Ranking Network Embedding via Adversarial Learning

Quanyu Dai[1(✉)], Qiang Li[2], Liang Zhang[3], and Dan Wang[1]

[1] Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
{csqydai,csdwang}@comp.polyu.edu.hk
[2] Y-tech, Kwai, Beijing, China
leetsiang.cloud@gmail.com
[3] JD.com, Beijing, China
zhangliang16@jd.com

Abstract. Network Embedding is an effective and widely used method for extracting graph features automatically in recent years. To handle the widely existed large-scale networks, most of the existing scalable methods, e.g., DeepWalk, LINE and node2vec, resort to the negative sampling objective so as to alleviate the expensive computation. Though effective at large, this strategy can easily generate false, thus low-quality, negative samples due to the trivial noise generation process which is usually a simple variant of the unigram distribution. In this paper, we propose a Ranking Network Embedding (RNE) framework to leverage the ranking strategy to achieve scalability and quality simultaneously. RNE can explicitly encode node similarity ranking information into the embedding vectors, of which we provide two ranking strategies, vanilla and adversarial, respectively. The vanilla strategy modifies the uniform negative sampling method with a consideration of edge existance. The adversarial strategy unifies the triplet sampling phase and the learning phase of the model with the framework of Generative Adversarial Networks. Through adversarial training, the triplet sampling quality can be improved thanks to a softmax generator which constructs hard negatives for a given target. The effectiveness of our RNE framework is empirically evaluated on a variety of real-world networks with multiple network analysis tasks.

1 Introduction

Network embedding, i.e., learning low-dimensional representations for nodes in graph-structured data, can help encode meaningful semantic, relational and structural information of a graph into embedding vectors. Typically, such learning process is conducted in an unsupervised manner [1–3] due to the lack of labeled data, and thus the learned representations can be used to facilitate different kinds of downstream tasks such as network visualization, link prediction and node classification. In real-world applications, data entities with complicated relationships can be well organized with graphs. For example, paper citation networks characterize the information of innovation flow, social networks

Q. Yang et al. (Eds.): PAKDD 2019, LNAI 11441, pp. 27–39, 2019.
https://doi.org/10.1007/978-3-030-16142-2_3

entail complicated relationships among people, groups and organizations, and protein-protein interaction networks capture information between different proteins. Therefore, it is of great application interest to develop effective and scalable methods for unsupervised network embedding.

Network data are usually high-dimensional, very sparse and non-linear, which makes network embedding a challenging problem. Some classical methods, such as MDS [4], IsoMap [5] and LLE [6], can be used for network representation learning. However, they can neither effectively capture highly nonlinear structure of networks, nor scale to large networks. When handling large-scale networks, DeepWalk [1], LINE [2] and node2vec [7] are shown to be quite effective and efficient. These three methods preserve network structural properties in the embedding vectors through the negative sampling technique [8]. The negative sampling method is a simplified variant of negative contrastive estimation [9], which can help speed up the training process of the model. However, since the negative samples are constructed according to a unigram noise generation process, this strategy may generate false negative samples that violate pairwise relationships presented in the network structure. Here, we aim to answer two questions: (1) can we find some other ways for encoding pairwise relationships into node representations instead of the negative sampling approach? (2) how to sample better negative nodes for target-positive pairs (i.e., closely related node pairs) for training?

In this paper, we propose a Ranking Network Embedding (**RNE**) framework based on triplet ranking loss for preserving pairwise relationships of nodes in embedding vectors. Specifically, we firstly construct triplets based on network structure where each triplet consists of a target, a positive and a negative node. In the training process, the distance between embedding vectors of the target and positive node will be minimized while the distance between that of target and negative node will be maximized until they are separated by a predefined margin. Different from the negative sampling technique used in [1,2,7], the ranking strategy enforces a non-trivial margin between similar node pairs and dissimilar ones, thus explicitly encodes similarity ranking information among node pairs into the embedding vectors.

With the RNE framework, we propose two network embedding models by using a vanilla ranking strategy and an adversarial ranking technique respectively. In the vanilla RNE model, we utilize a simple negative node sampling method to construct triplets, which uniformly samples nodes from the node set without direct link to the target. This vanilla approach can perfectly avoid false negative samples while maintain the efficiency. Though works well to some extend, this vanilla strategy may also generate totally unrelated negative nodes for the target node, which will be of little help for the training. This phenomenon is even more common in very high-dimensional and sparse networks. To improve the vanilla RNE, we propose a generative adversarial model to unify the triplet sampling process and the learning process with the framework of Generative Adversarial Networks (GANs) [10], which leads to an adversarial RNE model. It leverages a generator for generating hard negative nodes with respect to a

given target to help construct high-quality triplets, and thus achieves better node similarity rankings in the embedding space. We empirically evaluate the proposed vanilla and adversarial RNE models through several network analysis tasks, including network visualization, link prediction and node classification, on benchmark datasets. Experimental results show that both models achieve competitive performance with state-of-the-art methods.

2 Related Work

Many scalable network embedding methods, such as DeepWalk [1], LINE [2] and node2vec [7], have been proposed to learn node representations to facilitate downstream tasks. They model node conditional probability with softmax function over the whole network, which is computationally expensive. Further, the negative sampling approach [8] is usually leveraged to replace the log likelihood objective, and thus enabling a scalability to large networks. However, it can generate some negative samples violating pairwise relationships reflected by network structure because of the simple unigram noise generation process. To handle this issue, we propose to use the triplet ranking loss to learn embedding vectors and leverage an adversarial sampling method to sample negative nodes. We noticed that the triplet ranking loss is also employed by [11] in learning embeddings, but for networks with node attributes.

Recently, some methods are proposed to learn node representations through adversarial training [12,13]. In ANE [12], a prior distribution is imposed on node representations through adversarial training to achieve robustness. In [13], the authors proposed to unify the generative models and discriminative models of network embedding into the framework of GANs to help learn a stronger generator. Different from these two methods, our method aim to learn a stronger discriminator to obtain node representations.

Some knowledge graph embedding methods are also related [14–16]. TransE [14] is a translation-based knowledge graph embedding model, which learns embeddings for both data entities and relations with triplet ranking loss. KBGAN [16] is an adversarial learning framework for knowledge graph embedding. Our method is by part inspired by these works. However, this line of research has notable differences with our work. Firstly, knowledge graph is fundamentally different from the networks we study. The assumption, that two connected nodes should be similar and close in embedding space, of network embedding methods does not hold in knowledge graph. Secondly, knowledge graph embedding learns representations for both data entities (nodes) and relations (edges) simultaneously, while network embedding is designed to learn node representations only.

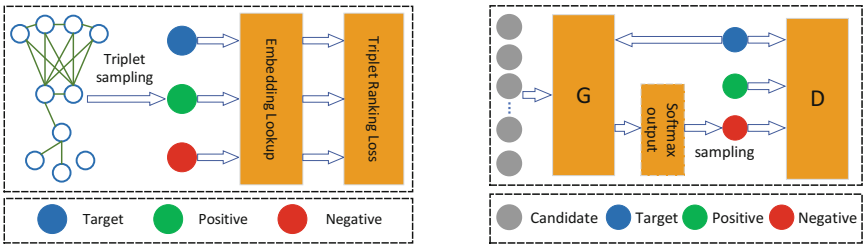
3 RNE: Ranking Network Embedding

3.1 Framework

The framework of Ranking Network Embedding method is shown in Fig. 1(a). It consists of two phases, i.e., the triplet construction phase and the learning

phase. Firstly, we leverage some sampling methods to construct triplets based on network structure, which can help specify the similarity ranking of some pairwise relationships. Then, in the learning process, triplet ranking loss is minimized by directly updating embeddings to pull similar nodes closer in the embedding space, while pushing dissimilar nodes apart.

To help better understand our model, we first introduce some notations and describe the research problem. A network is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with a set of nodes $\mathcal{V}$ representing data entities and a set of edges $\mathcal{E}$ each representing the relationship between two nodes. We mainly consider undirected graph in this paper. Given a graph $\mathcal{G}$, we aim to learn low-dimensional representations $\boldsymbol{u_i} \in R^d$ for each node $v_i \in \mathcal{V}$, which can capture network structural properties. We denote U as embedding matrix with $\boldsymbol{u_i}$ as its ith row.



(a) RNE. (b) Adversarial RNE.

Fig. 1. Model architecture.

3.2 Vanilla Ranking Network Embedding

The vanilla RNE model is a simple instantiation of the proposed RNE framework with uniform negative sampling method. Some detailed descriptions of its triplet sampling method and loss function are provided below.

Vanilla Triplet Sampling. Triplet sampling method plays an important role in learning good embedding vectors for downstream learning tasks. The constructed triplets directly specify pairwise relationships from network structure which will be regarded as ground-truth in learning process to be encoded into embedding vectors. We only explicitly consider first-order proximity when constructing positive pairs. The triplet set $\mathcal{T}$ is defined as follows:

$$\mathcal{T} = \{(v_t, v_p, v_n) | (v_t, v_p) \in \mathcal{E}, (v_t, v_n) \notin \mathcal{E}\}, \tag{1}$$

where (v_t, v_p, v_n) is a triplet with v_t, v_p and v_n as the target, positive and negative node, respectively. Since network is usually very sparse, for each positive pair, there can be a large number of negative nodes. To improve model efficiency, we only uniformly sample K negative nodes from the negative space for each positive pair.

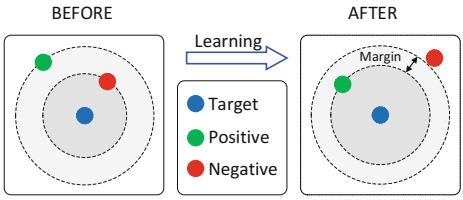


Fig. 2. The triplet ranking loss minimizes the distance between a target node and a positive node while maximizing that of the target and a negative node until they are separated by at least a margin distance. The pairwise relationships can be well preserved in embedding vectors after the learning process.

Triplet Ranking Loss. For vanilla RNE model, we seek to minimize the following loss function:

$$\mathcal{L} = \sum_{(v_t, v_p, v_n) \in \mathcal{T}} [m + D(v_t, v_p; \theta_D) - D(v_t, v_n; \theta_D)]_+, \tag{2}$$

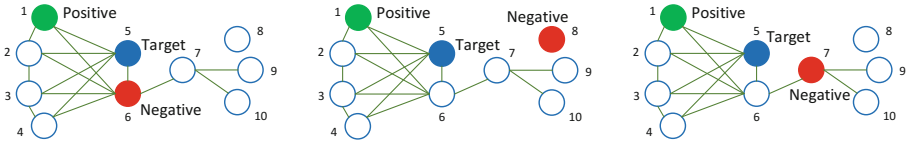
where $[x]_+$ denotes the positive part of x, $D(v_1, v_2; \theta_D)$ is a distance function of two nodes, θ_D is the union of all node embeddings, and $m > 0$ is a margin hyper-parameter separating the positive pair and the corresponding negative one. We use the squared L2 distances in the embedding space, i.e., $D(v_1, v_2; \theta_D) = \|\boldsymbol{u_1} - \boldsymbol{u_2}\|^2$. The triplet ranking loss explicitly encodes similarity ranking among node pairs into the embedding vectors, and the visualization explanation can be found in Fig. 2 [17].

3.3 Adversarial Ranking Network Embedding

For the vanilla RNE model, we only use uniform negative sampling method for constructing triplets. It can easily generate totally unrelated negative nodes for the target node due to the sparsity and high-dimensionality of the network, which will be of little help for the training process. To help alleviate this problem, we propose an Adversarial Ranking Network Embedding model, which unifies the triplet sampling phase and the learning phase of the RNE method with the framework of GANs. The model architecture is presented in Fig. 1(b). It consists of a generator G and a discriminator D (we abuse the notation and directly use the distance function D to represent the discriminator). In the learning process, the discriminator tries to pull similar nodes closer in the embedding space, while pushing dissimilar nodes apart. The generator aims to generate difficult negative nodes for a given target from a set of negative candidates by optimizing its own parameters.

Discriminator. The discriminator is aimed at optimizing the following triplet ranking loss function similar to the vanilla RNE model:

$$\mathcal{L}_D = \sum_{(v_t, v_p) \in \mathcal{P}} \mathbb{E}_{v_n \sim G(\cdot | v_t; \theta_G)} [m + D(v_t, v_p; \theta_D) - D(v_t, v_n; \theta_D)]_+, \tag{3}$$



(a) Negative sampling. (b) Uniform sampling. (c) Adversarial sampling.

Fig. 3. For the negative sampling approach, each node is sampled according to its unigram distribution (regard each node as a word) raised to the 3/4 power, which can violate pairwise relationships reflected by network structure. For example, node 6 is very likely to be sampled as negative node for target-positive pair (5, 1), even though node 5 and 6 have strong relationship. For our triplet sampling method, such problem can be well avoided. However, simple uniform sampling method can easily generate totally unrelated nodes (node 8 in the example graph), which can be improved with adversarial sampling method.

where $\mathcal{P} = \{(v_1, v_2), (v_2, v_1) | (v_1, v_2) \in \mathcal{E}\}$ is the positive pair set in graph $\mathcal{G}$, and $G(\cdot|v_t; \theta_G)$ is the generator. Only first-order proximity is directly considered, and each edge $(v_i, v_j) \in \mathcal{E}$ corresponds to two positive pairs (v_i, v_j) and (v_j, v_i). Particularly, a softmax generator is employed to construct high-quality triplets instead of simple uniform sampling method. More detailed illustrations of this sampling method will be introduced below. Note that Eq. (3) can be directly optimized with gradient descent technique.

Generator. Softmax function is widely used in network embedding literature [1, 18] to model node conditional probability. In this paper, we also employ softmax function as the generator to sample negative nodes given a target, but it is defined over the negative node space with respect to the given positive pair according to network structure. Specifically, the generator $G(v_n|v_t; \theta_G)$ is defined as a softmax function over a set of negative candidates:

$$G(v_n|v_t; \theta_G) = \frac{\exp(\boldsymbol{u_n}^T \boldsymbol{u_t})}{\sum_{v_{n_i} \in Neg(v_t, v_p)} \exp(\boldsymbol{u_{n_i}}^T \boldsymbol{u_t})}, \tag{4}$$

where $Neg(v_t, v_p) = \{v_{n_1}, v_{n_2}, \cdots, v_{n_{N_c}}\}$ is a set of negative candidates with size as N_c. In implementation, $Neg(v_t, v_p)$ is a subsample of the original negative space of the positive pair to reduce the computation complexity, which is actually a common practice in network embedding literature [1,2]. For each positive pair, N_c negative nodes will be first uniformly randomly sampled from the negative space, and used as input for the generator. Then, a hard negative node will be sampled from $Neg(v_t, v_p)$ according to the probability distribution $G(v_n|v_t; \theta_G)$. Besides, in the training process, K hard negatives will be sampled for each positive pair.

The loss function of the generator is defined as follows:

$$\mathcal{L}_G = \sum_{(v_t, v_p) \in \mathcal{P}} \mathbb{E}_{v_n \sim G(\cdot|v_t; \theta_G)}[D(v_t, v_n; \theta_D)]. \tag{5}$$

It can encourage the softmax generator to generate useful negative nodes for a given positive pair instead of totally unrelated ones. The sampling process of hard negatives is discrete, which hinders the objective from directly being optimized by gradient descent method as that of the discriminator. According to [19,20], this loss can be optimized with the following policy gradient:

$$\begin{aligned}\nabla_{\theta_G}\mathcal{L}_G &= \nabla_{\theta_G}\sum_{(v_t,v_p)}\mathbb{E}_{v_n\sim G(\cdot|v_t;\theta_G)}[D(v_t,v_n;\theta_D)] \\ &= \sum_{(v_t,v_p)}\mathbb{E}_{v_n\sim G(\cdot|v_t;\theta_G)}[D(v_t,v_n;\theta_D)\nabla_{\theta_G}\log G(v_n|v_t;\theta_G)].\end{aligned} \tag{6}$$

The gradient of $\mathcal{L}_G$ is an expected summation of the gradient $\nabla_{\theta_G}\log G(v_n|v_t;\theta_G)$ weighted by the distance of node pair (v_t, v_n). When training the generator, the parameters will be shifted to involve high-quality negatives with high probability from softmax generator, i.e., node pairs (v_t, v_n) with small distance from discriminator will be encouraged to be generated. In practice, the expectation can be approximated with sampling in the negative space. Besides, the REINFORCE algorithm suffers from the notorious high variance, which can be alleviated by subtracting a baseline function from the reward term of the objective, i.e., adding a baseline function to the reward term in the loss [21]. Specifically, we replace $D(v_t, v_n;\theta_D)$ in the loss by its advantage function as follows:

$$D(v_t,v_n;\theta_D) + \sum_{(v_t,v_p)}\mathbb{E}_{v_n\sim G(\cdot|v_t;\theta_G)}[D(v_t,v_n;\theta_D)], \tag{7}$$

where $\sum_{(v_t,v_p)}\mathbb{E}_{v_n\sim G(\cdot|v_t;\theta_G)}[D(v_t,v_n;\theta_D)]$ is the average reward of the whole training set, and acts as the baseline function in policy gradient.

A comparison of the sampling methods is presented in Fig. 3 with toy examples. Our proposed adversarial sampling method can help select difficult negative nodes with respect to given target. With high-quality triplets, the tricky pairwise relationship rankings can be encoded into node representations through the training of the discriminator as illustrated in Fig. 2. Note that false negative nodes can still be generated by the generator due to the incompleteness and non-linearity of the real-world networks, but in a very low probability since the subsampling trick is employed for generating negative candidates among a very large negative space. So, the embedding vectors can be improved in general. This is also validated by our experiments.

Algorithm 1 presents the pseudocode of the adversarial RNE model, which employs a joint training procedure. The overall time complexity of the algorithm is linear to the number of edges, i.e., $\mathcal{O}(dKN_c|\mathcal{E}|)$ (d, K and N_c are some constants independent of the network size), which enables it scale to large networks.

Algorithm 1. The adversarial RNE algorithm

Input : $\mathcal{G}(\mathcal{V}, \mathcal{E})$, Dimension d, Margin m, Negative size K, Candidate size N_c
Output: The parameters of Discriminator θ_D

Initialize the Generator $G(v_n|v_t;\theta_G)$ and Discriminator $D(v_1, v_2;\theta_D)$ with pretrained embedding vectors;
while *not converge* **do**
  Sample a batch of positive pairs $\mathcal{B}$ from positive set $\mathcal{P}$;
  $\mathcal{T} = \{\}$; $\mathcal{N} = \{\}$;
  // Adversarial negative sampling with softmax generator
  for *each* $(v_t, v_p) \in \mathcal{B}$ **do**
    repeat
      Sample N_c negative candidates $Neg(v_t, v_p)$ uniformly from the negative space of (v_t, v_p);
      Sample a hard negative v_n from $Neg(v_t, v_p)$ according to $G(v_n|v_t, \theta_G)$;
      $\mathcal{T} = \mathcal{T} \cup \{v_n\}$; $\mathcal{N} = \mathcal{N} \cup \{Neg(v_t, v_p)\}$;
    until K *times*;
  end
  // Parameters updating
  update θ_D according to Eq. (3) with $\mathcal{T}$ as training batch;
  update θ_G according to Eq. (5) and (4) with $\mathcal{T}$ and $\mathcal{N}$ as training batch;
end

4 Experiments

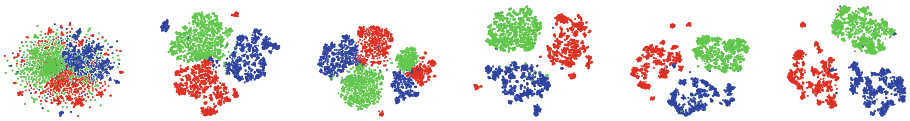
4.1 Experiment Setup

Datasets. We conduct experiments on benchmark datasets from various real-world applications. Table 1 shows some statistics of them. Note that we regard all paper citation networks as undirected networks, and do some preprocessing on the original datasets by deleting self-loops and nodes with zero degree.

Table 1. Statistics of benchmark datasets from real-world applications

Name	Citeseer [22]	Cit-DBLP [23]	PubMed [24]	CA-GrQc [25]	CA-HepTh [25]	Wiki [26]	USA-AIR [27]
$\|V\|$	3,264	5,318	19,717	5,242	9,877	2,363	1,190
$\|E\|$	4,551	28,065	44,335	14,484	25,973	11,596	13,599
Avg. degree	1.39	5.28	2.25	2.76	2.63	4.91	11.43
#Labels	6	3	3	-	-	17	4

Baseline Models. We only consider scalable baselines in this paper. Some matrix factorization based methods such as M-NMF [3,28] are excluded from the baselines due to the $O(|V|^2)$ time complexity. The descriptions of the baseline models are as follows: Graph Factorization (GF) [29] directly factorizes the adjacency matrix to obtain the embeddings. DeepWalk (DW) [1] regards node sequence obtained from truncated random walk as word sequence, and then uses skip-gram model to learn node representations. LINE [2] preserves proximities through modeling node co-occurrence probability and node conditional probability. node2vec (n2v) [7] develops a biased random walk procedure to explore neighborhood of a node, which can strike a balance between local and global properties. We denote the vanilla RNE model as V-RNE, and the adversarial RNE model as A-RNE in the rest of the paper.



(a) GF. (b) DW. (c) LINE. (d) n2c. (e) V-RNE. (f) A-RNE.

Fig. 4. Visualization of Cit-DBLP network.

Parameter Settings. The window size, walk length and the number of walks per node of both DeepWalk and node2vec are set to 10, 80 and 10, respectively. We use node2vec in an unsupervised manner by setting both in-out and return hyperparameters to 1.0 for fair comparison. For LINE, we follow the original paper [2] to set the parameters. For our method, the parameter settings are the margin $m = 2.5$, the negative size per edge $K = 5$, and the negative candidate size $N_c = 5$. The learning rate of V-RNE is set to 0.01, while A-RNE to 0.0001. L2-normalization is conducted on node embeddings for both the V-RNE and A-RNE model after each training epoch. Besides, the dimension of embedding vectors are set to 128 for all methods.

4.2 Network Visualization

We leverage a commonly used toolkit *t-SNE* [30] to visualize node embeddings of Cit-DBLP generated by different models. Cit-DBLP is a citation network constructed from the DBLP datsest [23], which consists of papers from publication venues including Information Sciences, ACM Transactions on Graphics and Human-Computer Interaction. These papers are naturally classified into three categories according to their publication venues, and represented with different colored nodes in the visualization.

Experimental Results. Figure 4 displays the visualization results. Papers from different publication venues are mixed together terribly for GF as shown in Fig. 4(a). In the center part of both DeepWalk and LINE, papers from different categories are mixed with each other. Visualizations from node2vec, V-RNE and A-RNE are much better as three clusters are formed with quite clear margin. Compared with V-RNE, A-RNE model has better visualization result, since the margin between different clusters are larger. The reason is that adversarial sampling method aims to generate hard negative nodes, i.e., negative nodes near the boundary, which directly contributes to producing more clear margin between different clusters. On the whole, this experiment demonstrates that ranking network embedding method can help capture intrinsic structure of original network in embedding vectors.

4.3 Link Prediction

We conduct link prediction on three benchmark datasets. For each network, we randomly and uniformly sample 20% and 50% of the edges as test labels and use

the remaining network as input to the models, i.e., training ratio as 80% and 50%. When sampling edges, we ensure the degree of each node is greater than or equal to 1 to avoid meaningless embedding vectors. The prediction performance is measured by AUC score. To calculate AUC score, we first obtain the edge features from the learned node embeddings through Hadamard product of embeddings of two endpoints as many other works [7], and then train a L2-SVM classifier with under-sampling to get prediction results.

Table 2. AUC score for link prediction

Training ratio	80%			50%		
Dataset	Wiki	CA-GrQc	CA-HepTh	Wiki	CA-GrQc	CA-HepTh
GF	0.583 ± 0.008	0.593 ± 0.003	0.554 ± 0.001	0.566 ± 0.002	0.572 ± 0.003	0.531 ± 0.001
DeepWalk	0.656 ± 0.001	0.694 ± 0.001	0.683 ± 0.001	0.639 ± 0.001	0.657 ± 0.002	0.630 ± 0.001
LINE	0.649 ± 0.007	0.638 ± 0.005	0.630 ± 0.001	0.627 ± 0.014	0.600 ± 0.003	0.561 ± 0.002
node2vec	0.634 ± 0.016	0.690 ± 0.007	0.668 ± 0.003	0.621 ± 0.010	0.667 ± 0.010	0.624 ± 0.007
V-RNE	0.647 ± 0.008	0.691 ± 0.005	0.657 ± 0.005	0.627 ± 0.007	0.655 ± 0.004	0.606 ± 0.004
A-RNE	$\mathbf{0.670 \pm 0.005}$	$\mathbf{0.708 \pm 0.004}$	$\mathbf{0.688 \pm 0.004}$	$\mathbf{0.655 \pm 0.006}$	$\mathbf{0.673 \pm 0.004}$	$\mathbf{0.639 \pm 0.004}$

Experimental Results. The link prediction results are the average of 10 different runs, which are shown in Table 2. The AUC scores of A-RNE model consistently outperform those of the V-RNE model. It validates that A-RNE can help achieve better node similarity rankings in embedding space, since link prediction task can be considered as similarity ranking among node pairs. The performance of the proposed RNE method is competitive with the baselines, which shows that using ranking strategy for learning node representations is a good practice. In particular, the AUC scores of A-RNE model are superior to all the baselines in all test datasets when the training ratios are 80% and 50%.

Table 3. Accuracy (%) of multi-class classification on USA-AIR and PubMed

Dataset	USA-AIR					Pubmed				
Ratio	10%	30%	50%	70%	90%	10%	30%	50%	70%	90%
GF	41.10	42.21	42.27	41.12	41.60	35.63	36.69	37.56	37.74	38.08
DeepWalk	43.43	51.79	53.41	55.74	56.05	69.43	71.33	71.74	71.82	72.37
LINE	48.80	53.95	56.35	56.72	58.91	67.23	69.20	69.84	69.97	70.48
node2vec	42.76	47.07	48.62	49.86	50.76	79.66	80.89	81.09	81.07	81.27
V-RNE	55.20	58.96	60.05	61.29	61.09	77.56	79.08	79.39	79.46	79.73
A-RNE	**56.94**	**61.96**	**62.79**	**65.71**	**64.12**	**80.48**	**81.20**	**81.58**	**81.56**	**81.64**

4.4 Node Classification

Node classification can be conducted to dig out missing information. In this section, we carry out experiments on the air-traffic network USA-AIR and paper

citation network PubMed. The learned embedding vectors are used as feature input for the classification model. We randomly sample a portion of nodes as training data ranging from 10% to 90%, and the rest for testing. For both datasets, multi-class classification is conducted, and accuracy score is used for performance comparison. All experiments are conducted with support vector classifier in Liblinear package[1] [31] with default settings.

Experimental Results. The experimental results are presented in Table 3. Both V-RNE and A-RNE perform competitively with baseline models for these two datasets while varying the train-test split from 10% to 90%. It shows the effectiveness of the proposed Ranking Network Embedding models for learning discriminative embedding vectors for classification. Specifically, both V-RNE and A-RNE achieve better performance in USA-AIR, and A-RNE obtains the best results in these two datasets across all training ratios. In particular, A-RNE gives us 13.32% gain on average over the best baseline, i.e., LINE on USA-AIR. Besides, A-RNE consistently achieves more excellent performance than V-RNE as shown in the tables, which demonstrates that adversarial sampling method contributes to learning more discriminative node representations.

5 Conclusion

This paper presented a novel scalable Ranking Network Embedding method, which can explicitly encode node similarity ranking information into the embedding vectors. Firstly, a vanilla RNE model was proposed with uniform negative sampling method. Then, we improved the vanilla RNE model by unifying the triplet sampling phase and the learning phase with the framework of GANs which leads to an adversarial RNE model. The adversarial RNE model utilizes a softmax generator to generate hard negatives for a given a target, which can help strengthen the discriminator. Empirical evaluations prove the effectiveness of the proposed method on several real-world networks with a variety of network analysis tasks.

References

1. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: KDD, pp. 701–710 (2014)
2. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: WWW, pp. 1067–1077 (2015)
3. Cao, S., Lu, W., Xu, Q.: Grarep: learning graph representations with global structural information. In: CIKM, pp. 891–900 (2015)
4. Cox, T.F., Cox, M.A. (eds.): Multidimensional Scaling. CRC Press, Boca Raton (2000)
5. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**, 2319–2323 (2000)

[1] https://www.csie.ntu.edu.tw/~cjlin/liblinear/.

6. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**, 2323–2326 (2000)
7. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD, pp. 855–864 (2016)
8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
9. Gutmann, M., Hyvärinen, A.: Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. J. Mach. Learn. Res. **13**, 307–361 (2012)
10. Goodfellow, I.J., et al.: Generative adversarial nets. In: NIPS, pp. 2672–2680 (2014)
11. Duran, A.G., Niepert, M.: Learning graph representations with embedding propagation. In: NIPS, pp. 5125–5136 (2017)
12. Dai, Q., Li, Q., Tang, J., Wang, D.: Adversarial network embedding. In: AAAI (2018)
13. Wang, H., et al.: Graph representation learning with generative adversarial nets. In: AAAI, Graphgan (2018)
14. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS (2013)
15. Wang, P., Li, S., Pan, R.: Incorporating GAN for negative sampling in knowledge representation learning. In: AAAI (2018)
16. Cai, L., Wang, W.Y.: KBGAN: adversarial learning for knowledge graph embeddings. CoRR, abs/1711.04071 (2017)
17. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: a unified embedding for face recognition and clustering. In: CVPR, pp. 815–823 (2015)
18. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: ICML, pp. 40–48 (2016)
19. Schulman, J., Heess, N., Weber, T., Abbeel, P.: Gradient estimation using stochastic computation graphs. In: NIPS, pp. 3528–3536 (2015)
20. Yu, L., Zhang, W., Wang, J., Yu, Y.: SeqGAN: sequence generative adversarial nets with policy gradient. In: AAAI, pp. 2852–2858 (2017)
21. Sutton, R.S., Mcallester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: NIPS, pp. 1057–1063. MIT Press (2000)
22. McCallum, A., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Inf. Retr. **3**(2), 127–163 (2000)
23. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: ArnetMiner: extraction and mining of academic social networks. In: KDD, pp. 990–998 (2008)
24. Nandanwar, S., Narasimha Murty, M.: Structural neighborhood based classification of nodes in a network. In: KDD, pp. 1085–1094 (2016)
25. Leskovec, J., Kleinberg, J.M., Faloutsos, C.: Graph evolution: densification and shrinking diameters. TKDD **1**(1), 2 (2007)
26. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93–106 (2008)
27. Ribeiro, L.F.R., Saverese, P.H.P., Figueiredo, D.R.: struc2vec: Learning node representations from structural identity. In: KDD, pp. 385–394 (2017)
28. Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., Yang, S.: Community preserving network embedding. In: AAAI, pp. 203–209 (2017)

29. Ahmed, A., Shervashidze, N., Narayanamurthy, S.M., Josifovski, V., Smola, A.J.: Distributed large-scale natural graph factorization. In: WWW, pp. 37–48 (2013)
30. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. JMLR **9**, 2579–2605 (2008)
31. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J.: LIBLINEAR: a library for large linear classification. JMLR **9**, 1871–1874 (2008)