Similarity-Aware Network Embedding with Self-Paced Learning

Chao Huang[*]
University of Notre Dame
chuang7@nd.edu

Baoxu Shi[*]
LinkedIn
bshi@nd.edu

Xuchao Zhang
Virginia Tech
xuczhang@vt.edu

Xian Wu
University of Notre Dame
xwu9@nd.edu

Nitesh V. Chawla
University of Notre Dame
nchawla@nd.edu

ABSTRACT

Network embedding, which aims to learn low-dimensional vector representations for nodes in a network, has shown promising performance for many real-world applications, such as node classification and clustering. While various embedding methods have been developed for network data, they are limited in their assumption that nodes are correlated with their neighboring nodes with the same similarity degree. As such, these methods can be suboptimal for embedding network data. In this paper, we propose a new method named *SANE*, short for *Similarity-Aware Network Embedding*, to learn node representations by explicitly considering different similarity degrees between connected nodes in a network. In particular, we develop a new framework based on self-paced learning by accounting for both the explicit relations (*i.e.*, observed links) and implicit relations (i.e., unobserved node similarities) in network representation learning. To justify our proposed model, we perform experiments on two real-world network data. Experiments results show that *SNAE* outperforms state-of-the-art embedding models on the tasks of node classification and node clustering.

CCS CONCEPTS

• **Theory of computation → Social networks**; • **Computing methodologies → Knowledge representation and reasoning**; **Learning latent representations**.

KEYWORDS

Network Embedding; Deep Neural Network; Self-paced Learning

ACM Reference Format:
Chao Huang, Baoxu Shi, Xuchao Zhang, Xian Wu, and Nitesh V. Chawla. 2019. Similarity-Aware Network Embedding with Self-Paced Learning. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM'19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3357384.3358163

[*]These authors contributed equally.

CIKM'19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3358163

1 INTRODUCTION

Mining large-scale information networks (*e.g.*, social networks, academic networks) has benefited many real-world applications, such as friend recommendation and user classification in online social platforms [11]. In those applications, identifying effective features play a crucial role, however, involves huge amounts of human efforts and massive handcrafted feature engineering based on domain-specific knowledge. To tackle this issue, a line of research on network embedding [4, 8, 12], which aims to learn low-dimensional vector representations of nodes, has attracted a lot of attention. These methods have been shown to be effective in various network mining tasks (*e.g.*, node classification and clustering).

In particular, existing network embedding techniques aim to learn node representations by preserving the proximities between nodes based on their structural properties. To simplify the design, these approaches assumed the similarities between all pairs of connected nodes in a network with the same degree. However, this assumption does not hold in reality where nodes may have various degrees of similarity with their neighbors. Take the network between users on Twitter as an example, the followers of a movie star could come from: i) his/her friends who are most familiar with this movie star; ii) the people who share similar interests to the movie star, such as his/her colleagues; iii) or the ordinary people who are only interested in the movie star, but do not share similar interests and are personally unrelated. Such difference in node (user) similarity is expected to have a clear effect on network representation learning. This leads to the question investigated in this paper: can we explore different degrees of node similarity to learn more robust node representations for a network?

To solve the above question, we develop a new embedding framework called Similarity-Aware Network Embedding (SANE), aiming to learn robust node representations by considering different degrees of node similarity in a network. Since the similarity between different pairs of nodes may be different, each relationship between the target node and its neighbors should be treated differently during the representation learning process. In this work, we develop a self-paced learning algorithm to automatically learn the weights of node correlations in learning node representations. Finally, we perform extensive experiments on two real-world network datasets to evaluate the performance of our representation learning framework. Experimental results on both multi-label classification and node clustering show that SANE outperforms state-of-the-art network embedding methods. In summary, we highlight the contributions of this paper as follows:

- We propose to study the similarity-aware network representation learning, which aims to learn node representations by exploring different degrees of node similarity in a network.
- We propose a new framework SANE which is capable of preserving node proximity in terms of both observed network structure and unobserved node similarity for a consensus embedding representation in a robust way.
- We conduct experiments on two real-world network datasets. Experimental results on two tasks demonstrate the effectiveness of SANE model over competitive network embedding methods.

2 METHODOLOGY

2.1 SANE Model with Self-paced Learning

There exist several ways to infer the latent similarity degrees between different nodes, such as Expectation Maximization (EM) algorithm [2]. However, it heavily relies on the defined margin likelihood distribution and is very time consuming, which is not scalable to large-scale network data [6]. To address this issue, we propose to utilize the Self-Paced Learning (SPL) [6] to estimate the latent similarity degrees between nodes with no prior knowledge.

Inspired by the learning principle of humans: start with easier concepts and then gradually takes more complex examples into consideration [1], SPL, a learning algorithm, is proposed to mimic such training strategies to facilitate the learning process. In particular, SPL is proposed to quantify the easiness of each sample/data point as a hidden variable/weight. Given the training samples as $D = (x_1, y_1), ..., (x_n, y_n)$, where $x_r \in R^m$ denotes the i-th observed sample, and y_i represents i-th label, SPL learns the objectives by jointly learning model parameters and the easiness of each training data point (*i.e.*, hidden variable):

$$\arg\max_{\theta, v} \sum_i -w_i \cdot L(y, g(x_i, \theta)) + f(w_i, t) \tag{1}$$

where $L(y_i, g(x_i, \theta))$ denotes the loss function which calculates the cost between the real label y_i and the estimated label $g(x_i, \theta)$. w_i indicates the easiness of the observed i-th training sample ($w_i > 0$). $f(w_i, t)$ is the self-paced function which controls the pace to increase the difficulty level. The parameter t is the iteration index which is termed as "age" of the SPL model to control the learning pace. In particular, when t is small, the "easy" samples with small losses are considered. As t grows, more samples with larger losses will be gradually added into the training set.

In our work, we assume that a training node pair is proximate if their similarity is high. Motivated by SPL algorithm, we propose to first learn embedding vectors of nodes with higher similarity degree in order to avoid interfering by irrelevant neighbor nodes, then target at nodes with complex relationships between the already learned nodes. Incorporating the similarity degree into the SPL process is challenging due to dependencies existing between the similarity degrees of different node pairs. To incorporate the similarity degree of each training pair, the skip-gram based loss function could be defined as:

$$\arg\max_{\theta} \sum_{u \in \mathrm{V}} \sum_{c \in C(u)} \mathbf{w}_{u,v} \log(\Pr(c|u;\theta)) + f(u,t), \tag{2}$$

where $w_{u,v}$ represents the similarity degree of the training pair $\langle u, v \rangle$. However, the calculation on the similarities between center node u and other nodes in $\Pr(c|u;\theta) = \frac{exp(X_c, X_u)}{\sum_{u \in \mathrm{V}} exp(X_v, X_u)}$ is computational expensive[4]. Therefore, in our framework, we adopt the negative sampling technique to modify the conditional probability $\Pr(c|u;\theta)$ as follows:

$$\Pr(c|u;\theta) \propto \log(\sigma(X_c X_u)) + \sum_j^{k_{\text{neg}}} \mathbb{E}_{v_j \sim \text{Dist}(u)} \log(\sigma(-X_c X_u)) \tag{3}$$

By incorporating both the self-paced based skip-gram loss function and negative sampling strategy, we further give our loss functions as:

$$\mathcal{L} = \sum_{u \in \mathrm{X}} \sum_{p \in \mathrm{P}} \sum_i^{k_w} \mathbb{E}_{c_i \sim C_p(u)} \mathbf{w}_{c_i,u} \Big(\log(\sigma(\mathrm{X}_{c_i} \mathrm{X}_u^T)) + \sum_j^{k_{\text{neg}}} \mathbb{E}_{v_j \sim \text{Dist}(u)} \log(\sigma(-\mathrm{X}_{v_j} \mathrm{X}_u^T)) + \lambda_1 \lambda_2^{\kappa} \mathbf{w'}_{c_i,u} \Big),$$

where λ_1 and λ_2 represents the similarity degree learning strength and difficulty increasing rate, respectively.

2.2 Model Optimization

In this work, we apply stochastic gradient decent algorithm to derive the solutions by maximizing objective function Eq. 4. Specifically, we partition the variables into two disjoint sets. In each iteration, our algorithm optimize one set of variables and keep another set of variables fixed. In the learning process, when we fix similarity degree w, the skip-gram learning algorithm is employed to obtain the node embedding vectors which maximize the likelihood of preserving both the structural and similarity information in the network. As the number of iterations increases, more node training pairs with low similarity degree are fed into the learning process. Formally, we derive the gradients as follows:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{w}_u} &= \mathbf{w'}_{c_i} \log(\sigma(\mathrm{X}_{c_i} \mathrm{X}_u^T)) + \sum_j \mathbf{w'}_{v_j} \log(\sigma(-\mathrm{X}_{v_j} \mathrm{X}_u^T)) + \\
&\quad \lambda_1 \lambda_2^{\kappa} (\mathbf{w'}_{c_i} + \sum_j \mathbf{w'}_{v_j}), \\
\frac{\partial \mathcal{L}}{\partial \mathbf{w'}_{c_i}} &= \mathbf{w}_u \log(\sigma(\mathrm{X}_{c_i} \mathrm{X}_u^T)) + \lambda_1 \lambda_2^{\kappa} \mathrm{X}_u, \\
\frac{\partial \mathcal{L}}{\partial \mathbf{w'}_{v_j}} &= \mathbf{w}_u \log(\sigma(-\mathrm{X}_{v_j} \mathrm{X}_u^T)) + \lambda_1 \lambda_2^{\kappa} \mathrm{X}_u, \\
\frac{\partial \mathcal{L}}{\partial \mathrm{X}_u} &= \mathbf{w'}_{c_i} (1 - \sigma(\mathrm{X}_{c_i} \mathrm{X}_u^T)) \mathrm{X}_{c_i} \mathrm{X}_u - \sum_j \mathbf{w'}_{v_j} \sigma(\mathrm{X}_{v_j} \mathrm{X}_u^T) \mathrm{X}_{v_j} \mathrm{X}_u, \\
\frac{\partial \mathcal{L}}{\partial \mathrm{X}_{c_i}} &= \mathrm{X}_u \mathrm{X}_{c_i} (1 - \sigma(\mathrm{X}_{c_i} \mathrm{X}_u^T)) \mathrm{X}_u; \quad \frac{\partial \mathcal{L}}{\partial \mathrm{X}_{v_j}} = -\mathrm{X}_u \mathbf{w'}_{v_j} \sigma(\mathrm{X}_{v_j} \mathrm{X}_u^T) \mathrm{X}_u,
\end{aligned} \tag{4}$$

3 EVALUATION

3.1 Experimental Setup

3.1.1 **Data.** . We perform experiments to evaluate our *SANE* framework on the following two real-world network datasets:

- **Twitter Social Circle Data**: This dataset is collected from Twitter to contain the friend relationships (edges) between users (nodes). Each user is associated with 84 labels representing his/her profile information.
- **Facebook Social Circle Data**: Similar as the above Twitter data, this dataset contains the friendships (edges) between users

Table 1: Dataset Statistics

Dataset	# Nodes	# Edges	#Labels
Twitter Social Circle	4,039	88,234	84
Facebook Social Circle	81,306	1,768,149	1,603

(nodes) on Facebook and 1603 labels is leveraged to represent users' profile information.

3.1.2 **Baselines.** We evaluate the effectiveness of our *SANE* framework on two standard tasks: multi-label classification and node clustering. We consider the following state-of-the-art representation learning techniques as the baselines for both tasks:

- **MF** [7]: It assigns a D-dimensional latent vector for the generated matrix by decomposing a matrix into a product of sub-matrices.
- **Walkets** [9]: It is a network embedding method for learning multi-scale representations of nodes in a network by capturing network structure from multiple resolutions.
- **BPR** [10]: It is a learning algorithm for personalized ranking and can be applied in obtaining network embeddings using stochastic gradient descent and bootstrap sampling.
- **LINE** [12]: It is a network representation learning method which samples both one-hop and two-hop neighbors of each node to represent the network into a latent space.
- **DeepWalk** [8]/**node2vec** [4]: They are other two state-of-the-art network embedding methods. Although they use different techniques (DeepWalk–hierarchcal softmax,node2vec–negative sampling) in the random walk procedure, they generate similar results with the same random walk path input.

3.1.3 **Parameter Settings.** In this work, the developed *SANE* framework is implemented in TensorFlow and we optimize all morels with the Adam optimizer [5]. For *SANE* model, the embedding dimension d is set as 128 and the window size k_w is 10. Furthermore, we set the path length l and walks per node r as 80 and 10, respectively. For other two hyperparameters: similarity degree learning strength λ_1 and difficulty increasing rate λ_2, we set $\lambda_1 = \lambda_2 = 1.1$. We also investigate the influence of different hyper-parameter settings in the later subsection.

3.2 Multi-label Classification

We first present the evaluation results on the node classification task. We use the node label to determine the class of each node and apply *Marco-F1* and *Micro-F1* to evaluate the classification accuracy across different categories of node label [4]. In our experiments, we use external labels contained in the collected datasets as the ground truth, and train logistic regression classifier [3] on top of the learned embeddings from different methods to perform the prediction. We vary the training size from 5% to90% using a stratified split and consider the rest for testing. We repeat each prediction experiment 10 times and report the average performance.

Table 2 lists the evaluation results of all approaches on Facebook dataset. We can observe that our *SANE* framework consistently outperforms other baselines significantly in all cases. For example, given the training data percentage as 50%, *SANE* achieves relatively 16.4%, 7.6% improvement over LINE and DeepWalk/node2vec, respectively. We repeat the experiments on Twitter dataset and report the evaluation results in Table 3. From this table, we can observe that SANE achieves the best performance in terms of Macro-F1 and Micro-F1 with different size of training set, which further demonstrate the efficacy of our *SANE* which is capable of learning significantly better node embeddings than existing state-of-the-art methods in multi-label classification task. In summary, the advantage of *SANE* lies in its proper consideration of different degrees of node similarity which are usually unavailable in network data.

3.3 Node Clustering

In our experiments, we also investigate how the latent representations learned by embedding methods can help the node clustering task on the aforementioned two datasets. The embeddings learned from each method is considered as input to k-means algorithm and the clustering results is evaluated in terms of *Normalized Mutual Infomration (NMI)* [4]. All clustering experiments are conducted 10 times and the average performance is reported in Table 4. In this table, we can observe that *SANE* outperforms other competitive baselines in node clustering task on both Facebook and Twitter datasets. In summary, *SANE* could generate more appropriate embeddings for nodes in the network as compared to baselines, which suggest its ability to jointly capture the underlying structural and semantic relationships between nodes during the process of network representation learning.

3.4 Parameter Study

In this subsection, we analyze the influences of key hyperparameters on the performance of our *SANE* model. Due to space limit, we only present the evaluation results of multi-label classification task on Facebook dataset in Figure 1. From the evaluation results, we can observe that the proposed *SANE* is not strictly sensitive to different hyperparameter settings. We could observe that increasing the embedding dimension slightly improve the performance till d reaches 128, we attribute the improvement to the stronger representation ability with larger hidden state dimensionality. Additionally, we can notice that the performance becomes stable as long as the path length l is above 120. The number of walks per node r is positively correlated with the classification accuracy. The performance is improved slightly as window size k_w increases and then saturates when $k_w \geq 10$. We further observe that both similarity degree learning strength λ_1 and difficulty increasing rate λ_2 have low impact on the model performance.

4 CONCLUSION

This paper presents a similarity-ware embedding method for network embedding with a self-paced learning framework. The proposed approach can jointly incorporate the network structure and semantic relationship information between nodes into the network representation learning process. We evaluate the performance of our proposed approach on two real-world networks in both multi-label classification and node clustering tasks. Experimental results show that our SANE significantly outperforms competitive baselines by generating better embeddings for nodes in a network.

ACKNOWLEDGMENTS

This work is supported in part by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research

Table 2: Multi-label node classification results on Facebook data.

Metric	Model	Training Set %									
		5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	MF	0.6800	0.7224	0.7647	0.7792	0.7894	0.7951	0.7981	0.8021	0.8130	0.8148
	Walklets	0.6701	0.7058	0.7344	0.7491	0.7567	0.7629	0.7655	0.7681	0.7720	0.7735
	BPR	0.6566	0.6928	0.7298	0.7444	0.7602	0.7644	0.7689	0.7742	0.7833	0.7856
	LINE	0.6578	0.6973	0.7324	0.7475	0.7633	0.7682	0.7734	0.7820	0.7889	0.7912
	DeepWalk/Node2Vec	0.7298	0.7661	0.8008	0.8171	0.8261	0.8313	0.8359	0.8428	0.8501	0.8515
	SANE	**0.7478**	**0.8005**	**0.8435**	**0.8687**	**0.8830**	**0.8941**	**0.9005**	**0.9085**	**0.9159**	**0.9202**
Macro-F1	MF	0.3431	0.4155	0.5226	0.5633	0.5936	0.6167	0.6292	0.6425	0.6594	0.6158
	Walklets	0.3591	0.4280	0.4947	0.5253	0.5444	0.5610	0.5665	0.5746	0.5759	0.5481
	BPR	0.3302	0.3866	0.4678	0.4983	0.5323	0.5448	0.5542	0.5643	0.5745	0.5437
	LINE	0.3706	0.4462	0.5292	0.5693	0.6046	0.6239	0.6356	0.6495	0.6574	0.6141
	DeepWalk/Node2Vec	0.4527	0.5412	0.6224	0.6593	0.6829	0.7033	0.7181	0.7285	0.7276	0.6810
	SANE	**0.5079**	**0.6220**	**0.7125**	**0.7574**	**0.7826**	**0.8124**	**0.8258**	**0.8363**	**0.8345**	**0.7861**

Table 3: Multi-label node classification results on Twitter data.

Metric	Model	Training Set %									
		5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	MF	0.4889	0.5282	0.5529	0.5714	0.5875	0.5996	0.6095	0.6197	0.6261	0.6277
	Walklets	0.7527	0.8081	0.8552	0.8788	0.8919	0.9040	0.9116	0.9181	0.9251	0.9293
	BPR	0.4247	0.4539	0.4811	0.4989	0.5115	0.5226	0.5280	0.5372	0.5449	0.5465
	LINE	0.7192	0.7680	0.8123	0.8361	0.8516	0.8637	0.8736	0.8814	0.8885	0.8924
	DeepWalk/Node2Vec	0.7672	0.8335	0.8884	0.9123	0.9269	0.9361	0.9435	0.9484	0.9531	0.9561
	SANE	**0.8400**	**0.8948**	**0.9326**	**0.9473**	**0.9561**	**0.9617**	**0.9658**	**0.9685**	**0.9712**	**0.9733**
Macro-F1	MF	0.1716	0.1984	0.2289	0.2481	0.2654	0.2786	0.2892	0.2986	0.3057	0.3083
	Walklets	0.4641	0.5745	0.6772	0.7282	0.7610	0.7845	0.8003	0.8124	0.8148	0.7977
	BPR	0.1264	0.1463	0.1669	0.1768	0.1858	0.1918	0.1952	0.2000	0.2039	0.2056
	LINE	0.4043	0.4908	0.5804	0.6318	0.6669	0.6928	0.7121	0.7259	0.7337	0.7220
	DeepWalk/Node2Vec	0.4222	0.5415	0.6644	0.7287	0.7698	0.7968	0.8161	0.8288	0.8340	0.8159
	SANE	**0.5647**	**0.6830**	**0.7797**	**0.8223**	**0.8487**	**0.8665**	**0.8777**	**0.8850**	**0.8878**	**0.8634**

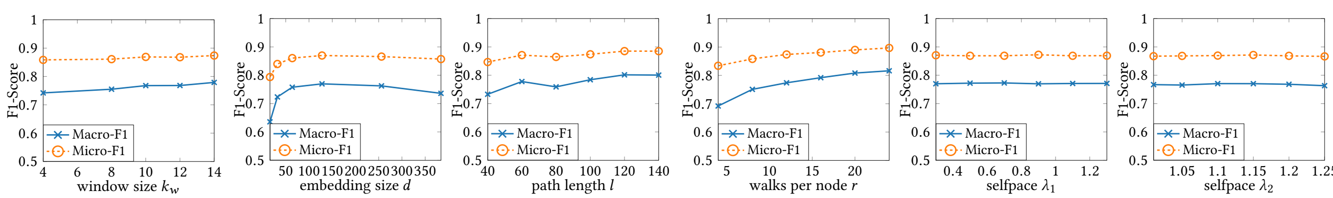

Figure 1: Hyper-parameter sensitivity evaluation of *SANE* in multi-label classification.

Table 4: Node clustering results on both Facebook and Twitter data in terms of Normalized Mutual Infomration (NMI).

Algorithm	Facebook Dataset	Twitter Dataset
MF	0.7871	0.7168
Walklets	0.7912	0.8570
BPR	0.7943	0.5628
LINE	0.7291	0.7500
DeepWalk/Node2Vec	0.7854	0.8621
SANE	0.7958	0.8892

Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*. ACM, 41–48.

[2] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)* (1977), 1–38.

[3] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, and etc. 2008. LIBLINEAR: A library for large linear classification. *JMLR* 9, Aug (2008), 1871–1874.

[4] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.

[5] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[6] M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *NIPS*. 1189–1197.

[7] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *NIPS*. 556–562.

[8] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. ACM, 701–710.

[9] Bryan Perozzi, Vivek Kulkarni, and etc. 2017. Don't Walk, Skip!: Online Learning of Multi-scale Network Embeddings. In *ASONAM*. ACM/IEEE, 258–265.

[10] Steffen Rendle, Christoph Freudenthaler, and etc. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. AUAI Press, 452–461.

[11] Yizhou Sun, Jiawei Han, and etc. 2012. When will it happen?: relationship prediction in heterogeneous information networks. In *WSDM*. ACM, 663–672.

[12] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and etc. 2015. Line: Large-scale information network embedding. In *WWW*. ACM, 1067–1077.