Wire Before You Walk

Tesfa Asmara[1,2], Dhananjay Bhaskar[3], Ian Adelstein[4], Smita Krishnaswamy[3,5,6,7,8], Michael Perlmutter[9†]
[1]Department of Computer Science, Pomona College, Claremont, CA, USA
[2]Mathematics and Statistics, Pomona College, Claremont, CA, USA
[3]Department of Genetics, Yale School of Medicine, New Haven, CT, USA
[4]Department of Mathematics, Yale University, New Haven, CT, USA
[5]Department of Computer Science, Yale University, New Haven, CT, USA
[6]Program for Applied Mathematics, Yale University, New Haven, CT, USA
[7]Computational Biology and Bioinformatics Program, Yale University, New Haven, CT, USA
[8]Wu Tsai Institute, Yale University, New Haven, CT, USA
[9]Department of Mathematics, University of California, Los Angeles, CA, USA
[†]Corresponding Author: perlmutter@math.ucla.edu

***Abstract*—Node embeddings aim to associate a vector to every vertex of a graph which can then be used for downstream tasks such as clustering, classification, or link prediction. Many popular node embeddings such as node2vec and DeepWalk are based upon counting which nodes frequently co-occur in random walks of the graph. In this paper, we show that the performance of such algorithms can be improved by rewiring the edges of the graph through a variety of network indices before running DeepWalk. These rewirings effectively give the random walker an inductive bias and increase the accuracy of a logistic regression classifier applied to the node embedding on several benchmark data sets.**

***Index Terms*—learning on graphs, node embeddings, skip-gram methods**

I. Introduction

Node embeddings aim to represent each vertex of a graph by a vector in a relatively low dimensional space. Typically, these vector representations are obtained in an unsupervised manner and only rely on the network's geometry, rather than features or labels associated with each of the nodes.

Many popular algorithms for obtaining node embeddings rely on the skip-gram framework. These skip-gram based methods adapt algorithms designed for natural language processing, such as Word2vec [4] to the network setting. Whereas Word2vec aims to produce similar representations of words that frequently co-occur in real-world sentences, algorithms such as DeepWalk [5] and Node2vec [2] aim to produce similar representations of nodes which frequently co-occur in (possibly biased) random walks of the graph. A notable advantage of these skip-gram based algorithms is that they are able to learn representations of the vertices purely from the geometry of the graph. In particular, they do not require one to be given a matrix of informative node features.

In this paper we introduce WireWalk, a novel method for producing node embeddings. Our method is based upon (i) rewiring the graph via a variety of indices from network science and then (ii) running DeepWalk. Despite the simplicity of this idea, we show that our method improves the performance of DeepWalk on several benchmark datasets.

D.B. was supported by a Yale-Boehringer Ingelheim Biomedical Data Science Fellowship. S. Krishnaswamy received funding from the NIH (grants R01GM135929, R01GM130847, and R01HD100035), National Science Foundation Career Grant (2047856), and the Sloan Fellowship (grant FG-2021-15883). This project has been made possible in part by grant number 2019- 202662 from the Chan Zuckerberg Foundation.

II. Overview of Method

A. Notation

We let $G = (V, E)$ denote an unweighted, undirected graph with vertices $V = \{1, \dots, N\}$. We let A denote the adjacency matrix with entries $a_{xy} = 1$ if $(x, y) \in E$ and $a_{xy} = 0$ otherwise. For $x \in V$, we let $N(x)$ denote the neighbors of x, i.e., $N(x) = \{y : (x, y) \in E\}$, and we let $|N(x)|$ denote the degree of x.

B. The WireWalk algorithm

The WireWalk algorithm is based upon rewiring the graph. In particular, given an undirected, unweighted graph $G = (V, E)$ and a bias $\Pi : V \times V \to \mathbb{R}_{\geq 0}$, we define a new, directed graph G' with adjacency matrix $(\pi_{x,y})_{x,y \in V}$ where $\pi_{x,y} = \Pi(x, y)$. We then run the well-known DeepWalk [5] algorithm on G'.

The DeepWalk algorithm is based on extracting information about the vertices of G by running simple random walks on the graph and counting which vertices frequently appear in the same walk. In effect, WireWalk runs perturbed random walks where the walker takes steps which are biased by Π. In our experiments, we use a variety of bias functions as described below.

1) Common Neighbors:

$$\pi_{xy} = |N(x) \cap N(y)|$$

2) Salton Index:

$$\pi_{xy} = \frac{|N(x) \cap N(y)|}{\sqrt{|N(x)||N(y)|}}$$

3) Jaccard Index:

$$\pi_{xy} = \frac{|N(x) \cap N(y)|}{|N(x) \cup N(y)|}$$

4) Sorenson Index:

$$\pi_{xy} = \frac{2|N(x) \cap N(y)|}{|N(x)| + |N(y)|}$$

5) Hub Promoted Index:

$$\pi_{xy} = \frac{|N(x) \cap N(y)|}{\min\{|N(x)|, |N(y)|\}}$$

6) Hub Depressed Index:

$$\pi_{xy} = \frac{|N(x) \cap N(y)|}{\max\{|N(x)|, |N(y)|\}}$$

7) Leicht-Holme-Newman Index:

$$\pi_{xy} = \frac{|N(x) \cap N(y)|}{|N(x)||N(y)|}$$

8) Preferential Attachment:

$$\pi_{xy} = |N(x)||N(y)|$$

9) Adamic–Adar Index:

$$\pi_{xy} = \sum_{z \in N(x) \cap N(y)} \frac{1}{\log|N(z)|}$$

10) Resource Allocation Index:

$$\pi_{xy} = \sum_{z \in N(x) \cap N(y)} \frac{1}{|N(z)|}$$

11) Tversky Index:

$$\pi_{xy} = \frac{|N(x) \cap N(y)|}{|N(x) \cup N(y)| + \alpha|N(x) \setminus N(y)| + \beta|N(y) \setminus N(x)|}$$

Algorithm 1: WireWalk$(G, \pi, w, d, \gamma, t)$

Input: Graph $G = (V, E)$
bias Π, embedding size d
additional DeepWalk parameters w, γ, t
Output: Matrix of vertex representations $\Phi \in \mathbb{R}^{N \times d}$

1) Rewire G using the bias Π to obtain directed graph G'
2) DeepWalk(G', w, d, γ, t)

III. Results and Discussion

To evaluate WireWalk we consider the task of node classification on the BlogCatalog [6], PPI (Homo Sapiens) [1], and POS Wikipedia [3] datasets and evaluate performance according to Macro-F1 score, Micro-F1 score, the Fowlkes Mallows index, and overall accuracy. The method column indicates the bias Π used before running DeepWalk (except for the DeepWalk row which is the results without any preprocessing). On all three datasets, we observe that the rewiring steps improve performance according to the Micro-F1 score, the Fowlkes Mallows index, and overall accuracy. Curiously, performance actually decreases with respect to the Macro-F1 index indicating that our method may not be applicable in settings where one is highly concerned about imbalances in class sizes.

Method	Macro-F1	Micro-F1	Fowlkes Mallows	Accuracy
Jaccard Index	0.0065	0.1380	0.2366	0.1380
Common Neighbors	0.0076	0.1359	0.2267	0.1359
Sorenson Index	0.0065	0.1380	0.2366	0.1380
Salton Index	0.0065	0.1380	0.2366	0.1380
Hub Depressed	0.0065	0.1380	0.2366	0.1380
Hub Promoted	0.0065	0.1377	0.2363	0.1377
Preferential Attachment	0.0072	0.1339	0.2274	0.1339
L.H.N. Index	0.0080	0.1337	0.2215	0.1337
DeepWalk	0.0179	0.1141	0.1514	0.1141

TABLE I
Evaluation on BlogCatalog graph

Method	Macro-F1	Micro-F1	Fowlkes Mallows	Accuracy
Jaccard Index	0.0021	0.0483	0.1534	0.0483
Adamic-Adar Index	0.0042	0.0550	0.1289	0.0550
Resource Allocation Index	0.0031	0.0447	0.1253	0.0447
Common Neighbors	0.0033	0.0478	0.1377	0.0478
Sorenson Index	0.0032	0.0504	0.1535	0.0504
Salton Index	0.0039	0.0463	0.1234	0.0463
Hub Depressed	0.0029	0.0478	0.1454	0.0478
Hub Promoted	0.0034	0.0416	0.1278	0.0416
Preferential Attachment	0.0040	0.0478	0.1236	0.0478
L.H.N. Index	0.0034	0.0491	0.1329	0.0491
DeepWalk	0.0177	0.0288	0.0330	0.0288

TABLE II
Evaluation on PPI (Homo Sapiens)

In all of our experiments, the parameters w, d, γ, and t used for WireWalk are chosen based on typical values used for DeepWalk [2]. Specifically, we set $w = 10, d = 128, \gamma = 10$, and $t = 80$. Moreover, for the Tversky index, we set $\alpha = |N(x)|$ and $\beta = |N(y)|$. The node feature representations are input to a one-vs-rest logistic regression classifier with L2 regularization. The train and test data is split using stratified 10-fold cross-validation.[1]

IV. Conclusion

We have introduced WireWalk, a novel method for improving the performance of skip-gram based methods such as DeepWalk by rewiring the graph in accordance to a variety of network-science indices and show that our method improves the performance of DeepWalk on several benchmark datasets. For the sake of simplicity, in our experiments, we simply run DeepWalk on the transformed graph after the preprocessing

[1]Code available at https://github.com/TesfaAsmara/wirewalk

Method	Macro-F1	Micro-F1	Fowlkes Mallows	Accuracy
Jaccard Index	0.0289	0.4702	0.5152	0.4702
Common Neighbors	0.0300	0.4687	0.5141	0.4687
Sorenson Index	0.0289	0.4702	0.5152	0.4702
Salton Index	0.0313	0.4710	0.5163	0.4710
Hub Depressed	0.0289	0.4702	0.5152	0.4702
Hub Promoted	0.0289	0.4702	0.5153	0.4702
Preferential Attachment	0.0289	0.4702	0.5155	0.4702
L.H.N. Index	0.0289	0.4702	0.5152	0.4702
Tversky Index	0.0297	0.4656	0.5078	0.4656
DeepWalk	0.0324	0.4651	0.5070	0.4651

TABLE III
Evaluation on POS Wikipedia graph

step. However, we note that our method could also be combined with other methods (after making adjustments to deal with the fact that the transformed graph is directed). In this work, so far, we have focused on relatively small datasets and therefore allow the transformed graph to be dense. However, in future work, one could develop a more scalable version of our method by requiring the transformed graph to have the same sparsity pattern as the original.

REFERENCES

[1] Bobby-Joe Breitkreutz, Chris Stark, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, Michael Livstone, Rose Oughtred, Daniel H Lackner, Jürg Bähler, Valerie Wood, et al. The biogrid interaction database: 2008 update. *Nucleic Acids Research*, 36:D637–D640, 2007.

[2] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

[3] Matt Mahoney. Large text compression benchmark, 2011.

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

[5] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

[6] Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 817–826, 2009.