UNSUPERVISED AND SCALABLE ALGORITHM FOR LEARNING NODE REPRESENTATIONS

Tiago Pimentel[1], **Adriano Veloso**[1] **and Nivio Ziviani**[1,2]
[1]CS Dept, Universidade Federal de Minas Gerais (UFMG) & [2]Kunumi
{tpimentel,adrianov,nivio}@dcc.ufmg.br, nivio@kunumi.com

ABSTRACT

Representation learning is one of the foundations of Deep Learning and allowed big improvements on several Machine Learning fields, such as Neural Machine Translation, Question Answering and Speech Recognition. Recent works have proposed new methods for learning representations for nodes and edges in graphs. In this work, we propose a new unsupervised and efficient method, called here Neighborhood Based Node Embeddings (NBNE), capable of generating node embeddings for very large graphs. This method is based on SkipGram and uses nodes' neighborhoods as contexts to generate representations. NBNE achieves results comparable or better than state-of-the-art feature learning algorithms in three different datasets and, differently from our main baseline (Node2Vec), which needs to have its parameters tuned in a validation set, is completely unsupervised.

1 INTRODUCTION

Many important problems involving graphs require the use of learning algorithms to make predictions about nodes and edges. These predictions and inferences in nodes and edges from a graph are, typically, done using classifiers with carefully engineered features (Grover & Leskovec, 2016). These features, besides taking time and costs to be developed and acquired, usually do not generalize well to other problems or contexts.

The field of Natural Language Processing (NLP) has had many advances due to the use of algorithms that learn word representations, instead of (or together with) manually extracted features. Recent works have proposed new methods for learning representations for nodes and edges in graphs, based on random walks (Perozzi et al., 2014; Grover & Leskovec, 2016) or autoencoding adjacency vectors (Wang et al., 2016).

We introduce the most relevant work in detail: (1) **Deep Walk** (Perozzi et al., 2014): Random walks on the graph are used to generate sentences, where each "word" is a node, which are used with a SkipGram (Mikolov et al., 2013a) algorithm to generate node embeddings. (2) **Node2Vec** (Grover & Leskovec, 2016): Also uses random walks with SkipGram and can be seen as a generalization of Deep Walk. The difference between the two methods is that Node2Vec's walks are random, but biased by two pre-assigned parameters p and q. During the creation of the walks, these parameters are used to increase the chance of the walk returning to a parent node or going farther from it. This method is also scalable to large graphs, but this is a semi-supervised approach which requires several models to be generated and a small sample of labelled nodes to be used so that the best parameters p and q can be chosen.

In this work, we propose a new method for generating node embeddings in graphs, based on the Word2Vec's algorithm SkipGram and using nodes neighborhoods as contexts. We claim our proposed method, called here Neighborhood Based Node Embeddings (NBNE), achieves: (a) a faster training time than current state-of-the-art methods (i.e. Deep Walk (Perozzi et al., 2014) and Node2Vec (Grover & Leskovec, 2016)); (b) general purpose embeddings, differently from our baseline Node2Vec, which tunes its embeddings for specific tasks; (c) state-of-the-art results in the two different tasks examined here: Link prediction and Node Classification; (d) the use of a unique parameter (number of permutations, which will be further explained later) that can be tuned at once on the training set to avoid overfitting the representations.

2 NEIGHBORHOOD BASED NODE EMBEDDINGS

In NLP, a word context is a simple concept, it is the words surrounding it. In graphs, a node context is a more complex concept. While DeepWalk and Node2Vec use random walks as sentences (or contexts), in this work the sentences are based solely on nodes' neighborhoods, focusing on second-order proximities. Nodes with similar neighborhoods, then, will have similar representations.

2.1 SENTENCE GENERATION

NBNE sentences are created using nodes' neighborhoods. However, there are two main issues: (a) for a specific highly connected root node, most neighbors would be distant from each other in the neighborhood sentence, not influencing each other's representations, and not directly influencing the root node; (b) there is no explicit order in the nodes in a neighborhood.

Our solution – Using each node as root once, we divide random permutations of these root nodes' neighborhoods in small sentences, with only k neighbors in each. Each generated sentence has the form: $[v_{root} \quad v_1 \quad v_2 \ldots \quad v_k], \exists\, edge_{root,i} \in E, \forall 1 \leq i \leq k$. As a trade off between training time and increasing the training dataset, the user can select a parameter $n = num\_permutations$. Selecting a higher value for n creates a more uniform distribution on possible neighborhood sentences, but increases training time.

2.2 LEARNING REPRESENTATIONS

As mentioned previously, SkipGram (Mikolov et al., 2013a) was used to learn vector representations for graph nodes. For this, it used the sentences created as described in Section 2.1, which form a set S of sentences. SkipGram learns a model by maximizing the log probability of predicting a node given another node within a distance of at most k and a set of representations r. The log probability maximized by NBNE is shown in Eq. 1.

$$\max_{r} \quad \frac{1}{|S|} \cdot \sum_{s \in S} \left(\frac{1}{|s|} \cdot \sum_{t=1}^{|s|} \left(\sum_{-k \leq j \leq k, j \neq 0} \left(log\left(p\left(v_{t+j} | v_t, r \right) \right) \right) \right) \right) \tag{1}$$

In this equation, v_t is a vertex in the graph and v_{t+j} will be the other vertices in the same sentence. The probabilities in this model are learned using the feature vector r_{v_i}, which are then used as the vertex representations. The probability $p\left(v_{t+j} | v_t, r\right)$ can, then, be rewritten as shown in Eq. 2, where $r'^{T}_{v_j}$ is the transposed output feature vector of vertex j, used to make predictions. r'_v and r_v are learned simultaneously by optimizing Eq. 1, which is done using stochastic gradient ascent with negative sampling (Mikolov et al., 2013b).

$$p\left(v_o | v_i, r\right) = \frac{exp\left(r'^{T}_{v_o} r_{v_i} \right)}{\sum_{v \in V} \left(exp\left(r'^{T}_{v} r_{v_i} \right) \right)} \tag{2}$$

By optimizing Eq. 2, the algorithm maximizes the predictability of a neighbor given a node, creating node embeddings where nodes with similar neighborhoods have similar representations.

Complexity – SkipGram's complexity is linear on the number of sentences, embedding size (d) and logarithmic on the size of the vocabulary (Mikolov et al., 2013a). Since the number of sentences is linear on the number of permutations (n), branching factor of the graph (b) and on the number of nodes, which is the size of the vocabulary ($|V|$), the algorithm takes a time bounded by $O\left(d \cdot n \cdot b \cdot |V| \cdot log(|V|)\right)$.

3 EXPERIMENTS AND RESULTS

NBNE was tested in two different tasks: link prediction and node classification. On node classification, we tested our embeddings by running NBNE on the entire graph and then training a logistic classifier on 80% of the resulting embeddings to predict its classes, while using the other 20% as

a test set. A blog dataset (Zafarani & Liu, 2009) was used, where nodes were bloggers and edges were friendships between them. This graph contains 10,312 nodes and 333,983 edges, with each node having one class out of 39 possible ones.

On link prediction, we proposed to test our embeddings by trainning a logistic classifier on the concatenation of the embeddings from both nodes that possibly form an edge and predict the existence or not of the edge. Two datasets were used to evaluate NBNE's performance in this task: (1) Facebook (McAuley & Leskovec, 2012): nodes represent users and edges represent friendships. Contains 4,039 nodes and 88,234 edges; (2) Astro (Leskovec et al., 2007): a collaboration network which covers scientific collaborations between authors whose papers were submitted to the Astro Physics category in Arxiv. Contains 18,772 nodes and 198,110 edges.

To train NBNE on this task, we first obtained a sub-graph with 90% randomly select edges from each dataset, and obtained the node embeddings by training NBNE on this sub-graph. We, then, trained a logistic regression with these sub-graph edges and these trained embeddings. After the training was complete, the other 10% of the edges were used as a test set to predict new links.

The results[1] are shown in Table 1. For all these experiments we used sentences of size $k = 5$ and word embeddings of size $d = 128$. On the Blog dataset, which had more edges per node, we used $n = 10$, on Facebook, we used $n = 5$, while on the Astro dataset, since the number of edges per node was only 10.55, we used only one permutation $n = 1$.

Table 1: Results on Facebook, Astro and Blog datasets

	Facebook		Astro		Blog	
	Precision	AUC	Precision	AUC	Precision	Macro F1
NBNE	0.9070	0.9688	**0.7552***	**0.8328***	**0.3290**	**0.2004**
DeepWalk	0.9218	0.9730	0.6836*	0.7548*	0.3108*	0.1451*
Node2vec	**0.9251****	**0.9762****	0.7163*	0.7738*	0.3257	0.1637*
Gain	-1.95%	-0.76%	5.43%	7.62%	1.00%	22.45%

* statistically significant with $p = 0.01$

** statistically significant with $p = 0.05$

On Facebook, the results show a slightly smaller AUC and precision value while using NBNE's embeddings, although these differences are not statistically significant when comparing NBNE and Deep Walk. Table 1 also shows that, on Astro, NBNE's results are considerably better than DeepWalk's and Node2Vec's. We suspect that NBNE achieved such an improvement on Astro, while a small decrease against Node2Vec on Facebook, due to the results on Facebook being "saturated". Node2Vec trains 25 models, with its parameters as suggested in its original work, for each fold in the cross-validation, and then chooses the best among them. In the Facebook dataset, no pattern to its choice on the best model was found by us, so, choosing randomly one out of 25 models gives it an advantage. On Blog, NBNE's Macro F1 score was considerably better than both Deep Walk's and Node2Vec's. This indicates that NBNE's embeddings didn't only get a better accuracy, but the correct answers were also better distributed across the 39 possible classes.

4 Conclusion

The new node embedding method NBNE shows results similar or better than the state-of-the-art algorithms Deep Walk and Node2Vec. It shows promising results in the two applications considered here: link prediction and node classification, while being efficient and easy to compute for large graphs, having a faster training time than Node2Vec unless if setting $n > 1000$.

5 Acknowledgments

This work was partially funded by projects InWeb (grant MCT/CNPq 573871/2008- 6) and MASWeb (grant FAPEMIG/PRONEX APQ-01400-14), and by the authors individual grants from Kunumi, CNPq and FAPEMIG.

[1]Results using 10-fold cross-validation. DeepWalk and Node2Vec were trained and tested under these same conditions and using the parameters as proposed in (Grover & Leskovec, 2016).

REFERENCES

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*, KDD '16, pp. 855–864, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939754. URL http://doi.acm.org/10.1145/2939672.2939754.

Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *TKDD*, 1(1):2, 2007. doi: 10.1145/1217299.1217301. URL http://doi.acm.org/10.1145/1217299.1217301.

Julian J McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In *NIPS*, volume 2012, pp. 548–56, 2012.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pp. 3111–3119, 2013b.

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*, pp. 701–710. ACM, 2014.

Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 1225–1234, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939753. URL http://doi.acm.org/10.1145/2939672.2939753.

R. Zafarani and H. Liu. Social computing data repository at ASU, 2009. URL http://socialcomputing.asu.edu.