A3embed: Attribute Association Aware Network Embedding

Jihwan Lee
Amazon Alexa Brain & Purdue University
Seattle, Washington
jihwl@amazon.com

Sunil Prabahkar
Purdue University
West Lafayette, Indiana
sunil@purdue.edu

ABSTRACT

Network embedding aims to learn low-dimensional vector representations for nodes in a network that preserve structural characteristics. It has been shown that such representations are helpful in several graph mining tasks such as node classification, link prediction, and community detection. Some recent works have attempted to extend the approach to attributed networks in which each node is associated with a set of attribute values. They have focused on homophily relationships by forcing nodes with similar attribute values to obtain similar vector representations. This is unnecessarily restrictive and misses the opportunity to harness other types of relationships revealed by patterns in attribute values of connected nodes for learning insightful relationships. In this paper, we propose a new network attributed embedding framework called *A3embed* that is aware of attribute associations. *A3embed* favors significant attribute associations, not merely homophily relationships, which contributes to its robustness to diverse attribute vectors and noisy links. The experimental results on real-world datasets demonstrate that the proposed framework achieves better performance on different graph mining tasks compared to existing models.

ACM Reference Format:
Jihwan Lee and Sunil Prabahkar. 2018. *A3embed*: Attribute Association Aware Network Embedding. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3184558.3191563

1 INTRODUCTION

Data mining and machine learning tasks that aim to exract insightful information from real-world data must increasingly handle complex network data. However, it is not realistic to apply standard machine learning models directly to network data because the network itself lacks fruitful feature representations that can provide informative patterns among nodes and links. To overcome this challenge, researchers have proposed methods for learning new network representations that are usually represented by low-dimensional continuous vectors. Such vectors in a continuous feature space represent nodes in a more abstract form while preserving structural proximities among the nodes and thus are better suitable for various data mining and machine learning tasks.

While recently proposed network embedding algorithms show acceptable performance on various tasks [2, 3, 13, 15, 18, 20], they are limited to networks without attributes. However, an increasing number of real-world objects and applications are modeled with attributed networks and it has become increasingly important to analyze the attributes together with network structure. For example, in social networks such as Twitter and Facebook where user profile information is captured using attribute values, many users that have similar attributes are not connected to each other. That is, structural proximity is not sufficient to explain whether nodes in a network are similar or dissimilar. In that case, if available, node attributes can bring us a huge opportunity to capture the nodes' underlying similarity.

Alternative methods taking into account node attribute values in network embedding have been proposed more recently [4, 5]. They basically have the same motivation, where nodes with similar attribute values are located closely in the low-dimensional embedding space. It seems quite reasonable because many previous works have shown that nodes in a network tend to establish homophily relationship in terms of their attributes [6, 9, 16]. However, there actually exist more diverse relationships in real-world networks [7]. Also, even though such relationships may not be observed as frequently as homophily relationships, they can be more important for understanding various dynamics in complex networks and can be captured by considering statistical significance [7]. Unfortunately, the notion of attribute associations, defined as co-occurred attribute values between connected nodes, along with their significance has been ignored by existing network embedding methods despite its potential impact on network embedding.

Consider an attribute network where each node is associated with its attribute values. A pattern of node attribute values which co-occur between connected nodes might be of interest because it can reveal the type of relationships among nodes clearly along with their structural proximity. As the number of attributes increases, it is unlikely that homophily relationships alone are dominant in the entire network. For example, in a social network, people working at `Google` may establish many links to co-workers but they may have different alma maters (e.g. connections between {`Google`, `Stanford`} – {`Google`, `UCLA`}). Moreover, if we consider other attributes such as *nationality* and *major*, one expects to observe much more diverse patterns of co-occurring attribute values on the connections among Google employees. That is, the homophily relationship may not be sufficient to capture underlying similarities among the nodes in a network. In such cases it is clear why it is important to consider such patterns which are called attribute associations and possibly significant, as well as attribute similarity represented by homophily relationship for successful attributed network embedding. Even if a particular attribute association is frequently observed among connected nodes, the frequency itself does not tell us how meaningful it is. Whether it is really meaningful or not depends more on how many nodes hold the attribute vectors involved with the attribute association and how many of them are connected to each other.

WWW '18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3191563

The relative frequency of attribute association over the number of such nodes is more indicative of whether two nodes with the attribute association should be considered similar – and therefore be close to each other – in a low-dimensional embedding space.

In this paper, we study the network embedding problem, especially for attributed networks, and propose a new embedding method that exploits attribute associations in learning low-dimensional representations. We experimentally evaluate our proposed method *A3embed* using two real-world attributed networks including BlogCatalog and Flickr. We compare the performance of *A3embed* with state-of-the-art network embedding methods [3, 4, 18, 20]. Our observations from the experiments demonstrate that new network representations learned by *A3embed* can be better generalized to various prediction and visualization tasks. Especially, *A3embed* is superior to not only some baselines that use only network structure but also others that use augmented attribute information in addition to the structural information for all considered tasks.

We summarize the contributions of our proposed method as follows:

- We propose a novel network embedding method, called *A3embed* (**A**ttribute **A**ssociation **A**ware network **embed**ding). The method aims to obtain new representations of nodes in an attributed network by jointly modeling both structural and attribute information in the network while capturing attribute associations.
- We show why it is important to consider attribute associations on the task of network embedding.
- We empirically demonstrate how successfully *A3embed* learns new network representations in a low-dimensional space and how effective the learned representations are for downstream machine learning tasks on different real-world attributed networks.

The paper is organized as follows. In Section 2, we introduce previous works related to our problem and discuss how our problem differs from them. In Section 3, we define the problem of network embedding and provide basic background concepts, and then introduce a novel method to solve the problem of attributed network embedding. We present our experimental observations over different network embedding methods on real-world datasets in Section 4. Finally, we conclude the paper in Section 5.

2 RELATED WORK

The data mining and machine learning communities have been attracted to the problem of network embedding that aims to learn new representations for networks due to its practical importance in various applications such as node classification, link prediction, visualization, network compression, and clustering. Recently, many researchers have developed methods to learn network representations which are based on the Skip-gram model [10, 11] that aims to learn continuous feature representations for words in a corpus. *DeepWalk* [15] is the first work that established an analogy for networks by representing a network as a document. While a document includes a sequence of words, nodes in a network do not have any ordered sequences among them. The idea to obtain a sequence of nodes from a network is to consider a set of short truncated random walks as its own corpus, and the nodes as its own words.

Table 1: Basic notations

Notation	Meaning
$G = (V, E, X)$	attributed network
n	number of nodes
l	number of attributes
s_{ij}	weight of edge e_{ij}
$\mathbf{y_i}$	attribute embedding of node v_i
$\mathbf{z_i}$	structural embedding of node v_i
$\mathbf{h_i}$	joint representation of node v_i
$W_1^{(k)}$, $\mathbf{b_1}^{(k)}$	k-th layer weights and biases in attribute modeling
$W_2^{(k)}$, $\mathbf{b_2}^{(k)}$	k-th layer weights and biases in structure modeling
$W_3^{(k)}$, $\mathbf{b_3}^{(k)}$	k-th layer weights and biases in joint modeling
m_1, m_2, m_3	number of layers for each modeling component

Then the same optimization framework as one for the Skip-gram model can be applied to the set of node sequences obtained from repeated random walks. In [3] the authors proposed a new algorithmic framework called *node2vec* for learning continuous feature representations for nodes in networks using a biased random walk procedure that smoothly interpolate between Breadth-First Search (BFS) and Depth-First Search (DFS). However, those models exploit only network structure when learning feature representations without taking into account any other information such as node attributes.

In the case of citation networks, where nodes come with text information, such auxiliary information can be useful for learning richer representations. [22] proposed text-associated DeepWalk (*TADW*) that incorporates text features of nodes into network representation learning under the framework of matrix factorization. *TriDNR* [14], a tri-party deep network representation model, is based on a coupled neural network that exploits inter-node relationships, node-content correlation, and node-label correspondence in a network to learn an optimal representation for each node in the network. Even though *TADW* and *TriDNR* use rich information in addition to network structure for learning network representations, the text information is inherently different from node attributes in that text information itself includes a sequence of words so as to be easily exploited by neural networks based on the Skip-gram model.

While all the methods introduced above work only for networks without node attributes, [4, 5] exploit node attribute values to get richer representations for networks if node attribute are available. *LANE* [5] is a semi-supervised model that incorporates node labels into embedding representation learning for attributed networks. *AANE* [4] also learns low-dimensional representations based on the decomposition of attribute affinity and the embedding difference between connected nodes in a distributed way at scale. Both of the methods jointly model the network structure and node attributes but they are limited to attribute similarity. That is, nodes have a chance to have similar representations only when their attribute values are similar. In contrast, our proposed model considers more diverse patterns of co-occurring attribute values.

3 ATTRIBUTE ASSOCIATION AWARE NETWORK EMBEDDING

In this section, we first define the network embedding problem and then introduce our proposed method that learns network representations for attributed networks. Table 1 presents the notations we use throughout the paper.

3.1 Problem Definition

Consider an attributed network denoted by $G = (V, E, X)$ where $V = \{v_1, v_2, \cdots, v_n\}$ is a set of n number of nodes, $E = \{e_{ij}\}_{i,j=1}^{n}$ is a set of edges, and $X = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}\}$ is a set of attribute vectors, each of which is associated with a node in V. The attribute vector $\mathbf{x_i}$ of the node v_i that holds l different attributes is represented by a vector of l numerical values. An edge e_{ij} in E can be associated with its weight s_{ij} representing how strongly two individual nodes are connected to each other. If v_i and v_j are not connected by an edge, then $s_{ij} = 0$. In case of unweighted networks, $s_{ij} = 1$ for all edges e_{ij}.

Network embedding aims to learn new representations of nodes in a low-dimensional feature space by finding a mapping function $\mathcal{F} : V \mapsto \mathbb{R}^d$ where $d \ll n$ is the number of dimensions. Since a raw representation of nodes is too sparse, it is hard to observe interesting patterns, or obtain insightful knowledge, by applying standard machine learning models directly. However, the low-dimensional representations learned by network embedding may contain important underlying information over the network nodes in an abstract form. The key point to achieve good representations for attributed networks is to preserve the structural and attribute proximity of nodes [4, 5]. However, the sparsity of attribute space and diversity of attribute vectors render attribute proximity insufficient to account for actual similarity of nodes. In Section 3.2, we will introduce the notion of attribute association and explain why it needs to be considered for the network embedding task.

3.2 Attribute Associations

We first define an attribute association as follows,

Definition 3.1. Given two nodes v_i and v_j, the attribute association between them is defined as a relationship of co-occurred attribute values that appear in the pair of corresponding attribute vectors $\mathbf{x_i} = [x_i^1, x_i^2, \cdots x_i^l]$ and $\mathbf{x_j} = [x_j^1, x_j^2, \cdots x_j^l]$.

Every pair of nodes has its attribute association, and thus there are as many attribute associations as the number of edges in E if all the nodes in V have distinct attribute vectors. Note that an attribute association of $\mathbf{x_i}$ and $\mathbf{x_j}$ is associated with not only the nodes v_i and v_j but also any pairs of nodes that have the same attribute vectors as $\mathbf{x_i}$ and $\mathbf{x_j}$. As the number of attributes increases in a network, the sparsity of attribute vectors would be higher and it is more likely for nodes to have diverse attribute vectors. However, even though two nodes have different attribute vectors, it does not mean necessarily that they are not similar. That is because it is also possible for some different attribute values to share similar topics or be correlated to each other. For example, in a social network where each individual is associated with their personal profile, some users may have `google`, `facebook`, `J.P Morgan`, `Goldman Sachs`, and so on for the attribute *employer*. In terms of their context, `google` and `facebook`, Internet service companies, are closer to each other rather than to the other two finance companies, and vice versa, and thus it is expected that users working at `google` (or `J.P Morgan`) are more likely to be linked with users working at `facebook` (or `Goldman Sachs`) even if they have different values for the attribute. In other words, dissimilarity of node attribute values may not necessarily imply dissimilarity of nodes. This motivates us to consider various patterns of co-occured attribute values that are represented by attribute associations for network representation learning. Similarly, it is not always true that two nodes with exactly the same attribute vectors must be similar. Even though a number of nodes are associated with a particular attribute vector, if only a few of them are connected to each other, it is hard to say that all the nodes with the attribute vector are similar and should be located closely in the embedding space. Thus, it is important to consider statistically significant attribute associations for more insightful network analysis [7]. In this paper, we do not compute the actual statistical significance of attribute associations but introduce the basic idea of jointly modeling node attributes and network structure for the task of learning network representations. That is, for a given attribute association of $\mathbf{x_i}$ and $\mathbf{x_j}$, we say the attribute association between them is more **significant** than another association of $\mathbf{x_m}$ and $\mathbf{x_n}$ if the nodes with the association of $\mathbf{x_i}$ and $\mathbf{x_j}$ are more densely connected to each other compared to the connections among the nodes with $\mathbf{x_m}$ and $\mathbf{x_n}$. Such nodes with more significant associations should be closer to each other than ones with less significant associations in the embedding space. We explain how the notion of significance should be considered in 3.3.1.

3.3 *A3embed*

We now propose a new network embedding method called *A3embed* using attribute associations for attributed networks. *A3embed* consists of two parts: one is for modeling attribute associations and the other is for modeling network structure. The idea is straightforward. If two nodes share similar attribute values and/or the attribute association between them is significant, then they should be close to each other in the low-dimensional embedding space. Likewise, if two nodes share many common neighbors and therefore are structurally similar to each other, then they should be located closely as well. In this way, we can preserve both attribute proximity and structural similarity while keeping patterns of significant attribute associations in the embeddings. The overall framework of *A3embed* is illustrated in Figure 1.

3.3.1 Modeling Attribute Associations. We basically want to not only preserve the attribute similarity but also employ significant attribute associations. First of all, in order to model the attribute similarity among network nodes, we apply a deep autoencoder [1, 17, 20] to the set of all attribute vectors X. An autoencoder neural network, consisting of the encoder and decoder, is an unsupervised learning algorithm that applies backpropagation, setting the target values or outputs to be equal to the inputs. In other words, it tries to learn an approximation to the identity function, so as to output $\hat{\mathbf{x}}_\mathbf{i}$ that is similar $\mathbf{x_i}$, by having a non-linear function that encodes $\mathbf{x_i}$ to new representations $\mathbf{y_i}$ and another non-linear function that reconstructs $\hat{\mathbf{x}}_\mathbf{i}$ from $\mathbf{y_i}$. As a result, the learned $\mathbf{y_i}$ in the middle of the autoencoder can be considered as compressed and latent representations of $\mathbf{x_i}$. If we have multiple layers for the encoder and the decoder, then the latent representation $\mathbf{y_i}^{(k)}$ is formulated

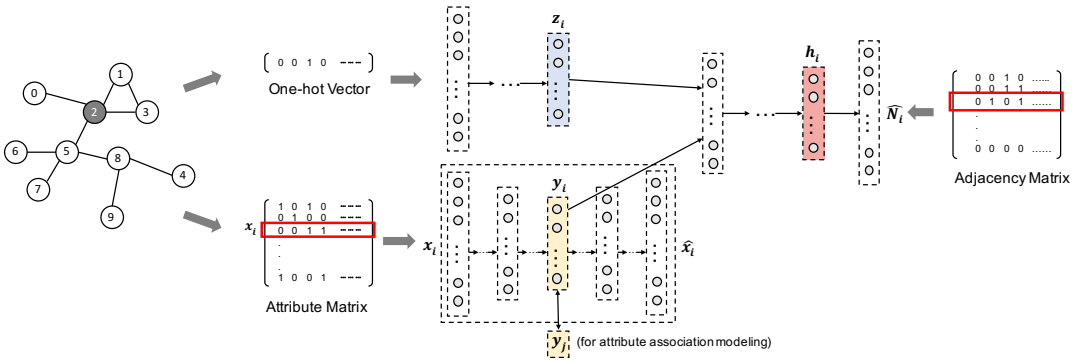


Figure 1: Framework of *A3embed*: For each node, its attribute vector and one-hot vector are fed into the deep model, and then its attribute and structural information are jointly modeled to predict its neighbors.

as follows:

$$\begin{aligned}\mathbf{y_i}^{(k)} &= \sigma(W_1^{(k)}\mathbf{y_i}^{(k-1)} + \mathbf{b_1}^{(k)}) \\ &= \sigma(W_1^{(k)}\sigma(W_1^{(k-1)}\mathbf{y_i}^{(k-2)} + \mathbf{b_1}^{(k-1)}) + \mathbf{b_1}^{(k)}) \\ &= \sigma(W_1^{(k)}(\cdots\sigma(W_1^{(1)}\mathbf{x_i} + \mathbf{b_1}^{(1)})\cdots) + \mathbf{b_1}^{(k)})\end{aligned} \tag{1}$$

where σ is an element-wise activation function such as a sigmoid function or a rectified linear unit. Similarly, the reconstruction $\hat{\mathbf{x}}_\mathbf{i}$ that has the same shape as $\mathbf{x_i}$ is mapped from $\mathbf{y_i}$ by stacking hidden layers of non-linear functions on the top of $\mathbf{y_i}$ in the reversed shape of the encoder. Then the autoencoder is trained to minimize reconstruction errors, represented by

$$\mathcal{L}_{sim} = \mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{i=1}^{n} \|\mathbf{x_i} - \hat{\mathbf{x}}_\mathbf{i}\|_2^2 \tag{2}$$

As discussed in [20], if an input vector, which is an attribute vector in our setting, is very sparse, then the autoencoder is prone to reconstruct zero values in an attribute vector rather than non-zero values. We avoid that by imposing a higher penalty to the reconstruction error of non-zero values than the error of zero-values as follows:

$$\mathcal{L}_{sim} = \sum_{i=1}^{n} \|(\mathbf{x_i} - \hat{\mathbf{x}}_\mathbf{i}) \odot \mathbf{t_i}\|_2^2 \tag{3}$$

where $\mathbf{t_i} = [t_{i1}, t_{i2}, \cdots, t_{il}]$ and $t_{ij} = \tau > 1$ for $j = 1, \cdots, l$ if v_i has a value for the j-th attribute, otherwise $t_{ij} = 1$. The $\odot$ operator performs an element-wise multiplication between two vectors.

As we put attribute vectors of nodes repeatedly into the autoencoder, nodes with similar attribute vectors must have similar latent representations $\mathbf{y}$. However, the objective function above is not enough to model potentially significant attribute associations that exist in many real-world applications as well.

The key idea of using significant attribute associations is to see, given an attribute association, how many nodes have the attribute vectors involved with the association and how frequently the association is observed over links among the nodes relatively. That is, if two nodes v_i and v_j are associated with a particular attribute association and its attribute vectors $\mathbf{x_i}$ and $\mathbf{x_j}$ appear many times on other connected nodes as well, then we make the corresponding latent representations $\mathbf{y_i}$ and $\mathbf{y_j}$ similar, no matter if they have dissimilar attribute values or not. Note that the frequency of attribute vectors itself may not be important. In contrast, if there are many nodes that hold either $\mathbf{x_i}$ or $\mathbf{x_j}$ but only few of them are connected, then we make $\mathbf{y_i}$ and $\mathbf{y_j}$ away from each other. The following objective function takes care of attribute associations with the idea above:

$$\mathcal{L}_{ass} = \sum_{i,j=1}^{n} s_{ij} \cdot \|\mathbf{y_i} - \mathbf{y_j}\|_2^2 \tag{4}$$

where s_{ij} is the edge weight between v_i and v_j and $s_{ij} = \delta < 0$ if there is no edge between v_i and v_j. For nodes v_i and v_j that are not connected, we give a negative penalty δ to their corresponding latent representations $\mathbf{y_i}$ and $\mathbf{y_j}$ such that they are not close. The effect of the negative penalty term δ guarantees that 1) even two nodes with the same attribute vectors could be apart if such attribute association is very rare in the network and 2) even two nodes with different attribute vectors could be close if other nodes with such vectors are connected more densely than usual. The choice of δ also controls how aggressively *A3embed* models attribute associations. If δ is very low, we rarely penalize attribute associations that appear between unconnected nodes. Thus, only node pairs with very significant attribute associations will be mapped in close proximity to each other in the embedding space.

3.3.2 Modeling Structural Proximity. While some existing works take into account preserving the first-order and second-order proximity simultaneously [18, 20], we focus on the second-order proximity, i.e., common neighbor structure. This is acceptable because the first-order proximity is already considered to some extent when modeling attribute associations. Of course, even though two nodes are directly connected, they may not have similar low-dimensional representations if the attribute association is too weak. This sounds reasonable unless a pair of connected nodes necessarily share common values on most attributes.

For a given a node v_i, we use its one-hot vector $\mathbf{v_i}$ as an input to *A3embed*. It is fed-forward into a multi-layer perceptron and its latent representation $\mathbf{z_i}$ is combined with another latent representation $\mathbf{y_i}$ to produce a joint representation $\mathbf{h_i}$ of both network structure and node attribute values. Then the joint representation $\mathbf{h_i}$ is further fed-forward into following hidden layers and the final

joint representation is used to predict neighbors of the node v_i. The formulations of the latent representations at each layer are as follows:

$$\begin{aligned}\mathbf{z_i}^{(k)} &= \sigma(W_2^{(k)}\mathbf{z_i}^{(k-1)} + \mathbf{b_2}^{(k)}) \\ &= \sigma(W_2^{(k)}(\cdots\sigma(W_2^{(1)}\mathbf{v_i} + \mathbf{b_2}^{(1)})\cdots) + \mathbf{b_2}^{(k)})\end{aligned} \tag{5}$$

$$\begin{aligned}\mathbf{h_i}^{(0)} &= [\omega\mathbf{y_i}, \mathbf{z_i}] \\ \mathbf{h_i}^{(k)} &= \sigma(W_3^{(k)}\mathbf{h_i}^{(k-1)} + \mathbf{b_3}^{(k)})\end{aligned} \tag{6}$$

where ω is a hyperparameter that controls weights on the latent representation from modeling node attributes when constructing the first joint representation by concatenation.

Lastly, we predict the neighbors $\mathcal{N}_i$ of the input node v_i using the final joint representation $\mathbf{h_i}$. The output vector $\hat{\mathcal{N}}_i$ in *A3embed* should be close to a row or column vector of an adjacency matrix indicating neighbor nodes, and thus it is a multi-label classification task. The predictive probabilities for the neighbor nodes in $\mathcal{N}_i$ are obtained independently by placing a vector of sigmoids. Then the output vector $\hat{\mathcal{N}}_i$ is computed as follows:

$$\begin{aligned}\hat{\mathcal{N}}_i &= [p(v_1|v_i), p(v_2|v_i), \cdots, p(v_n|v_i)] \\ &= [\frac{1}{1+e^{-\mathbf{u_1}\cdot\mathbf{h_i}}}, \frac{1}{1+e^{-\mathbf{u_2}\cdot\mathbf{h_i}}}, \cdots, \frac{1}{1+e^{-\mathbf{u_n}\cdot\mathbf{h_i}}}]\end{aligned} \tag{7}$$

where $\mathbf{u_j}$ is a column vector of the weight matrix between the last two layers, which corresponds to a contextual vector of the neighbor v_j. We then construct the loss function as:

$$\begin{aligned}\mathcal{L}_{net} &= -\sum_{i=1}^{n}\log p(v_1, v_2, \cdots, v_n|v_i) \\ &= -\sum_{i=1}^{n}\sum_{v_j\in\mathcal{N}_i}\log p(v_j|v_i)\end{aligned} \tag{8}$$

Modeling jointly the network structure based on neighbors and the attribute information including attribute similarity and significant attribute associations, our proposed method *A3embed* is trained while aiming to find optimal weight parameters in the following final objective function:

$$\arg\min_{f_1,f_2,f_3}\sum_{i=1}^{3}\lambda_i\cdot R(f_i) + \alpha\mathcal{L}_{sim} + \gamma\mathcal{L}_{ass} + \mathcal{L}_{net} \tag{9}$$

where f_i is a set of weight matrices and biases for each component in the deep neural network framework of *A3embed*, R is a regularization function which is defined as $R(f_i) = \frac{1}{2}\sum_{k=1}^{m_i}\|W_i^{(k)}\|_F^2$, and λ_i is a regularization term.

3.3.3 Optimization. Our goal is to find the optimal f_1, f_2, and f_3 that minimize the objective function formulated in Eq 9. We train *A3embed* using stochastic gradient descent. Specifically, we adopt RMSProp [19], an adaptive learning rate method, to update gradients during training. We omit the mathematical formulation of the partial derivative for each of the loss function because it is straightforward.

Table 2: Dataset Statistics

Name	Synthetic	BlogCatalog	Flickr
No. Nodes	1,024	5,196	7,575
No. Edges	varied	171,743	239,738
No. Attributes	1,000	8,189	12,047
No. Labels	varied	6	9

4 EXPERIMENTS

4.1 Datasets

We evaluate our proposed method as well as competitors using one synthetic and three real-world attributed networks. Each of the networks contains a set of nodes forming network edges and associated attribute vectors for each node. Table 2 presents the statistics of the network datasets.

Synthetic Attributed Network We generated synthetic attributed networks to show the robustness of our model to diverse attribute associations, not only homophily. The networks are generated using the stochastic block model [12, 21]. We first generate several disjoint connected components, each of which corresponds to a community representing a group of nodes with the same class label, where nodes in the same community are connected to each other with p probability and nodes are connected with q probability across different communities. Every node belonging to the same community shares the same attribute vector. We then adjust the connection probabilities and perturb attribute vectors to introduce non-homophily attribute associations between nodes in the same community as well as noisy links. Embeddings for such contrived attributed networks can reveal how a model is able to capture diverse attribute associations as well as underlying node similarities from noisy connections of network nodes. See more details in Section 4.5.

BlogCatalog [4, 5] BlogCatalog is a blogging platform where users can form a network connecting each other. Each blog has a short description and the keywords in the description are considered as attributes. Users can assign to their blogs a category that represents a class label.

Flickr [4, 5] Flickr is an online photo management and sharing website where users can establish connections to others. For attributes, we use as attributes a set of tags that describe users' specific interests in their photos. The groups to which users subscribe in the platform are considered as class labels.

4.2 Baselines

We evaluate the following baseline methods as well as *A3embed* for comparison. All the baselines were published recently and are known as good performers for network embedding. They are categorized into two groups. *node2vec*, *SDNE*, and *LINE* use only the network structure information while *AANE* uses both structural and attribute information. The brief descriptions of the baselines are as follows:

node2vec [3] *node2vec*, extending *DeepWalk* [15], is one of the state-of-the art methods for network embedding and it uses only structural information. It exploits truncated random walk sequences to obtain context nodes for a given node while allowing flexibility between homophily and structural equivalence, and then computes node embeddings by maximizing the likelihood of observing context nodes.

Table 3: Node classification performance of different methods over different training-test split ratios on BlogCatalog

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro F_1	LINE	0.656	0.672	0.686	0.691	0.684	0.691	0.679	0.683	0.689
	node2vec	0.514	0.541	0.609	0.624	0.635	0.641	0.639	0.649	0.655
	SDNE	0.551	0.617	0.651	0.678	0.679	0.693	0.684	0.690	0.692
	AANE	0.755	0.858	**0.884**	0.885	0.886	0.883	0.876	0.889	0.889
	A3embed	**0.837**	**0.866**	0.881	**0.888**	**0.888**	**0.894**	**0.901**	**0.912**	**0.917**
Micro F_1	LINE	0.661	0.676	0.691	0.696	0.691	0.697	0.684	0.689	0.704
	node2vec	0.521	0.545	0.614	0.631	0.642	0.648	0.646	0.657	0.669
	SDNE	0.556	0.620	0.654	0.682	0.686	0.698	0.688	0.697	0.702
	AANE	0.783	0.865	**0.889**	0.890	0.890	0.887	0.879	0.893	0.892
	A3embed	**0.841**	**0.868**	0.883	**0.891**	**0.891**	**0.897**	**0.902**	**0.913**	**0.915**

Table 4: Node classification performance of different methods over different training-test split ratios on Flickr

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro F_1	LINE	0.576	0.601	0.604	0.610	0.617	0.621	0.627	0.626	0.624
	node2vec	0.358	0.438	0.470	0.479	0.497	0.508	0.514	0.517	0.521
	SDNE	0.506	0.559	0.582	0.592	0.600	0.609	0.609	0.610	0.609
	AANE	0.754	0.781	0.818	0.843	0.847	0.858	0.861	0.865	0.872
	A3embed	**0.816**	**0.840**	**0.849**	**0.855**	**0.856**	**0.864**	**0.865**	**0.874**	**0.890**
Micro F_1	LINE	0.585	0.608	0.611	0.619	0.626	0.630	0.638	0.639	0.640
	node2vec	0.363	0.444	0.475	0.486	0.507	0.518	0.524	0.529	0.533
	SDNE	0.508	0.562	0.586	0.597	0.607	0.617	0.617	0.617	0.620
	AANE	0.781	0.806	0.831	0.850	0.855	0.863	0.865	0.869	0.877
	A3embed	**0.819**	**0.843**	**0.852**	**0.856**	**0.860**	**0.867**	**0.868**	**0.877**	**0.894**

SDNE [20] This method uses structural information only as well but focuses on the first-order and second-order proximity among nodes to preserve the network structure.

LINE [18] As in *SDNE*, *LINE* preserves the first-order and second-order proximity, but it does not model them jointly. They are considered separately to learn low-dimensional representations for each, and then concatenated. In our experiments, only the second-order proximity is used because it does not differ much from the concatenated representations, in terms of the effectiveness on downstream tasks.

AANE [4] *AANE* models and incorporates node attribute proximity into network embedding in a distributed way. It learns a low-dimensional representation based on the decomposition of attribute affinity and the embedding between connected nodes. The key difference from *A3embed* is that the node attribute information is learned in *A3embed* allowing implicit similarity and diverse relationships of attribute values whereas the node attributes have to be explicitly similar in *AANE*.

4.3 Experimental Setup

All experiments were conducted on a machine with Intel i5-4690K 3.50GHz CPU, 32 GB memory and GTX Titan X GPU, running 64bit Ubuntu 14.04. *A3embed* is implemented using TensorFlow 1.2.1 [1] in Python 2.7, and for the implementations of all the baselines, we use the source code from the authors.

All the methods we evaluate include various hyperparameters that may affect the performances of the methods significantly and thus need to be tuned. We basically seek optimal hyperparameter values through grid-search and run each of the baseline algorithms with multiple epochs until we achieve the best results. Note that all the notations we use in the following discussion are ones from the original papers for the methods. For *AANE*, we use the parameter values that are already specified in the source code written by the author for each dataset ($\lambda \in \{1e-6, 0.0425\}$ and $\rho \in \{4, 5\}$). For *node2vec*, the search strategy parameters p and q are set to 2 and 0.5 respectively, and we use typical values for any other parameters such as the length of random walk ($l = 80$) and the size of network neighborhoods ($k = 10$). For *SDNE*, we also use $\alpha = 1, \beta = 5, \gamma = 5$, and the shape of its autoencoder structure is the same as ones described in its paper. All the parameters in *LINE* are set as used in the paper, except that we vary the number of samples used for optimization to find the best performance. For *A3embed*, we found that the following parameter setting works best for both BlogCatalog and Flickr: $\alpha = 1$, $\tau = 5$, $\gamma = 5$, $\delta = -0.5$, $\omega = 0.5$, $\lambda_1 = \lambda_2 = \lambda_3 = 1$, and the learning rate is set to 0.001, For fair comparisons, the dimensionality of the embeddings is set to 200 for BlogCatalog and Flickr and 100 for synthetic networks for all the methods. In addition to $d = 200$, the impact of different embedding dimensions is discussed in Section 4.6.

4.4 Multi-Label Classification

One of the most common analytics tasks in network data is node classification, and so we evaluate the effectiveness of different network representations obtained from considered network embedding algorithms through a multi-label classification task on the real-world datasets. Every node in the network data is associated with one or more labels. Given a set of low-dimensional representations of the nodes generated by a network embedding algorithm, we randomly split them into training and test sets with varied ratios and train a classification model over the training nodes and their labels using the learned representations as features. Then, we see how accurately the models predict the labels of test nodes using Macro- and Micro-F_1 metrics. Here, for the classification model, we use a one-vs-rest support vector machine classifier provided by scikit-learn library [2].

[1] https://www.tensorflow.org

[2] https://scikit-learn.org/

Table 5: Node classification performance on synthetic attributed networks

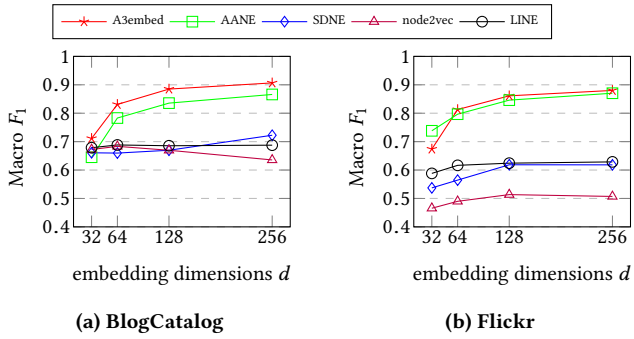
	$p = 0.3, q = 0.1, r = 5$		$p = 0.3, q = 0.3, r = 5$	
Algorithm	Macro F_1	Micro F_1	Macro F_1	Micro F_1
LINE	0.921	0.924	0.088	0.091
node2vec	0.911	0.912	0.061	0.068
SDNE	0.918	0.920	0.083	0.085
AANE	1.0	1.0	0.701	0.712
A3embed	1.0	1.0	1.0	1.0

Table 3 and Table 4 show the classification results of different methods on the two real-world attributed network, BlogCatalog and Flickr. By looking at the performance difference between structure-only based methods and joint models, it is clear that using attribute information, if available, is critical to learn better representations. Especially, *A3embed* almost consistently outperforms all the other competitive methods over different split ratios of training and test samples, which demonstrates the network representations learned from our proposed model can capture more meaningful underlying characteristics of the network nodes. Moreover, *A3embed* is very robust to even small training sample sizes. As we decreases the size of the training set, the improvement margin of *A3embed* over the baseline methods increases. This observation can tell us that our proposed method is better suited to many different real-world applications where only few nodes actually have labels.

4.5 Capturing Attribute Associations

A3embed not only takes into account attribute proximity, but also diverse attribute associations between different attribute vectors, whereas existing methods work only on homophily relationship. This property implies that the representation learning of *A3embed* is more robust and generalizes to various patterns of relationships between nodes. In order to highlight how robust *A3embed* is compared to the baselines, we generate synthetic attributed networks in such a way that we can control certain properties held by network data by changing link probabilities and attribute values, as described in 4.1. We start with a naive network where every node in the same community is assigned an identical attribute vector ($r = 1$) and nodes are more likely to be linked with others in the same community ($p > q$). We then change q such that nodes are connected across different communities. We also randomly divide the nodes in each community into r disjoint subsets and perturb the attribute values of the nodes such that there are r different attribute vectors in the same community. In this way, we have diverse attribute associations, not only homophily relationships.

Having different values of p, q, and r, we generate various synthetic attributed networks with ten communities and predict which communities the nodes in the test set belong to by using learned low-dimensional representations. Changing the value of r does not affect the behaviors of *LINE*, *node2vec*, and *SDNE* at all because it preserves the structural proximity only without using the node attributes. It is also not surprising that both *A3embed* and *AANE* perform the classification task with high accuracy if r is low, that is, diverse attribute associations are rare. Table 5 shows the methods' robustness to existence of diverse attribute associations. When $p = 0.3$, $q = 0.1$, and $r = 5$, while *LINE*, *node2vec*, and *SDNE* lose some accuracy due to the noisy links, *A3embed* and *AANE* classify every node perfectly. If nodes in the same community are tightly



(a) BlogCatalog **(b) Flickr**

Figure 2: Classification performance of learned representation over different embedding dimensions

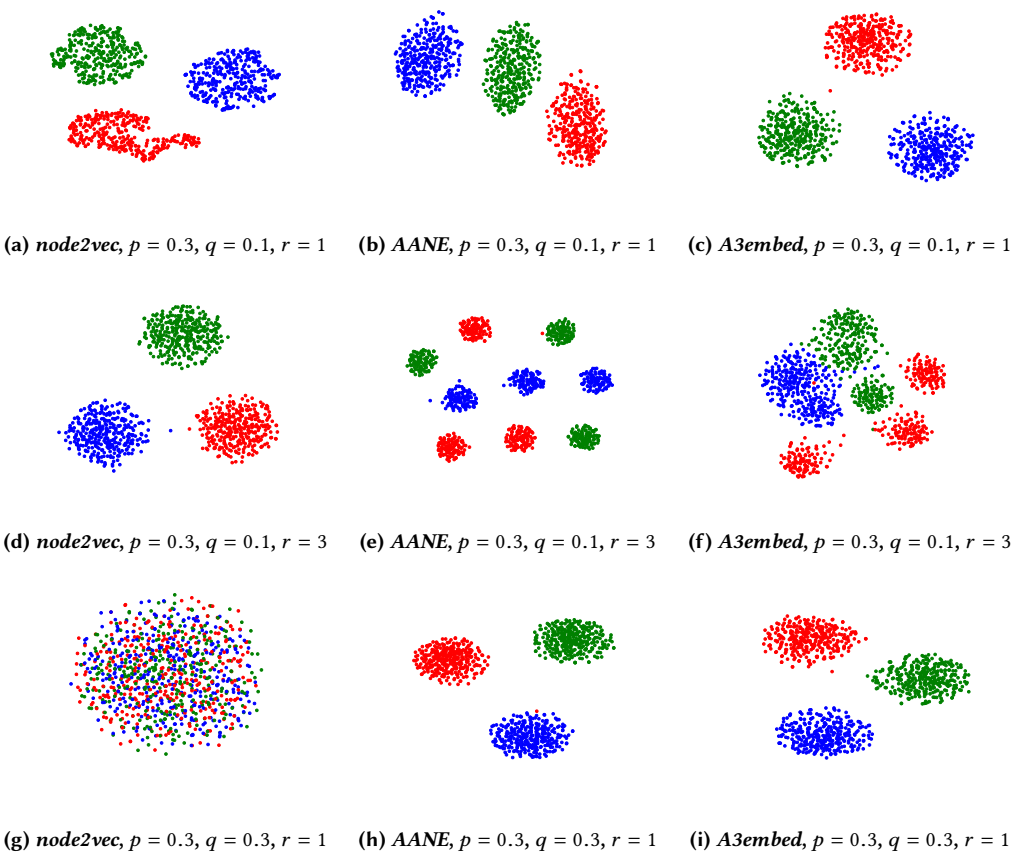
connected with a small fraction of noisy links, then the structural proximity can be a strong signal for such nodes to stay close in the low-dimensional embedding space even if r is high. However, it does not mean *AANE* is able to capture diverse attribute associations. We discuss more details in Section 4.7. If $p = 0.3$, $q = 0.3$, and $r = 5$, then the network structure is not helpful anymore (explaining poor performance of *LINE*, *node2vec*, and *SDNE*) and it becomes very important to be able to capture and model attribute associations. *A3embed* still achieves 100% accuracy but *AANE*'s performance gets worsen due to lack of its ability to model attribute associations.

4.6 Impact of Embedding Dimensions

We study how the classification performance of learned representations changes with respect to varying embedding dimensions $d \in \{32, 64, 128, 256\}$. Ideally, a network embedding method is expected to be able to learn good representations regardless of the embedding dimensions. Figure 2 illustrates the effect of embedding dimensions on node classification with the BlogCatalog and Flickr datasets. We here report only Macro-F_1 because we observed Macro-F_1 and Micro-F_1 have almost the same trend in this experiment. As demonstrated in Figure 2, *A3embed* and *AANE* work better as d increases while the other methods based on only network structure saturate or deteriorate after certain number of dimensions. Since *A3embed* and *AANE* use both network structure and node attributes for joint modeling, they have greater capacity to embed latent features compared to the other three. *node2vec* goes even worse when $d = 128$ or 256, which implies overfitting.

4.7 Visualization

In addition to measuring effectivenesses over different downstream tasks we have discussed so far, it is also very important to visualize a network because such visualizations can help us more intuitively understand how the network nodes are distributed and interact with each other. Since different network embedding methods preserve different properties of a network, they have different ability and interpretation of node visualization. We use the synthetic networks with different parameter settings as discussed in 4.5 and learn new representations of nodes using *A3embed*, *AANE*, and *node2vec*. We omit *SDNE* and *LINE* for the visualization task because they are basically not much different from *node2vec* in that all of them model



(a) ***node2vec***, $p = 0.3$, $q = 0.1$, $r = 1$ **(b)** ***AANE***, $p = 0.3$, $q = 0.1$, $r = 1$ **(c)** ***A3embed***, $p = 0.3$, $q = 0.1$, $r = 1$

(d) ***node2vec***, $p = 0.3$, $q = 0.1$, $r = 3$ **(e)** ***AANE***, $p = 0.3$, $q = 0.1$, $r = 3$ **(f)** ***A3embed***, $p = 0.3$, $q = 0.1$, $r = 3$

(g) ***node2vec***, $p = 0.3$, $q = 0.3$, $r = 1$ **(h)** ***AANE***, $p = 0.3$, $q = 0.3$, $r = 1$ **(i)** ***A3embed***, $p = 0.3$, $q = 0.3$, $r = 1$

Figure 3: Visualization of synthetic attributed networks. Color of a point indicates its community. (p: in-community link probability, q: cross-community link probability, r: number of distinct attribute vectors in a community)

only network structure. The learned representations are used as input to *t-SNE* [8] with its default parameter values.

Figure 3 illustrates visualization of low-dimensional representations of various synthetic attributed networks. The synthetic attributed networks are built with different parameter settings (p, q, and r) and all of them include three communities, each of which is indicated by a color. When $p = 0.3$, $q = 0.1$, and $r = 1$, all the methods produce nice visualization where every community is well-separated (Figure 3a, 3b, and 3c). However, if the number of distinct attribute vectors in each community increases ($r = 3$) and thus there are diverse attribute associations, *AANE* fails to keep every node in a community close to each other (Figure 3e). It makes r disjoint groups for each community in its visualization because *AANE* optimizes its objective based on only homophily relationship. In contrast, *A3embed* captures the attribute associations between even different attribute vectors and the nodes in the same community are better clustered together than *AANE* (Figure 3f). *node2vec* is not affected by changing r due to its inability to model node attributes (Figure 3d). If we have large numbers of noisy links (those that cross different communities), it must be critical to benefit from modeling node attributes. While *A3embed* and *AANE* can visualize the network in perfect shape (Figure 3h and 3i), *node2vec* fails to visualize the nodes correctly due to the absence of any clues to differentiate the in-community and cross-community links in the network (Figure 3g).

5 CONCLUSION

In this paper, we propose a novel network embedding method, called *A3embed*, for attributed networks. *A3embed* learns new network representations by jointly exploiting network structural information and node attribute values. While preserving the network structure, it also uses various attribute associations, not limited to homophily relationships, among nodes. The existence of non-homophily but significant attribute associations in networks can play an important role for finding well-represented embeddings. The experiments are conducted on two real-world attributed networks to demonstrate the effectiveness of *A3embed* in some downstream tasks such as multi-label classification and visualization. We also use synthetic attributed networks to show how well *A3embed* is able to capture diverse attribute associations. The experimental results show that our proposed method outperforms other network embedding methods in different downstream machine learning tasks, which confirms the importance of using attribute associations in representation learning.

REFERENCES

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[2] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep Neural Networks for Learning Graph Representations.. In *AAAI*. 1145–1152.

[3] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

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

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

[6] Myunghwan Kim and Jure Leskovec. 2012. Multiplicative attribute graph model of real-world networks. *Internet Mathematics* 8, 1-2 (2012), 113–160.

[7] Jihwan Lee, Keehwan Park, and Sunil Prabhakar. 2016. Mining Statistically Significant Attribute Associations in Attributed Graphs. In *IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 991–996.

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

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

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

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[12] Krzysztof Nowicki and Tom A B Snijders. 2001. Estimation and prediction for stochastic blockstructures. *J. Amer. Statist. Assoc.* 96, 455 (2001), 1077–1087.

[13] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. 2016. Tri-party Deep Network Representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press, 1895–1901. http://dl.acm.org/citation.cfm?id=3060832.3060886

[14] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. 2016. Tri-Party Deep Network Representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July2016*. 1895–1901. http://www.ijcai.org/Abstract/16/271

[15] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[16] Everett M Rogers and Dilip K Bhowmik. 1970. Homophily-heterophily: Relational concepts for communication research. *Public opinion quarterly* 34, 4 (1970), 523–538.

[17] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.

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

[19] T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning. (2012).

[20] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1225–1234.

[21] Yuchung J Wang and George Y Wong. 1987. Stochastic blockmodels for directed graphs. *J. Amer. Statist. Assoc.* 82, 397 (1987), 8–19.

[22] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina*. 2111–2117.