Comparing Graph Embedding Algorithms on Various Dataset

Ryosuke Kato
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093
rkato@ucsd.edu

Abstract

Three graph embedding algorithms, DeepWalk, Node2Vec, and Poincaré embeddings, were compared in two tasks: distortion and label prediction of graph datasets with different properties. Among the three, DeepWalk showed better performance on small but real datasets, while Poincaré embeddings outperformed others on random tree datasets in the prediction task. There is a tradeoff between the accuracy of prediction and computing time when adding the minimum spanning tree (MST) preprocess to Poincaré embeddings.

1. Introduction and Motivation

The graph is a generic tool to represent an entity where elements have relations or interact with one another. For instance, it can be used not only for originally graphical data such as social networks or wireless networks, but also for dynamic models. In a dynamic model, interaction occurs between each pair of agents comprising the system. Moreover, it is frequently used in preprocess of the high-dimensional vectorize data such as Isomap, a manifold learning algorithm. A graph has an inherently complex structure, and how to represent such a structure when we analyze it or use it as input data for machine learning pipelines is an important problem. Several embedding techniques, such as DeepWalk [1] or Node2Vec [2], have been proposed and are widely used. In addition, hyperbolic embedding algorithms have recently been proposed [3], and it turns out that they are useful for representing exponentially spread data such as trees.

The types of graphs that exist are much more varied than when we consider only the usual Euclidean metric space. If we think about mathematical and canonical graphs, there are a lot of structures. Moreover, there are many classes of random graphs or complex networks, and we see diverse graphical data in the real world; some are similar to canonical graphs such as line, star, or complete graphs, while others are far from canonical models. Therefore, it can be useful to apply these embedding algorithms to a broad spectrum of graphical datasets and measure their performance in various settings.

2. Related Work

2.1. Node Embedding

2.1.1 DeepWalk

DeepWalk is a simple and intuitive algorithm that transforms each node into a vector representation by using a sequence of steps of random walks on the given graph. The DeepWalk algorithm[1] consists of two main components: (i) a random walk generator and (ii) an update procedure. The random walk generator takes a graph $\mathcal{G}$ and uniformly samples a random node v_i as the root of the random walk $\mathcal{W}_{v_i}$. A walk samples uniformly from the neighbors of the last node visited until the maximum length set by a user is reached. This step is iterated for each node v_i to obtain $\mathcal{W}_{v_i}$, and then it is used to update the representation of each node v_i with the SkipGram algorithm [12]. SkipGram is widely used in natural language processing to minimize the following objective function:

$$minimize_{\Phi} - logPr(\{v_{i-w_i}, ..., v_{i-1}, v_{i+1}, ..., v_{i+w}\}|\Phi v_i), \quad (1)$$

where $\Phi : v \in V \rightarrow \mathbb{R}^{|V| \times d}$ maps the latent representation associated with each node v in the graph.

2.1.2 Node2Vec

The core of the Node2Vec algorithm [2] is the same as DeepWalk. That is, it uses a random walk and SkipGram to obtain the representation of a given node. Node2Vec added flexibility to the random walk procedure and produced a rich representation of the data. It adds a bias parameter, α, to the random walk whereas DeepWalk relies on unbiased sampling from the neighbors of a given node. Suppose

a walk is now on an edge (t, v) and we have just arrived at v from t. Then, when we consider which node will be next, we compute the bias parameter α for each neighboring node to v_i as follows:

$$\alpha_{p,q}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{t,x} = 0 \\ 1 & \text{if } d_{t,x} = 1 \\ \frac{1}{q} & \text{if } d_{t,x} = 2 \end{cases}, \quad (2)$$

using parameters p and q; $d_{t,x} \in \{0, 1, 2\}$ denotes the shortest path length from t to x.

2.2. Hyperbolic Embedding

2.2.1 Poincaré Embeddings

Poincaré embeddings[3] succeeded in reducing dimensions of representation of the data significantly by mapping each data point into hyperbolic space. It has been argued that we can efficiently embed exponential data such as a tree in the Poincaré ball model of hyperbolic space. In addition, we can compute distance on the model analytically. For example, the distance on the Poincaré ball is computed explicitly and simply as follows:

$$d(x,y) = arccosh\Big(1 + 2\frac{\|x-y\|^2}{(1-\|x\|^2)(1-\|y\|^2)}\Big), \quad (3)$$

where x and y are coordinates of two points on the disk model (i.e., $x \in \{(x_1, x_2) \mid x_1^2 + x_2^2 \leq 1\}$) and $\|\cdot\|$ is the $L-2$ norm. In Poincaré embeddings, the objective function below is optimized in a similar way to SkipGram [12] (i.e., we optimize it so that distances of positive samples become small and negative samples become large):

$$\mathcal{L}(\Theta) = \sum_{(i,j)\in\mathcal{D}} \log \frac{\exp(-d(v_i, v_j))}{\sum_{j'\in\mathcal{N}(i)} \exp(-d(v_i, v_{j'}))}, \quad (4)$$

where $\mathcal{D}$ is the given dataset and $\mathcal{N}(i)$ is the neighbor nodes of node i.

As well as the Poincaré embeddings model alone, Poincaré embeddings combined with a minimum spanning tree (MST) algorithm were implemented. Distortions and prediction errors of Poincaré embeddings tend to be large, and computing time is also long if an input graph data has many edges. This can be mitigated by deleting redundant edges with Kruskal's algorithm[13] and then inputting extracted minimum spanning tree into Poincaré embeddings algorithm.

3. Method

3.1. Task

3.1.1 Average Distortion

First, the average distortion for each model and each dataset is computed. Given two metrics d_1, d_2 on a finite set $X = x_1, ..., x_n$, the average distortion is:

$$\frac{1}{\binom{n}{2}} \sum_{i=1}^{n} \sum_{j<i} \frac{|d_1(x_i, x_j) - d_2(x_i, x_j)|}{d_2(x_i, x_j)}. \quad (5)$$

Smaller average distortion implies greater similarity between d_1 and d_2.

3.1.2 Label Prediction

Each node of the various datasets is labeled. The karate club dataset is binary labeled, while all the others are multi-labeled. For each dataset, embeddings of 50% of nodes were used as training data and performances were evaluated based on the macro F-1 score of the labels they predicted for the remaining nodes.

3.2. Dataset

Details of each dataset are shown in Table 1. There are three categories: (i) canonical graphs, (ii) grid and random graphs, and (iii) real datasets.

3.2.1 Synthesized: Canonical Graph

Six types of canonical graphs (line graph, star graph, complete graph, chordal graph, tree, and binary tree) are used. We also change the size of the graph ($|V| = \{100, 300, 1000\}$, except for the complete graph where $|V| = \{10, 30, 100\}$), to avoid expensive computation. Each graph is computationally generated. Starting from a root node, a random tree is generated by adding new nodes one-by-one and creating an edge between the new node and one of the existing nodes. A partner of the newly added node is selected from the existing nodes with equal probability. A chordal graph is generated by constructing a tree and adding edges between a randomly chosen pair with a distance of 2 when we assume all edges has equal weight (distance) 1.

For task 2, we use (i) a line graph, (ii) tree, and (iii) binary tree. Each node is labeled based on (i) distance from either end of a line (i.e., from a node whose degree is 1 for the line graph), (ii) distance from the root (random tree), or (iii) depth from the root (binary tree). The performances

Table 1. Property of Each Graph

Name	$\|V\|$	$\|E\|$	Type	Tree	Random	Edge
Line Graph	{100,300,1000}	{99,299,999}	Synthesized	Yes	No	Unweighted
Star Graph	{100,300,1000}	{99,299,999}	Synthesized	Yes	No	Unweighted
Complete Graph	{10,30,100}	$\{\binom{10}{2},\binom{30}{2},\binom{100}{2}\}$	Synthesized	No	No	Unweighted
Chordal Graph	{100,300,1000}	{599, 1799, 5999}	Synthesized	No	No	Unweighted
Random Tree	{100,300,1000}	{99,299,999}	Synthesized	Yes	Yes	Unweighted
Binary Tree	{127,255,1023}	{126,254,1022}	Synthesized	Yes	Yes	Unweighted
Grid Graph	400	1482	Synthesized	No	No	Unweighted
Random Graph	400	7883	Synthesized	No	Yes	Unweighted
Power Law Network	400	1975	Synthesized	No	Yes	Unweighted
Small World Network	400	1193	Synthesized	No	Yes	Unweighted
Community Network	400	6315	Synthesized	No	Yes	Unweighted
Karate Club	34	78	Real	No	No	Unweighted
Football	115	613	Real	No	No	Unweighted
Les Miserables	77	254	Real	No	No	Weighted

are evaluated in different sizes ($V = 100, 300, 1000$) for each graph.

3.2.2 Synthesized: Grid and Random Network

Five types of networks are used: grid, simple random network, power-law network, small-world network, and community network. The size of each network is set to $|V| = 400$. The simple random graph is called the Erdős-Rényi model [7] and is created with Gilbert's algorithms [8]. The power-law network and small-world network are created with the Barabási-Albert model [9] and Newman-Watts-Strogatz model [10], respectively. The community graph is generated by a generalization of the planted-l-partition described in [6].

For task 2, only a community network is used. Each node is labeled based on the community (community graph).

3.2.3 Real Data

Three real datasets were used: (i) Zachary's Karate Club graph [4], (ii) football teams [5], and (iii) Les Miserables characters [11]. Sets (i) and (ii) were used for the label prediction task. In (i), each node represents a person in the club, and each person belongs to either of two groups. Therefore, it is a binary classification problem. Set (ii) represents football games between colleges. Each node represents a college, which belongs to one of 11 conferences; we predict which conference it belongs to. Set (iii) represents the co-appearance network of characters in the novel Les Miserables.

3.3. Embedding Algorithms

Embedded vectors transformed by each embedding algorithm were input into logistic regression with an L_2 penalty. For a multi-labeled dataset, the macro F-1 score was computed by the one-versus-rest method.

4. Result and Discussion

4.1. Average Distortion

The results are shown in Table 2. DeepWalk and Node2Vec show a similar level of distortion, while that of Poincaré embeddings is large. When we use MST to preprocess Poincaré embeddings, the distortion is mitigated to some extent.

4.2. Label Prediction

The results are shown in Table 3. Although all three algorithms showed their strengths and weaknesses for each dataset, DeepWalk showed better results with small and real datasets, while for large datasets (i.e., if $|V|$ is large) computing time becomes non-negligible. In this case, Node2Vec is a good candidate to represent data with preserving its structure relatively precisely as well as computing in reasonable time. Poincaré embeddings achieved the highest scores for all sizes of random trees, more likely structure in real settings than a line or binary trees, and this result implies Poincaré embeddings may be useful for real tree-structured or tree-like (few redundant edges) datasets despite their poor performance in non-tree datasets (Karate, football, and community). By adding MST to Poincaré embeddings, macro F-1 scores worsen while computation time is improved. Therefore, we can decide whether to use MST on a case-by-case basis, considering the tradeoff between accuracy and time.

Table 2. Average Distortion for Each Embeddings of Each Dataset

	Line Graph			Star Graph			Complete Graph			Chordal Graph			Random Tree			Binary Tree		
$\|V\|$	100	300	1000	100	300	1000	10	30	100	100	300	1000	100	300	1000	127	255	1023
DeepWalk	0.51	0.57	0.58	0.14	0.08	0.07	0.10	0.14	0.20	0.13	0.20	0.22	0.18	0.19	0.20	0.15	0.17	0.16
Node2Vec	0.49	0.56	0.58	0.26	0.33	0.33	0.32	0.17	0.20	0.16	0.13	0.17	0.16	0.19	0.21	0.13	0.16	0.15
PE	0.50	0.55	0.58	0.39	0.32	0.26	0.50	0.88	0.88	0.42	0.41	0.38	0.33	0.31	0.33	0.32	0.28	0.25
MST+PE	—	—	—	—	—	—	0.27	0.25	0.19	0.29	0.22	0.27	—	—	—	—	—	—

	Grid	Random Graph	Power Law	Small World	Community	Karate	Football	Les Miserables
DeepWalk	0.33	0.10	0.13	0.14	0.17	0.29	0.14	0.16
Node2Vec	0.29	0.14	0.13	0.12	0.22	0.42	0.16	0.33
PE	0.28	0.71	0.55	0.32	0.63	0.55	0.43	0.39
MST+PE	0.37	0.18	0.21	0.20	0.28	0.56	0.25	0.33

Table 3. Macro-F1 Scores of the Label Prediction Task

		Line Graph			Binary Tree			Random Tree			Community	Karate	Football
	Tree	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
Property	$\|V\|$	100	300	1000	100	300	1000	100	300	1000	400	34	115
of Graph	$\|E\|$	99	299	999	99	299	999	99	299	999	6315	78	613
	# labels	5	5	5	7	9	10	5	5	5	4	2	12
	Macro-F1	**0.923**	0.955	0.759	0.555	**0.720**	**0.684**	0.208	0.209	0.235	0.946	**0.944**	**0.864**
DeepWalk	Std	0.043	0.015	0.017	0.076	0.086	0.039	0.065	0.044	0.023	0.026	0.049	0.057
	Time	0.76	3.21	12.94	0.66	3.36	13.28	1.15	2.75	14.42	0.84	0.25	0.84
	Macro-F1	0.916	**0.972**	**0.983**	0.570	0.527	0.670	0.198	0.163	0.206	**1.0**	0.767	0.657
Node2Vec	Std	0.029	0.015	0.0096	0.067	0.057	0.036	0.061	0.026	0.027	0.0	0.15	0.079
	Time	0.70	2.19	7.05	0.73	1.87	6.81	1.25	5.13	8.89	4.54	0.30	1.05
	Macro-F1	0.811	0.913	0.899	**0.607**	0.646	0.596	**0.579**	**0.418**	**0.358**	0.539	0.635	0.598
PE	Std	0.085	0.035	0.019	0.083	0.070	0.050	0.12	0.13	0.046	0.056	0.097	0.15
	Time	1.39	1.14	3.59	1.36	2.60	3.64	1.37	1.24	3.49	5.12	0.92	6.76
	Macro-F1	—	—	—	—	—	—	—	—	—	0.47	0.63	0.45
MST+PE	Std	—	—	—	—	—	—	—	—	—	0.045	0.097	0.070
	Time	—	—	—	—	—	—	—	—	—	2.08	0.97	2.05

"Macro-F1" and "Std" are average and standard deviation of macro-F1 value of 20 trials.

5. Conclusion and Future Work

We see that the three embedding algorithms have different properties, and we can choose appropriate embedding algorithms depending on the structure of the graphical dataset. Also, the better performance of Poincaré embeddings with tree data implies that we need to preprocess graphical data if it has a lot of edges. In this experiment, we did not test the weighted graph (a graph that has parameters such as edges or large-scale datasets, or that performs other possible tasks such as link prediction). We might be able to improve the performance of embedding algorithms by tuning parameters or preprocessing, which are avenues for future investigation.

References

[1] Bryan Perozzi, Rami Al-Rfou, Steven Skiena. *DeepWalk: Online Learning of Social Representations*. https://arxiv.org/abs/1403.6652. 2014.

[2] Aditya Grover, Jure Leskovec. *node2vec: Scalable Feature Learning for Networks*. https://arxiv.org/abs/1607.00653. 2016.

[3] Maximilian Nickel and Douwe Kiela. *Poincaré Embeddings for Learning Hierarchical Representations*. In NIPS. 2017.

[4] Zachary W. *An information flow model for conflict and fission in small groups*. Journal of Anthropological Research, 33, 452-473. 1977.

[5] M. Girvan and M. E. J. Newman. *Community structure in social and biological networks*. Proc. Natl. Acad. Sci. USA 99, 7821-7826. 2002.

[6] Santo Fortunato. *Community Detection in Graphs*. Physical Reports Volume 486, Issue 3-5 p. 75-174. https://arxiv.org/abs/0906.0612.

[7] P. Erdős and A. Rényi. *On Random Graphs*. Publ. Math. 6, 290. 1959.

[8] E. N. Gilbert. *Random Graphs*. Ann. Math. Stat., 30, 1141. 1959.

[9] A. L. Barabási and R. Albert. *Emergence of scaling in random networks*. Science 286, pp 509-512, 1999.

[10] M. E. J. Newman and D. J. Watts. *Renormalization group analysis of the small-world network model*. Physics Letters A, 263, 341, 1999.

[11] D. E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, Reading, MA, 1993.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. https://arxiv.org/abs/1301.3781. 2013.

[13] Kruskal, J. B. *On the shortest spanning subtree of a graph and the traveling salesman problem*. Proceedings of the American Mathematical Society. 7 (1): 48–50. 1956.