training. Thus, the embedding results can be different in different runs.

Node2vec, as an improved version of DeepWalk, normally achieves better performances. However, we need some prior knowledge to guide the 2nd-order random walkers towards a particular search strategy or some information about the downstream machine learning tasks. For example, some labels are required to tune the hyper-parameters in the multi-label classification task. If such prior knowledge is not available, DeepWalk is also a good alternative option.

6 Conclusion

In this paper, we have given a overview and a comprehensive comparison of matrix factorization based methods for network embedding. We found that the performance is network and application dependent; thus, there is no clear winner for all of the applications and in all of the networks. We analyzed the characteristics of each method and gave suggestions on how to choose a method for practical purposes.

Despite recent efforts in network embedding, some questions remain unanswered. The search for a faster and more accurate method is a never-ending pursuit. Additionally, we lack embedding methods that can cope with temporal networks. These will be left for our future work.

Acknowledgment. This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

1. Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., Smola, A.J.: Distributed large-scale natural graph factorization. In: Proceedings of the 22nd International World Wide Web Conference, WWW 2013, Rio de Janeiro, Brazil, pp. 37–48 (2013)
2. Barabási, A.L.: Linked: The New Science of Networks. Perseus Publishing, Cambridge (2002)
3. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. In: Social Network Data Analytics, pp. 115–148 (2011)
4. Cai, H., Zheng, V.W., Chang, K.C.C.: A comprehensive survey of graph embedding: problems, techniques and applications. arXiv preprint arXiv:1709.07604 (2017)
5. Cao, S., Lu, W., Xu, Q.: GraRep: learning graph representations with global structural information. In: Proceedings of the 24th ACM Conference on Information and Knowledge Management, CIKM 2015, Melbourne, Australia, pp. 891–900 (2015)
6. Cao, S., Lu, W., Xu, Q.: Deep neural networks for learning graph representations. In: Proceedings of 20th AAAI Conference on Artificial Intelligence, AAAI 2016, Phoenix, AZ, USA, pp. 1145–1152 (2016)
7. Chen, S., Niu, S., Akoglu, L., Kovačević, J., Faloutsos, C.: Fast, warped graph embedding: unifying framework and one-click algorithm. arXiv preprint arXiv:1702.05764 (2017)
8. Faerman, E., Borutta, F., Fountoulakis, K., Mahoney, M.W.: LASAGNE: locality and structure aware graph node embedding. arXiv preprint arXiv:1710.06520 (2017)
9. García-Durán, A., Niepert, M.: Learning graph embeddings with embedding propagation. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS 2017, Long Beach, CA, USA (2017)
10. Getoor, L., Diehl, C.P.: Link mining: a survey. ACM SIGKDD Explor. Newsl. **7**(2), 3–12 (2005)
11. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: a survey. arXiv preprint arXiv:1705.02801 (2017)
12. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2016, San Francisco, CA, USA, pp. 855–864 (2016)
13. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: methods and applications. arXiv preprint arXiv:1709.05584 (2017)
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France (2017)

15. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS 2014, pp. 2177–2185 (2014)
16. Lü, L., Medo, M., Yeung, C.H., Zhang, Y., Zhang, Z., Zhou, T.: Recommender systems. Phys. Rep. **519**, 1–49 (2012)
17. Lü, L., Zhou, T.: Link prediction in complex networks: a survey. Physica A **390**, 1150–1170 (2011)
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS 2013, pp. 3111–3119 (2013)
20. Newman, M.E.J.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. USA **103**(23), 8577–8582 (2006)
21. Newman, M.E.J.: Networks: An Introduction. Oxford University Press, New York (2010)
22. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2016, San Francisco, CA, USA, pp. 1105–1114 (2016)
23. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, New York, NY, USA, pp. 701–710 (2014)
24. Pržulj, N., Malod-Dognin, N.: Network analytics in the age of big data. Science **353**(6295), 123–124 (2016)
25. Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., Tang, J.: Network embedding as matrix factorization: unifying DeepWalk, LINE, PTE, and node2vec. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining, WSDM 2018 (2018)
26. Ribeiro, L.F.R., Saverese, P.H.P., Figueiredo, D.R.: struc2vec: learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2017, Halifax, Nova Scotia, Canada, pp. 385–394 (2017)
27. Scholtes, I.: Understanding complex systems: when big data meets network science. IT Inf. Technol. **57**(4), 252–256 (2015)
28. Tang, J., Qu, M., Mei, Q.: PTE: predictive text embedding through large-scale heterogeneous text networks. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2015, Sydney, Australia, pp. 1165–1174 (2015)
29. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: Proceedings of the 24th International World Wide Web Conference, WWW 2015, Florence, Italy, pp. 1067–1077 (2015)
30. Tang, L., Liu, H.: Relational learning via latent social dimensions. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 817–826 (2009)
31. Tang, L., Liu, H.: Leveraging social media networks for classification. Data Min. Knowl. Discov. **23**(3), 447–478 (2011)

32. Tu, C., Zhang, W., Liu, Z., Sun, M.: Max-Margin DeepWalk: discriminative learning of network representation. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, pp. 3889–3895 (2016)
33. Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., Yang, S.: Community preserving network embedding. In: Proceedings of the 21st AAAI Conference on Artificial Intelligence, AAAI 2017, San Francisco, CA, USA, pp. 203–209 (2017)
34. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.: Network representation learning with rich text information. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI 2015, Austin, TX, USA, pp. 2111–2117 (2015)
35. Yang, C., Sun, M., Liu, Z., Tu, C.: Fast network embedding enhancement via high order proximity approximation. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, pp. 3894–3900 (2017)