COL868 : Benchmarking - Node2Vec and Deepwalk

Nilaksh Agarwal
Indian Institute of Technology, Delhi
ph1150813@iitd.ac.in

Nihar Modi
Indian Institute of Technology, Delhi
ee1170461@iitd.ac.in

Abstract

Node2Vec and Deepwalk have revolutionized the field of Graphical Networks. In this paper, we present the benchmarking results for several prediction tasks such as Pair-wise node classification, Link Prediction, and Multi-class node classification using Node2Vec and Deepwalk models. We used Brightkite dataset, was once a location-based social networking service provider, protein protein interaction(PPI) dataset and Protein dataset, for the benchmarking tasks. Our results show that Node2vec outperforms Deepwalk across all the tasks, however most of the current SotA algorithms are significantly better than both Node2vec and Deepwalk.

1 Introduction

Graphical networks can depict many complex systems involving biological, social and informational connections between entities. At the most abstract level, these networks are modelled by graphs in which nodes represent individuals or agents and links denote the interactions or relationships between nodes. Structural properties of biological networks are of great interest as they directly correlate with biological function (Qi and Ge [7]; Wuchty et al. [11]).
Here we benchmark some tasks on biological networks PPI and protein and on location based network Brightkite using Node2Vec (Grover and Leskovec [3]) and Deepwalk (Perozzi et al. [6]).
There are a number of standard tasks which Graph Neural Networks are used to perform, including pair-wise node classification, link prediction and multi-label node classification.

Figure 1: The Color coded communities exhibiting homophily discovered by node2vec in the Les Misérables Network [3]

2 Related Works

This benchmarking assignment is being undertaken by 4 other groups of students, working on different models such as Struct2vec ([8]), Position Aware GNNs ([12]), Graph Attention Networks ([9]), Graph Convolutional Networks ([5]) and GraphSage ([4]).

Code: github.com/nilax97/col868benchmark

3 Datasets

We used the following 3 datasets for benchmarking:

Protein	(Borgwardt et al. [1])
Protein Protein Interaction (PPI)	(Zitnik and Leskovec [13])
Brightkite	(Cho et al. [2])

3.1 PPI

24 Protein-protein interaction networks from Zitnik and Leskovec [13]. Each graph has 3000 nodes with avg. degree 28.8, each node has 50 dimensional feature vector. Protein-protein interactions (PPIs) are essential to almost every process in a cell, so understanding PPIs is crucial for understanding cell physiology in normal and disease states. It is also essential in drug development, since drugs can affect PPIs. Protein-protein interaction networks (PPIN) are mathematical representations of the physical contacts between proteins in the cell.
We use the this dataset for the multilabel node classification task and the link prediction task.

3.2 Protein

1113 protein graphs from Borgwardt et al. [1]. Each node is labeled with a functional role of the protein. Each node has a 29 dimensional feature vector.
We use the this dataset for the pairwise node classification task

3.3 Brightkite

Brightkite from Cho et al. [2] was once a location-based social networking service provider where users shared their locations by checking-in. The friendship network was collected using their public API, and consists of 58,228 nodes and 214,078 edges. The network is originally directed but they have constructed a network with undirected edges when there is a friendship in both ways. We have also collected a total of 4,491,143 check-ins of these users over the period of Apr. 2008 - Oct. 2010
We use the this dataset for the link prediction task.

	Protein	PPI	Brightkite
# Nodes	43,471	56,944	58,228
# Edges	162,088	818,716	214,078
# Classes	3	121 (Multilabel)	-
# Features	29	50	-
Task	I	II, III	II

Table 1: Summary of the datasets used in our experiments

4 Prediction Tasks

In this section, we describe in detail the benchmark prediction tasks that we have carried out, as well as the generation of the task-specific dataset.

4.1 Pair-wise node classification

Given two nodes, predict if they belong to the same class. This task aims to showcase the similarity of node features/neighbourhoods which influence the class of the node.

We have used the Protein [1] dataset for this task. Our total dataset size was 400,000, which was split for 5-fold cross validation. This was generated by random node-pair sampling equally from the positive samples (both nodes have same class label) and negative samples. The weightage of each class (Class 0, Class 1 & Class 2) was proportionate to the number of nodes in that class.

4.2 Link Prediction

To predict if there is an edge between two nodes. In link prediction tasks two nodes are generally more likely to form a link, if they are close together in the graph.
We have used the PPI dataset [13] and the Brightkite dataset [2] for this task. Our total dataset size was 2 times the number of edges in the respective datasets. The negative samples were generated by random sampling of node pairs which don't have edges between them.

4.3 Multi-class node classification

To predict the class labels for a given node. For the multi-class/ multi-label setting, each node has an associated class label vector. We have used the PPI dataset [13] for this task. Each node had an associated 121-dimension binary class label vector, where multiple values can be 1. On average, a node has 37 class labels active (Range 0-101).

5 Experimental Setup

In this section we outline the scoring methods, implementation details, aggregating functions for both our models and tasks.

5.1 Scoring Methods

ROC AUC: Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.
Precision: It is the ratio TP / (TP + FP) where TP is the number of true positives and FP the number of false positives. The precision is intuitively the ability of the classifier for successful prediction. The best value is 1 and the worst value is 0.
Recall: It is the ratio TP / (TP + FN) where TP is the number of true positives and FN the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples. The best value is 1 and the worst value is 0.
F1 score: By itself, neither Precision nor Recall are able to capture the classifiers complete performance. So, the F1 score is used to convey the balance between the precision and recall. It is the harmonic mean of Precision and Recall.

5.2 DeepWalk

DeepWalk by Perozzi et al. [6] is an algorithm that is used to create embeddings of the nodes in a graph. The embeddings are meant to encode the community structure of the graph. It achieves this by using SkipGram to create the embeddings.
The method used to make predictions is skip-gram, just like in Word2vec architecture for text. Instead of running along the text corpus, DeepWalk runs along the graph to learn an embedding. The model can take a target node to predict it's "context", which in the case of a graph, means it's connectivity, structural role, and node features.
Although DeepWalk is relatively efficient with a score of $O(|V|)$, this approach is transductive, meaning whenever a new node is added, the model must be retrained to embed and learn from the new node.

5.3 Node2vec

Node2vec by Grover and Leskovec [3] is an algorithmic framework for representational learning on graphs. Given any graph, it can learn continuous feature representations for the nodes, which can then be used for various downstream machine learning tasks.
The node2vec framework learns low-dimensional representations for nodes in a graph by optimizing a neighborhood preserving objective. The objective is flexible, and the algorithm accommodates for various definitions of network neighborhoods by simulating biased random walks. Specifically, it provides a way of balancing the exploration-exploitation tradeoff that in turn leads to representations obeying a spectrum of equivalences from homophily to structural equivalence.

5.4 Differences

The difference between Node2vec and DeepWalk is subtle but significant. Node2vec features a walk bias variable α, which is parameterized by p and p. The parameter p prioritizes a breadth-first-search (BFS) procedure, while the parameter p prioritizes a depth-first-search (DFS) procedure. The decision of where to walk next is therefore influenced by probabilities $\frac{1}{p}$ or $\frac{1}{q}$.

BFS is ideal for learning local neighbors, while DFS is better for learning global variables. Node2vec can switch to and from the two priorities depending on the task. This means that given a single graph, Node2vec can return different results depending on the values of the parameters. As per DeepWalk, Node2vec also takes the latent embedding of the walks and uses them as input to a neural network to classify nodes.

Experiments demonstrated that BFS is better at classifying according to structural roles (hubs, bridges, outliers, etc.) while DFS returns a more community driven classification scheme.

Parameters	Range	Default Value
No. of walks	10,30,100,300	10
Embedding size	32,64,128,256	128
Walk Length	5,15,50,80,100	80
Window size - Skipgram	5,10,20	10
Iters of SGD	1,5,10,50,100	1
Return hyperparameter	0.1,0.5,1,2,10	1
Inout hyperparameter	0.1,0.5,1,2,10	1

Table 2: Paramters variation for Node2vec

Parameters	Range	Default
No. of walk	10,30,100,300	10
Embedding size	32,64,128,256	64
Walk Length	5,15,50,100	40
Window size for Skipgram	5,10,20	5

Table 3: Paramters variation for DeepWalk

5.5 Implementation details

We use the standard node2vec implementation available at github.com/aditya-grover/node2vec, implemented in Python2.7 and the standard DeepWalk implementation available at github.com/phanein/deepwalk, implemented in Python3.7

We perform extensive hyperparameter tuning to decide the optimal parameters for this model. A summary of the parameters used are summarised in Table 2 and Table 3.

Operator	Symbol	Definition
Average	$\boxplus$	$\frac{f_i(u)+f_i(v)}{2}$
Hadamard	$\boxdot$	$f_i(u) * f_i(v)$
Weighted-L1	$\|\|\cdot\|\|_{\tilde{1}}$	$\|f_i(u) - f_i(v)\|$
Weighted-L2	$\|\|\cdot\|\|_{\tilde{2}}$	$\|f_i(u) - f_i(v)\|^2$

Table 4: Choice of operators for learning edge features

For Task I (Pairwise Node Classification) and Task II (Link Prediction), as per [3] we try out 4 aggregator functions shown in Table 4 to join the embeddings of the two nodes and these are passed to a secondary classifier (using Logistic Regression as per [3]).
For Task III (Multi-Label Classification) we train multiple one-vs-rest logistic regression classifiers (as per [3]).
Additionally, we experiment with various other classifiers (Linear SVM Classifier, Extra Trees Classifier, Bagging Classifier, Gradient Boosting Classifier, AdaBoost Classifier and MLP Classifier). For all of them we use the sklearn-implementation.

6 Results

All results are averaged over 5-fold cross validation. The training specifications are as per those mentioned previously.

6.1 Pair-wise node classification

Network	Dataset	ROC	Precision	Recall	F_1
GCN[5]	Protein	0.515	-	-	-
GraphSAGE[4]	Protein	0.520	-	-	-
GAT[9]	Protein	0.528	-	-	-
GIN[10]	Protein	0.523	-	-	-
P-GNN[12]	Protein	**0.729**	-	-	-
Node2Vec[3]	Protein	0.604	0.600	0.625	0.612
Deepwalk[6]	Protein	0.557	0.555	0.567	0.561

Table 5: Pairwise Node Classification on Protein Dataset

Node2vec outperforms Deepwalk, and they both outperform other networks such as GCN, GraphSAGE, GAT, GIN etc.
However, this may be due to these networks performing this task in the inductive setting whereas Node2vec and Deepwalk use the transductive setting.
Hence, since no node/graph is technically "unseen", they can perform better than the indcutive setting algorithms

6.2 Link Prediction

Network	Dataset	ROC	Precision	Recall	F_1
GCN[5]	PPI	0.769	-	-	-
GraphSAGE[4]	PPI	0.803	-	-	-
GAT[9]	PPI	0.783	-	-	-
GIN[10]	PPI	0.782	-	-	-
P-GNN[12]	PPI	**0.808**	-	-	-
Node2Vec[3]	PPI	0.557	0.557	0.559	0.558
Deepwalk[6]	PPI	0.553	0.549	0.596	0.571
Node2Vec[3]	BrightKite	**0.813**	0.844	0.769	0.804
Deepwalk[6]	BrightKite	0.742	0.728	0.773	0.750

Table 6: Link Prediction on PPI and Brightkite Dataset

Node2vec outperforms Deepwalk, however they are both beaten by other networks such as GCN, GraphSAGE, GAT, GIN, P-GNN etc. This may be due to the fact that Node2vec and DeepWalk don't use any node features during their training, and rely only on the neighbourhood of the node

6.3 Multi-Label Node Classification

Node2vec outperforms Deepwalk, however they are both beaten by other networks such as GraphSAGE, GAT etc.
Since PPI has over 121 distinct labels per node, it is difficult to predict each and every one of them accurately with only the neighbourhood information of the node

Network	Dataset	Precision	Recall	Micro F_1
Random	PPI	-	-	0.396
MLP	PPI	-	-	0.422
GraphSAGE[4]	PPI	-	-	0.768
GAT[9]	PPI	-	-	**0.973**
Node2Vec[3]	PPI	0.419	0.609	0.479
Deepwalk[6]	PPI	0.363	0.594	0.431

Table 7: Node Classification on PPI Dataset

7 Variations and Improvements

This section serves to highlight certain experiments and variations to the models and parameters done by us.

7.1 Classifiers

Apart from the standard logistic regression classifier mentioned in the papers, we experimented with a couple of other classifiers and ensembles. RandomForests with 100 trees gives us the best results for both methods.

Network	Classifier	ROC	Precision	Recall	F_1
Node2vec	Logistic	0.604	0.6	0.625	0.612
	LinearSVC	0.604	0.6	0.626	0.612
	ExtraTrees	0.625	0.619	0.65	0.634
	Bagging	0.559	0.575	0.455	0.508
	RandomForest	**0.626**	0.62	0.654	0.636
	GradientBoost	0.612	0.602	0.661	0.63
	AdaBoost	0.57	0.568	0.582	0.575
Deepwalk	Logistic	0.557	0.555	0.567	0.561
	LinearSVC	0.557	0.556	0.568	0.562
	ExtraTrees	0.575	0.579	0.546	0.562
	Bagging	0.536	0.548	0.418	0.474
	RandomForest	**0.577**	0.579	0.562	0.57
	GradientBoost	0.561	0.559	0.577	0.568
	AdaBoost	0.537	0.537	0.537	0.537

Table 8: Variation across Classifiers for Task I

7.2 Aggregator Functions

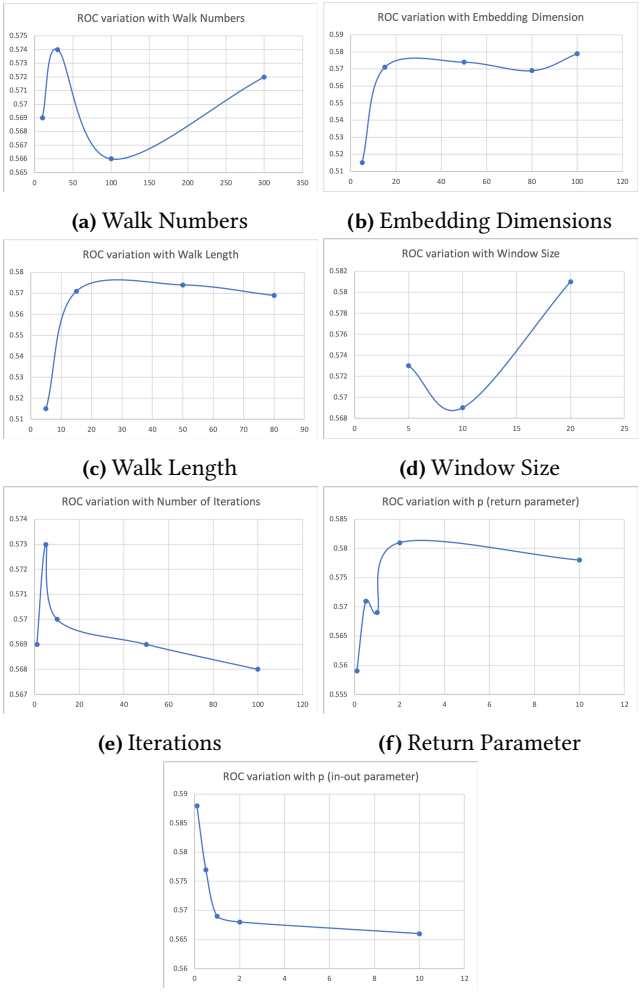
The Average binary operator performs the best among the four aggregator functions. This might be since the Hadamard operator supresses a dimension even if one operand is zero in the same, and the L1 and L2 values capture only the relative distance between the operands however are unable to incorporate the position of the operands in a global coordinate.

Network	Operator	ROC	Precision	Recall	F_1
Node2Vec	Average	**0.604**	0.600	0.625	0.612
	Hadamard	0.561	0.561	0.566	563
	Weighted-L1	0.555	0.555	0.563	0.559
	Weighted-L2	0.557	0.556	0.567	0.561
Deepwalk	Average	**0.557**	0.555	0.567	0.561x
	Hadamard	0.517	0.517	0.516	0.516
	Weighted-L1	0.531	0.531	0.530	0.530
	Weighted-L2	0.531	0.531	0.533	0.532

Table 9: Variation across Binary Aggregator Functions for Task I

7.3 Parameters

This section contains a variation analysis of both methods varying some common hyper-parameters of the models for Task I.



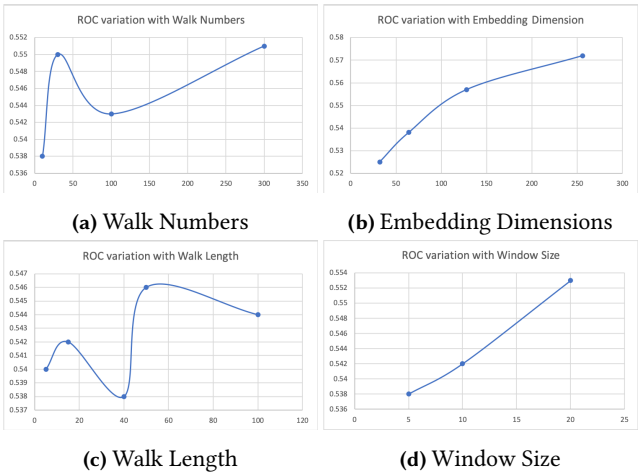
(a) Walk Numbers **(b)** Embedding Dimensions

(c) Walk Length **(d)** Window Size

(e) Iterations **(f)** Return Parameter

(g) In-Out Parameter

Figure 2: The results for Parameter Variation on Node2vec



(a) Walk Numbers **(b)** Embedding Dimensions

(c) Walk Length **(d)** Window Size

Figure 3: The results for Parameter Variation on DeepWalk

8 Conclusion

From this benchmarking excercise, we observe Node2vec has an improvement over DeepWalk in all 3 tasks, i.e., Pairwise Node Classification, Link Predicion and Multi-Label Node Classification. This is since Node2vec looks at both local and global neighbourhoods through it's combination of BFS and DFS. It is therefore able to learn a better context for the node as compared to Deepwalk.
We also would like to point out that DeepWalk was released in 2014, and Node2vec in 2016. Moreover, since then there have been some significant improvements in GNNs which have vastly improved the results on these three tasks.

References

[1] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. 2005. Protein Function Prediction via Graph Kernels. *Bioinformatics* 21, 1 (Jan. 2005), 47–56. https://doi.org/10.1093/bioinformatics/bti1007

[2] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1082–1090.

[3] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[4] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[5] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[6] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, New York, NY, USA, 701–710. https://doi.org/10.1145/2623330.2623732

[7] Yuan Qi and Hui Ge. 2006. Modularity and dynamics of cellular networks. *PLoS computational biology* 2, 12 (2006).

[8] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 385–394.

[9] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[10] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 968–977.

[11] Stephan Wuchty, Zoltán N Oltvai, and Albert-László Barabási. 2003. Evolutionary conservation of motif constituents in the yeast protein interaction network. *Nature genetics* 35, 2 (2003), 176–179.

[12] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware graph neural networks. *arXiv preprint arXiv:1906.04817* (2019).

[13] Marinka Zitnik and Jure Leskovec. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* 33, 14 (2017), 190–198.