Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Deep Attributed Network Embedding

Hongchang Gao, Heng Huang
Department of Electrical and Computer Engineering
University of Pittsburgh, USA
hongchanggao@gmail.com, heng.huang@pitt.edu

Abstract

Network embedding has attracted a surge of attention in recent years. It is to learn the low-dimensional representation for nodes in a network, which benefits downstream tasks such as node classification and link prediction. Most of the existing approaches learn node representations only based on the topological structure, yet nodes are often associated with rich attributes in many real-world applications. Thus, it is important and necessary to learn node representations based on both the topological structure and node attributes. In this paper, we propose a novel deep attributed network embedding approach, which can capture the high non-linearity and preserve various proximities in both topological structure and node attributes. At the same time, a novel strategy is proposed to guarantee the learned node representation can encode the consistent and complementary information from the topological structure and node attributes. Extensive experiments on benchmark datasets have verified the effectiveness of our proposed approach.

1 Introduction

Networks are ubiquitous in the real world, such as social networks, academic citation networks, and communication networks. Among various networks, the attributed network has attracted much attention in recent years. Unlike the plain network where only the topological structure is available, nodes of attributed networks possess rich attributed information. These informative attributes can benefit network analysis. For example, in an academic citation network, the citation among different articles compose a network where each node is an article and each node has substantial text information about the article's topic. Another example is the social network where users connect with others and post their profiles as the attribute. Furthermore, the social science [Marsden and Friedkin, 1993; McPherson *et al.*, 2001] has shown that attributes of nodes can reflect and affect their community structures [Huang *et al.*, 2017b; Yang *et al.*, 2010]. Thus, it is necessary and important to study the attributed network.

Network embedding as a fundamental tool to analyze networks has attracted a surge of attention in data mining and machine learning community recently. It is to learn a low-dimensional representation for each node of a network while preserving the proximity. Then, the downstream tasks, such as node classification, link prediction, and network visualization, can benefit from the learned low-dimensional representation. In recent years, various network embedding methods have been proposed, such as DeepWalk [Perozzi *et al.*, 2014], Node2Vec [Grover and Leskovec, 2016], and LINE [Tang *et al.*, 2015]. However, most of the existing approaches mainly focus on the plain network, ignoring useful attributes of nodes. For example, in the social network such as Facebook or Twitter, each user connects with others, constituting a network. Most of existing methods only focus on the connection when learning node representations. But the attribute of each node can also provide useful information. A good example is the user profile. A young user may have much more similarity with another young guy, rather than an old user. Thus, it is important to incorporate node attributes when learning node representations.

Moreover, the topological structure and attributes of networks are highly non-linear [Wang *et al.*, 2016]. Thus, it is important to capture the highly non-linear property to discover the underlying pattern. Then, the proximity can be preserved better in the learned node representations. However, most of existing methods, such as [Huang *et al.*, 2017a; Yang *et al.*, 2015], only employ the shallow model, failing to capture the highly non-linear property. Furthermore, how to capture this highly non-linear property is difficult due to the complicated topological structure and attributes. Therefore, it is challenging to capture the highly non-linear property for attributed network embedding.

To address the aforementioned problems, we propose a novel deep attributed network embedding (DANE) approach for attributed networks. In detail, a deep model is proposed to capture the underlying high non-linearity in both topological structure and attributes. Meanwhile, the proposed model can enforce the learned node representations to preserve the first-order and high-order proximity in original networks. Moreover, to learn the consistent and complementary representation from the topological structure and attributes of a network, we propose a novel strategy to combine these two kinds of information. Furthermore, to obtain a robust node representa-

tion, an efficient most negative sampling strategy is proposed to make the loss function robust. At last, extensive experiments have been conducted to verify the effectiveness of our proposed approach.

2 Related Works

2.1 Plain Network Embedding

Network embedding can be traced back to the graph embedding problem, such as Laplacian Eigenmaps [Belkin and Niyogi, 2001], LPP [He and Niyogi, 2003]. These methods are to learn data embedding while preserving the local manifold structure. However, these methods are not applicable for large-scale network embedding, since they involve the time-consuming eigendecomposition operation whose time complexity is $O(n^3)$ where n is the number of nodes. Recently, with the development of large-scale networks, a variety of network embedding methods [Perozzi *et al.*, 2014; Grover and Leskovec, 2016; Cao *et al.*, 2015; Tang *et al.*, 2015; Gao and Huang, 2018] have been proposed. For example, DeepWalk [Perozzi *et al.*, 2014] adopts random walk and Skip-Gram to learn node representations, based on the observation that the distribution of nodes in the random walk is similar to that of words in natural language. LINE [Tang *et al.*, 2015] proposes to preserve the first- and second-order proximity when learning node representations. And GraRep [Cao *et al.*, 2015] is proposed to preserve high order proximity. Furthermore, Node2Vec [Grover and Leskovec, 2016] is proposed by designing a biased random walk on the flexible node's neighborhood. However, all of these methods only utilize the topological structure, ignoring the useful attribute of nodes.

2.2 Attributed Network Embedding

Attributed network embedding has attracted much attention in recent years. A wide variety of models have been proposed for attributed networks. For example, [Yang *et al.*, 2015] proposes an inductive matrix factorization method to combine the topological structure and attributes of networks. However, it is a linear model essentially, which is not enough for the complicated attributed network. [Huang *et al.*, 2017a; 2017b] employ the graph Laplacian technique to learn the joint embedding from the topological structure and attributes. [Kipf and Welling, 2016a] proposes a graph convolutional neural network model for attributed networks. But this model is a semi-supervised approach, failing to deal with the unsupervised case. [Pan *et al.*, 2016] proposes to combine DeepWalk with a neural network for network representation. Nonetheless, the DeepWalk part is still a shallow model. Recently, two unsupervised deep attributed network embedding approaches [Kipf and Welling, 2016b; Hamilton *et al.*, 2017] have been proposed. However, they can only explore the topological structure implicitly. Thus, it is necessary to explore the deep attributed network embedding method in a more effective way.

3 Deep Attributed Network Embedding

In this section, we first give the formal definition of attributed network embedding and then develop our novel deep attributed network embedding approach.

3.1 Problem Definition

Let $G = \{E, Z\}$ denote an attributed network with n nodes, where $E = [E_{ij}] \in \Re^{n\times n}$ is the adjacency matrix and $Z = [Z_{ij}] \in \Re^{n\times m}$ is the attribute matrix. In detail, if there exists an edge between the i-th node and the j-th node, $E_{ij} > 0$. Otherwise, $E_{ij} = 0$. $Z_{i\cdot} \in \Re^m$ is the i-th row of Z, which denotes the attribute of the i-th node.

Before giving the problem definition, we first introduce the definition of different proximities, which is important for our approach.

Definition 1. *(First-Order Proximity [Tang* et al.*, 2015]) Given a network $G = \{E, Z\}$. The first-order proximity of two nodes i and j is determined by E_{ij}. Specifically, a larger E_{ij} denotes a larger proximity between the i-th node and the j-th one.*

The first-order proximity indicates that if there exists a link between two nodes, they are similar. Otherwise, they are dissimilar. Thus, it can be viewed as a local proximity.

Definition 2. *(High-Order Proximity [Cao* et al.*, 2015]) Given a network $G = \{E, Z\}$, the high-order proximity of two nodes i and j is determined by the similarity of $M_{i\cdot}$ and $M_{j\cdot}$, where $M = \hat{E} + \hat{E}^2 + \cdots + \hat{E}^t$ is the high-order proximity matrix, $\hat{E}$ is the 1-step probability transition matrix which is obtained from the row-wise normalization of the adjacency matrix E.*

The high-order proximity actually indicates the neighborhood similarity. Specifically, if two nodes share similar neighbors, they are similar. Otherwise, they are not similar. Here, the high-order proximity can be viewed as a global proximity.

Definition 3. *(Semantic Proximity) Given a network $G = \{E, Z\}$, the semantic proximity of two nodes i and j is determined by the similarity of $Z_{i\cdot}$ and $Z_{j\cdot}$.*

The semantic proximity denotes that if two nodes have similar attributes, they are similar. Otherwise, they are dissimilar.

Attributed network embedding is to learn the low-dimensional representation of each node based on the adjacency matrix E and the attribute matrix Z, such that the learned representation can preserve the proximity existing in the topological structure and node attributes. Formally, we aim at learning a map $f : \{E, Z\} \to H$ where $H \in \Re^{n\times d}$ is the node representation, such that H can preserve *first-order proximity*, *high-order proximity* and *semantic proximity*. Then, downstream tasks, such as node classification and link prediction, can be performed on the learned H.

3.2 Deep Attributed Network Embedding

Essentially, the attributed network embedding faces three great challenges to obtain a good embedding result. They are:

• Highly non-linear structure: The underlying structure of the topological structure and attributes are highly non-linear, thus it is difficult to capture this non-linearity.

• Proximity preservation: The proximity in an attributed network depends both on the topological structure and the

attribute, thus how to discover and preserve the proximity is a tough problem.

• Consistent and complementary information in the topological structure and attributes: These two kinds of information provide different views for each node, thus it is important to make the learned node representations preserve the consistent and complementary information in these two modalities.

To address these three challenges, we develop a novel deep attributed network embedding (DANE) approach. The architecture is shown in Figure 1.

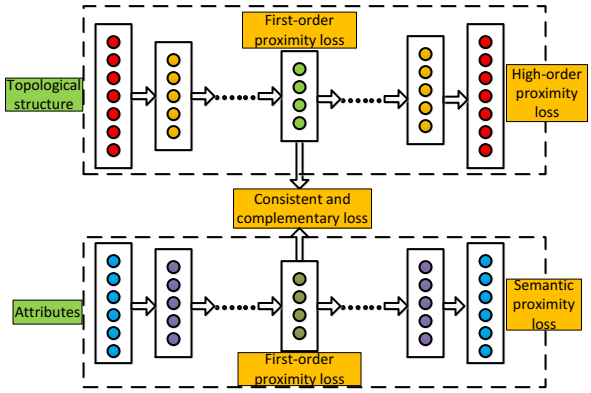


Figure 1: The architecture of our proposed DANE. The input of the network in the first row is the topological structure, and that in the second row is the node attribute.

Overall, there are two branches where the first branch is composed of a multi-layer non-linear function, which can capture the highly non-linear network structure, to map the input M to the low-dimensional space. Similarly, the second branch is to map the input Z to the low-dimensional space to capture the high non-linearity in attributes.

Highly Non-linear Structure

To capture the highly non-linear structure, each branch in Figure 1 is an autoencoder. Autoencoder is a powerful unsupervised deep model for feature learning. It has been widely used for various machine learning applications [Jiang *et al.*, 2016]. The basic autoencoder contains three layers, they are the input layer, the hidden layer, and the output layer, which are defined as follows:

$$h_i = \sigma(W^{(1)}x_i + b^{(1)}), \quad \hat{x}_i = \sigma(W^{(2)}h_i + b^{(2)}) \,. \tag{1}$$

Here, $x_i \in \Re^d$ is the i-th input data, $h_i \in \Re^{d'}$ is the hidden representation from the encoder, and $\hat{x}_i \in \Re^d$ is the reconstructed data point from the decoder. $\theta = \{W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}\}$ are model parameters. $\sigma(.)$ denotes the non-linear activation function.

One can learn model parameters by minimizing the reconstruction error:

$$\min_{\theta} \sum_{i=1}^{n} ||\hat{x}_i - x_i||_2^2 \,. \tag{2}$$

To capture the high non-linearity in the topological structure and attributes, the two branches in Figure 1 employ K layers in the encoder as follows:

$$\begin{aligned} h_i^{(1)} &= \sigma(W^{(1)}x_i + b^{(1)}), \\ &\cdots \\ h_i^{(K)} &= \sigma(W^{(K)}h_i^{(K-1)} + b^{(K)}) \,. \end{aligned} \tag{3}$$

Correspondingly, there will be K layers in the decoder. Here, $h_i^{(K)}$ is the desired low-dimensional representation of the i-th node.

For our approach, the input of the first branch in Figure 1 is the high-order proximity matrix to capture the non-linearity in the topological structure. The explanation is delayed to the next section. The input of the second branch is the attribute matrix Z to capture the non-linearity in the attribute. Here, we denote the learned representation from the topological structure and attributes as H^M and H^Z respectively.

Proximity Preservation

To preserve the **semantic proximity**, we minimize the reconstruction loss between the input Z of the encoder and the output $\hat{Z}$ of the decoder:

$$L_s = \sum_{i=1}^{n} \|\hat{Z}_{i\cdot} - Z_{i\cdot}\|_2^2 \,. \tag{4}$$

The reason is disclosed in [Salakhutdinov and Hinton, 2007]. Specifically, the reconstruction loss can enforce the neural network to capture the data manifold smoothly and thus can preserve the proximity among samples [Wang *et al.*, 2016]. Therefore, by minimizing the reconstruction loss, our approach can preserve the semantic proximity in attributes.

Similarly, to preserve the **high-order proximity**, we also minimize the reconstruction loss as follows:

$$L_h = \sum_{i=1}^{n} \|\hat{M}_{i\cdot} - M_{i\cdot}\|_2^2 \,. \tag{5}$$

Specifically, the high-order proximity M denotes the neighborhood structure. If two nodes have similar neighborhood structure, which means $M_{i\cdot}$ and $M_{j\cdot}$ are similar, the learned representation $H_{i\cdot}^M$ and $H_{j\cdot}^M$ by minimizing the reconstruction loss will also be similar with each other.

As discussed before, we need to preserve the **first-order proximity** which captures the local structure. Recall Definition 1, two nodes are similar if there exists an edge between them. Thus, to preserve this proximity, we maximize the following likelihood estimation:

$$L_f = \prod_{E_{ij}>0} p_{ij} \,, \tag{6}$$

where p_{ij} is the joint probability between the i-th node and the j-th node. Note that we should preserve the first-order proximity in the topological structure and attributes simultaneously so that we can obtain the consistent result between these two kinds of information. For the topological structure, the joint probability is defined as follows:

$$p_{ij}^M = \frac{1}{1 + \exp(-H_{i\cdot}^M (H_{j\cdot}^M)^T)} \,. \tag{7}$$

Similarly, the joint probability based on the attribute is defined as follows:

$$p^Z_{ij} = \frac{1}{1+\exp(-H^Z_{i\cdot}(H^Z_{j\cdot})^T)} \ . \tag{8}$$

Thus, we can preserve the first-order proximity in the topological structure and attributes simultaneously by minimizing the negative log-likelihood as follows:

$$L_f = -\sum_{E_{ij}>0} \log p^M_{ij} - \sum_{E_{ij}>0} \log p^Z_{ij} \ . \tag{9}$$

Consistent and Complementary Representation

Since the topological structure and attributes are the two-modal information of the same network, we should guarantee the learned representation from them is consistent [Gao *et al.*, 2015a; 2015b]. On the other hand, these two kinds of information describe different aspects of the same node, providing complementary information. Thus, the learned representation should also be complementary. All in all, how to learn the *consistent* and *complementary* low-dimensional representation is very important.

A direct and simple method is to concatenate these two representations H^M and H^Z directly as the embedding result. Although this method can maintain the complementary information between two modalities, yet it cannot guarantee the consistency between these two modalities. Another widely used method is to enforce the two branches in Figure 1 to share the same highest encoding layer, that is $H^M = H^Z$. Although this method can guarantee the consistency between two modalities, it will lose too much complementary information from two modalities due to the exactly same highest encoding layer. Therefore, how to combine the topological structure and attributes together for attributed network embedding is a challenging problem.

To address this challenging problem, we propose to maximize the following likelihood estimation:

$$L_c = \prod_{i,j}^{n} p^{s_{ij}}_{ij}(1-p_{ij})^{1-s_{ij}} \ , \tag{10}$$

where p_{ij} is the joint distribution between two modalities, which is defined as follows:

$$p_{ij} = \frac{1}{1+\exp(-H^M_{i\cdot}(H^Z_{j\cdot})^T)} \ . \tag{11}$$

Additionally, $s_{ij} \in \{0,1\}$ denotes whether $H^M_{i\cdot}$ and $H^Z_{j\cdot}$ are from the same node. In detail, $s_{ij} = 1$ if $i = j$. Otherwise, $s_{ij} = 0$. Furthermore, Eq. (10) is equivalent to minimize the negative log-likelihood as follows:

$$L_c = -\sum_i \{\log p_{ii} - \sum_{j\neq i} \log(1-p_{ij})\} \ . \tag{12}$$

By minimizing Eq. (12), we can enforce $H^M_{i\cdot}$ and $H^Z_{j\cdot}$ as consistent as possible when they are from the same node while pushing away them when they are from different nodes. On the other hand, they are not exactly same, thus they can preserve the complementary information in each modality.

However, the second term in the right hand side of Eq. (12) is over strict. For instance, if two nodes i and j are similar according to the first-order proximity, the representation $H^M_{i\cdot}$ and $H^Z_{j\cdot}$ should also be similar although they are from different nodes. That is to say we should not push them away. Thus, we relax Eq. (12) as follows:

$$L_c = -\sum_i \{\log p_{ii} - \sum_{E_{ij}=0} \log(1-p_{ij})\} \ . \tag{13}$$

Here, we push $H^M_{i\cdot}$ and $H^Z_{j\cdot}$ together when they are from the same node while pushing them away when two nodes are not connected.

As a result, to preserve proximities and learn the consistent and complementary representation, we optimize the following objective function jointly:

$$\begin{aligned} L = &-\sum_{E_{ij}>0} \log p^M_{ij} - \sum_{E_{ij}>0} \log p^Z_{ij} + \sum_{i=1}^{n} \|\hat{M}_{i\cdot} - M_{i\cdot}\|_2^2 \\ &+ \sum_{i=1}^{n} \|\hat{Z}_{i\cdot} - Z_{i\cdot}\|_2^2 - \sum_i \{\log p_{ii} - \sum_{E_{ij}=0} \log(1-p_{ij})\} \ . \end{aligned} \tag{14}$$

By minimizing this problem, we can obtain $H^M_{i\cdot}$ and $H^Z_{i\cdot}$, then we concatenate them as the final low-dimensional representation of the node such that we can preserve the consistent and complementary information from the topological structure and attributes.

3.3 Most Negative Sampling Strategy

Take a more detailed look at Eq. (13), it is to solve the following problem with respect to each node:

$$L_{c_i} = -\log p_{ii} - \sum_{j:s.t.E_{ij}=0} \log(1-p_{ij}) \ . \tag{15}$$

In practice, the adjacency matrix E is very sparse since many edges are not discovered. However, the undiscovered edge does not really mean two nodes are not similar. If we push away two potential similar nodes, the learned representation will become worse.

In detail, the gradient of L_{c_i} with respect to $H^M_{j\cdot}$ where $E_{ij} = 0$ is

$$\frac{\partial L_{c_i}}{\partial H^M_{j\cdot}} = p_{ij} H^Z_{i\cdot} \ . \tag{16}$$

According to the gradient descent method, the updating rule for $H^M_{j\cdot}$ is $H^M_{j\cdot} \leftarrow H^M_{j\cdot} - \alpha p_{ij} H^Z_{i\cdot}$ where α is the step size. Since $H^Z_{i\cdot}$ is constant when updating $H^M_{j\cdot}$, p_{ij} determines the updating rule. Furthermore, p_{ij} depends on $H^M_{i\cdot}(H^Z_{j\cdot})^T$. If two nodes i and j are similar potentially but there is no direct link, p_{ij} will be large so that $H^M_{j\cdot}$ will be pushed much away from $H^Z_{i\cdot}$. As a result, the embedding result will become worse and worse.

To alleviate this problem, we propose a most negative sampling strategy to get a robust embedding result. Specifically, at each iteration, we calculate the similarity between H^M and H^Z by $P = H^M(H^Z)^T$. Then, for each node i, we select the most negative sample as follows:

$$j = \arg\min_{j, E_{ij}=0} P_{ij} \ . \tag{17}$$

Dataset	#Nodes	#Edges	#Attributes	#Labels
Cora	2,708	5,278	1,433	7
Citeseer	3,312	4,660	3,703	6
Wiki	2,405	12,761	4,973	17
PubMed	19,717	44,338	500	3

Table 1: Description of Benchmark Datasets

Based on this negative sample, the objective function Eq. (15) becomes

$$L_{c_i} = -\log p_{ii} - \log(1 - p_{ij}) \,, \tag{18}$$

where j is sampled based on Eq. (17). With this most negative sampling strategy, the potential similar nodes will not be violated as far as possible. Thus, the embedding result will be more robust.

Sampling Complexity The sampling complexity is dominated by the computation of the similarity P whose complexity is $O(n^2)$ while the complexity of Eq. (13) is also $O(n^2)$. Note that we omit the complexity of computing each p_{ij}. Thus, there is no much increased overhead to employ our sampling strategy. As a result, our proposed sampling strategy is efficient and effective.

4 Experiments

4.1 Experiment Settings

Datasets In our experiments, we employ four benchmark datasets [1]: Cora, Citeseer, PubMed, and Wiki. The first three datasets are paper citation networks. The edge of each network is the citation link. The attribute of each node is the bag-of-words representation of the corresponding paper. Wiki is a network with nodes as web pages. The link among different nodes is the hyperlink in the web page. We summarize the statistics of these benchmark datasets in Table 1.

Dataset	#neurons in each layer	Dataset	#neurons in each layer
Cora	2708-200-100 1433-200-100	Citeseer	3312-200-100 3703-500-100
Wiki	2405-200-100 4973-500-100	PubMed	19717-500-100 500-200-100

Table 2: The architecture of our approach for different datasets. For each dataset, the first row corresponds to the topological structure. The second row corresponds to node attributes. Here, we only show the architecture of the encoder. The decoder reverses the encoder.

Baselines To evaluate the performance of our proposed DANE, we compare it with 9 baseline methods, including 4 plain network embedding methods and 5 attributed network embedding methods. The former category contains DeepWalk [Perozzi *et al.*, 2014], Node2Vec [Grover and Leskovec, 2016], GraRep [Cao *et al.*, 2015], and LINE [Tang *et al.*, 2015]. The latter one includes TADW [Yang *et al.*, 2015], ANE [Huang *et al.*, 2017b], Graph Auto-Encoder (GAE) [Kipf and Welling, 2016b], Variational Graph Auto-Encoder (VGAE) [Kipf and Welling, 2016b], and SAGE [Hamilton *et al.*, 2017]. For DeepWalk and Node2Vec, we set the window size as 10, the walk length as 80, the number of walks as 10. For GraRep, the maximum transition step is set to 5. For LINE, we concatenate the first-order and second-order result together as the final embedding result. For the rest baseline methods, their parameters are set following the original papers. At last, the dimension of the node representation is set as 200.

To obtain the high-order matrix M in Definition 2, we utilize the random walk as DeepWalk [Perozzi *et al.*, 2014] to obtain the high-order context of each node and then construct the adjacency matrix M with a sliding window size as 10. Moreover, the architecture of our approach for four datasets is summarized in Table 2. We use LeakyReLu [Maas *et al.*,] as the activation function.

Method	10%		30%		50%	
	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
DeepWalk	0.7568	0.7498	0.8064	0.7943	0.8287	0.8177
Node2Vec	0.7477	0.7256	0.8201	0.8121	0.8235	0.8162
GraRep	0.7568	0.7441	0.7927	0.7893	0.7999	0.7921
LINE	0.7338	0.7191	0.8122	0.8105	0.8353	0.8254
TADW	0.7510	0.7234	0.8006	0.7801	0.8354	0.8187
ANE	0.7203	0.7150	0.8027	0.7906	0.8117	0.7987
GAE	0.7691	0.7573	0.8059	0.7921	0.8095	0.7989
VGAE	**0.7888**	0.7736	0.8054	0.7909	0.8117	0.7994
SAGE	0.7633	0.7475	0.8096	0.7999	0.8154	0.8080
DANE	0.7867	**0.7748**	**0.8281**	**0.8127**	**0.8502**	**0.8377**

Table 3: Node classification result of Cora.

Method	10%		30%		50%	
	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
DeepWalk	0.5052	0.4645	0.5783	0.5329	0.5900	0.5486
Node2Vec	0.5233	0.4832	0.6110	0.5651	0.6335	0.5972
GraRep	0.4817	0.4589	0.5511	0.5118	0.5707	0.5048
LINE	0.5139	0.4726	0.5761	0.5384	0.6075	0.5700
TADW	0.6048	0.5344	0.6481	0.5769	0.6578	0.5897
ANE	0.5877	0.5451	0.6718	0.6174	0.7071	0.6596
GAE	0.6058	0.5532	0.6550	0.5814	0.6540	0.5808
VGAE	0.6115	0.5662	0.6386	0.5824	0.6443	0.5837
SAGE	0.5351	0.4988	0.6304	0.5948	0.6528	0.6137
DANE	**0.6444**	**0.6043**	**0.7137**	**0.6718**	**0.7393**	**0.6965**

Table 4: Node classification result of Citeseer.

4.2 Results and Analysis

Node Classification To show the performance of our proposed DANE, we conduct node classification on the learned node representations. Specifically, we employ ℓ_2-regularized Logistic Regression as the classifier. To make a comprehensive evaluation, we randomly select $\{10\%, 30\%, 50\%\}$ nodes as the training set and the rest as the testing set respectively. With these randomly chosen training sets, we use five-fold cross-validation to train the classifier and then evaluate the classifier on the testing sets. To measure the classification result, we employ Micro-F1 (Mi-F1) and Macro-F1 (Ma-F1) as metrics. The classification results are shown in Table 3,4,5,6 respectively. From these four tables, we can find that our proposed DANE achieves significant improvement compared with plain network embedding approaches, and beats other attributed network embedding approaches in most situations.

[1] https://linqs.soe.ucsc.edu/data

Method	10%		30%		50%	
	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
DeepWalk	0.5621	0.4536	0.6479	0.5267	0.6675	0.5942
Node2Vec	0.5603	0.4131	0.6099	0.4760	0.6376	0.5203
GraRep	0.5801	0.4393	0.6223	0.5143	0.6642	0.5341
LINE	0.5806	0.4634	0.6538	0.5425	0.6766	0.5656
TADW	0.7266	**0.6300**	0.7565	0.6434	0.7764	0.6519
ANE	0.6263	0.5298	0.7144	0.5907	0.7298	0.6817
GAE	0.6245	0.4842	0.6526	0.5038	0.6567	0.5076
VGAE	0.6591	0.5215	0.6817	0.5621	0.7041	0.5790
SAGE	0.6259	0.5179	0.6764	0.5904	0.6866	0.6039
DANE	**0.7293**	0.6180	**0.7702**	**0.6597**	**0.7839**	**0.6838**

Table 5: Node classification result of Wiki.

Method	10%		30%		50%	
	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
DeepWalk	0.8047	0.7873	0.8168	0.8034	0.8156	0.8034
Node2Vec	0.8027	0.7849	0.8110	0.7965	0.8103	0.7981
GraRep	0.7951	0.7785	0.8031	0.7901	0.8051	0.7937
LINE	0.8037	0.7892	0.8129	0.8007	0.8110	0.7994
TADW	0.8358	0.8343	0.8586	0.8584	0.8643	0.8633
ANE	0.7977	0.7875	0.8263	0.8191	0.8284	0.8203
GAE	0.8285	0.8238	0.8310	0.8263	0.8306	0.8257
VGAE	0.8299	0.8240	0.8350	0.8291	0.8361	0.8299
SAGE	0.8170	0.8090	0.8250	0.8160	0.8267	0.8176
DANE	**0.8608**	**0.8579**	**0.8731**	**0.8706**	**0.8775**	**0.8749**

Table 6: Node classification result of PubMed.

Node Clustering To show the performance of our proposed approach on the unsupervised task, we conduct node clustering on the learned node representations. Here, we employ K-means as the clustering method and use clustering accuracy as the metric. The clustering result is summarized in Table 7. From this table, we can find that our approach achieves much better clustering performance than the others for most cases, which further verifies the effectiveness of our proposed DANE.

Network Visualization To further show the embedding result of our approach, we visualize the node representation by using t-SNE [Maaten and Hinton, 2008]. Due to the space limitation, we only post the result of three representative baseline methods for the Cora dataset. The visualization result is shown in Figure 2. From Figure 2, we can find that our approach can achieve more compact and separated clusters compared with the rest baseline methods. Thus, our approach can achieve better performance on both supervised and unsupervised tasks.

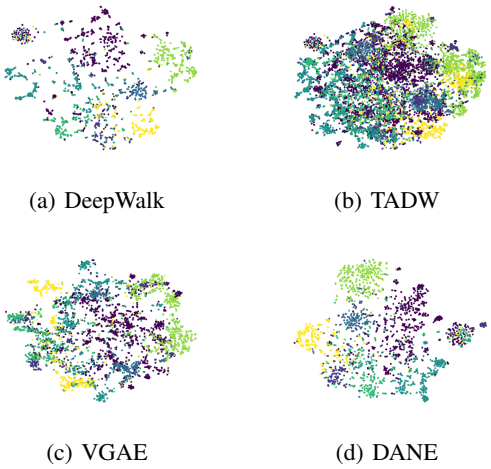
Effect of the Sampling Strategy To evaluate the effect of our sampling strategy, we compare it with two alternative methods. The first one is the random sampling method (DANE-RS). The second one is the importance sampling method (DANE-IS). For DANE-IS, we use $p^i = \frac{\exp(-P_{ij})}{\sum_j \exp(-P_{ij})}, \forall i$ as the sampling probability. Intuitively, this probability will let dissimilar points have large probability. With these two sampling methods, we learn node representations and then conduct node classification on the learned node representations for the Cora dataset. The result is shown in Table 8. We can find that DANE-RS has the worst performance since it fails to discriminate different negative samples. DANE-IS has a better result, but it is still worse than our method. Because our proposed sampling strategy can always find the most negative samples to refine node representations as far as possible.

Method	Cora	Citeseer	Wiki	PubMed
DeepWalk	0.6813	0.4145	0.4286	0.6660
Node2Vec	0.6473	0.4504	0.3792	0.6754
GraRep	0.5579	0.3777	0.3800	0.5501
LINE	0.4789	0.3913	0.4087	0.6614
TADW	0.5993	**0.6642**	0.4257	0.6257
ANE	0.3829	0.2557	0.3006	0.4486
GAE	0.6410	0.3677	0.4191	0.5915
VGAE	0.5546	0.3744	0.4445	0.6721
SAGE	0.6022	0.4589	0.4074	0.6020
DANE	**0.7027**	0.4797	**0.4731**	**0.6942**

Table 7: Clustering Accuracy

Method	10%		30%		50%	
	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
DANE-RS	0.7301	0.7151	0.8038	0.7887	0.8125	0.8015
DANE-IS	0.7748	0.7635	0.8228	0.8091	0.8443	0.8308
DANE	**0.7867**	**0.7748**	**0.8281**	**0.8127**	**0.8502**	**0.8377**

Table 8: Cora: Node classification of different sampling methods.



(a) DeepWalk (b) TADW

(c) VGAE (d) DANE

Figure 2: Visualization of different approaches for the Cora dataset.

5 Conclusion

In this paper, we propose a novel deep attributed network embedding approach, which can capture the high non-linearity and preserve the proximity both in the topological structure and node attributes. Meanwhile, the proposed approach can learn a consistent and complementary representation from the topological structure and node attributes. The effectiveness has been verified by extensive experiments.

Acknowledgements

This work was partially supported by the following grants: NSF-IIS 1302675, NSF-IIS 1344152, NSF-DBI 1356628, NSF-IIS 1619308, NSF-IIS 1633753, NIH R01 AG049371.

References

[Belkin and Niyogi, 2001] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[Cao *et al.*, 2015] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900. ACM, 2015.

[Gao and Huang, 2018] Hongchang Gao and Heng Huang. Self-paced network embedding. In *KDD*, 2018.

[Gao *et al.*, 2015a] Hongchang Gao, Chengtao Cai, Jingwen Yan, Lin Yan, Joaquin Goni Cortes, Yang Wang, Feiping Nie, John West, Andrew Saykin, Li Shen, et al. Identifying connectome module patterns via new balanced multigraph normalized cut. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 169–176. Springer, 2015.

[Gao *et al.*, 2015b] Hongchang Gao, Feiping Nie, Xuelong Li, and Heng Huang. Multi-view subspace clustering. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4238–4246. IEEE Computer Society, 2015.

[Grover and Leskovec, 2016] 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. ACM, 2016.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1025–1035, 2017.

[He and Niyogi, 2003] Xiaofei He and Partha Niyogi. Locality preserving projections. In *NIPS*, volume 16, 2003.

[Huang *et al.*, 2017a] Xiao Huang, Jundong Li, and Xia Hu. Accelerated attributed network embedding. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 633–641. SIAM, 2017.

[Huang *et al.*, 2017b] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 731–739. ACM, 2017.

[Jiang *et al.*, 2016] Wenhao Jiang, Hongchang Gao, Fu-lai Chung, and Heng Huang. The l2, 1-norm stacked robust autoencoders for domain adaptation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1723–1729. AAAI Press, 2016.

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

[Kipf and Welling, 2016b] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[Maas *et al.*,] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[Marsden and Friedkin, 1993] Peter V Marsden and Noah E Friedkin. Network studies of social influence. *Sociological Methods & Research*, 22(1):127–151, 1993.

[McPherson *et al.*, 2001] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[Pan *et al.*, 2016] S Pan, J Wu, X Zhu, C Zhang, and Y Wang. Tri-party deep network representation. In *International Joint Conference on Artificial Intelligence*. AAAI Press/International Joint Conferences on Artificial Intelligence, 2016.

[Perozzi *et al.*, 2014] 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. ACM, 2014.

[Salakhutdinov and Hinton, 2007] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *RBM*, 500(3):500, 2007.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[Wang *et al.*, 2016] 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*, pages 1225–1234. ACM, 2016.

[Yang *et al.*, 2010] Zi Yang, Jingyi Guo, Keke Cai, Jie Tang, Juanzi Li, Li Zhang, and Zhong Su. Understanding retweeting behaviors in social networks. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1633–1636. ACM, 2010.

[Yang *et al.*, 2015] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *IJCAI*, pages 2111–2117, 2015.