Accelerated Attributed Network Embedding

Xiao Huang* Jundong Li† Xia Hu*,‡

Abstract

Network embedding is to learn low-dimensional vector representations for nodes in a network. It has shown to be effective in a variety of tasks such as node classification and link prediction. While embedding algorithms on pure networks have been intensively studied, in many real-world applications, nodes are often accompanied with a rich set of attributes or features, aka attributed networks. It has been observed that network topological structure and node attributes are often strongly correlated with each other. Thus modeling and incorporating node attribute proximity into network embedding could be potentially helpful, though non-trivial, in learning better vector representations. Meanwhile, real-world networks often contain a large number of nodes and features, which put demands on the scalability of embedding algorithms. To bridge the gap, in this paper, we propose an accelerated attributed network embedding algorithm AANE, which enables the joint learning process to be done in a distributed manner by decomposing the complex modeling and optimization into many sub-problems. Experimental results on several real-world datasets demonstrate the effectiveness and efficiency of the proposed algorithm.

1 Introduction

Network analysis has become an effective tool in many real-world applications, such as targeted marketing and gene analysis [6, 38]. Identifying effective features is essential to these tasks, but it involves huge amounts of human efforts and massive engineering experimentations. As an alternative, network embedding [34, 40] maps the topological structure of each node into a low-dimensional vector representation, such that original network proximity can be well preserved. It has been shown that network embedding could benefit various tasks such as node classification [32, 42], link prediction [3, 31] and network clustering [25, 39].

While existing network embedding algorithms focus on pure networks, nodes in real-world networks are often associated with a rich set of features or attributes, known as attributed networks. Social science theories like homophily [23, 24] and social influence [38] suggest that node attributes are highly correlated with network structure, i.e., the formation of one depends on and also influences the other. Examples include the strong association between user posts and following relationships in microblogging, and high correlation of paper topics and citations in academic networks [41]. In various applications, such as sentiment analysis [13] and trust prediction [33], it has been shown that jointly exploiting the two information sources can enhance the learning performance. Motivated by the observations, we propose to study if node features can be potentially helpful in learning a better embedding representation.

Moreover, real-world networks are often large-scale with a sheer amount of nodes and high-dimensional features. For instance, there are over 65 million monthly active Twitter users in the United States as of 2016[1], and each user could post thousands of tweets. This puts more demands on the scalability of embedding algorithms. Attributed network embedding is challenging in three aspects as follows. (1) High computational time requirement might be a bottleneck that limits the usage of algorithms in practice. Some efforts have been devoted to leveraging network structure and attribute information for seeking a low-rank latent representation [19, 30]. They either require eigen-decomposition with $\mathcal{O}(n^3)$ time complexity in each iteration (n denotes the total number of nodes) [18] or employ gradient descent which usually has slow convergence rate [30, 43]. (2) Assessing node proximity in the joint space of network and attribute is challenging due to the bewildering combination of heterogeneous information. In addition, as the size of network scales up, node attribute proximity is often too large to be stored on a single machine, not to mention the operations on it. Hence, it would be appealing if the embedding algorithm is also scalable. (3) Both sources could be incomplete and noisy, which further exacerbates the embedding representation learning problem. Therefore, given the distinct characteristics of the data, existing methods cannot be directly applied to scalable attributed network embedding.

*Department of Computer Science & Engineering, Texas A&M University, College Station, USA. {xhuang, xiahu}@tamu.edu.

†Computer Science and Engineering, Arizona State University, Tempe, USA. jundongl@asu.edu.

‡Center for Remote Health Technologies and Systems, Texas A&M Engineering Experiment Station, College Station, USA.

[1] https://www.statista.com/statistics/274564/monthly-active-twitter-users-in-the-united-states

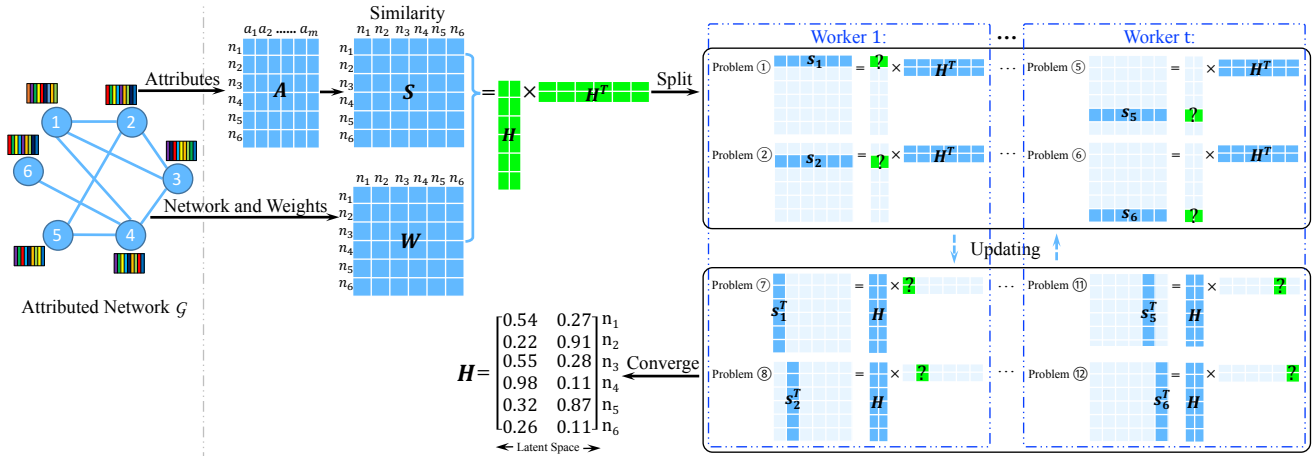

Figure 1: The basic idea of AANE is to represent nodes as continuous vectors, based on the decomposition of attribute affinity matrix and the penalty of embedding difference between connected nodes. The optimization is split into $2n$ sub-problems with low complexity, which can be solved separately by t workers.

To tackle the above challenges, in this paper, we study the problem of learning embedding representations on attributed networks efficiently. We aim at answering the following questions. (1) How to effectively model node proximity in the joint space composed of network structure and node attributes? (2) How to enable the vector representations learning process scalable and efficient? Through investigating these questions, we present a novel framework called Accelerated Attributed Network Embedding (AANE). We summarize the contributions of this paper as follows:

- Formally define the problem of attributed network embedding;
- Propose a scalable and efficient framework AANE, which can learn an effective unified embedding representation by incorporating node attribute proximity into network embedding;
- Present a distributed optimization algorithm to enable AANE to process each node efficiently by decomposing the complex modeling and optimization into many sub-problems with low complexity;
- Empirically validate the effectiveness and efficiency of AANE on three real-world datasets.

2 Problem Statement

Notations: In this paper, scalars are denoted by lowercase alphabets (e.g., n). Vectors are represented by boldface lowercase alphabets (e.g., $\mathbf{h}$). Matrices are denoted by boldface uppercase alphabets (e.g., $\mathbf{H}$). The i^{th} row of a matrix $\mathbf{H}$ is denoted by $\mathbf{h}_i$. The $(i,j)^{\text{th}}$ element of a matrix is denoted by h_{ij}. The transpose of $\mathbf{H}$ is represented as $\mathbf{H}^\top$. The dot product of two vectors is denoted by $\mathbf{a} \cdot \mathbf{b}$. The ℓ_2-norm of a vector is denoted by $\|\cdot\|_2$.The Frobenius norm of a matrix is represented as $\|\cdot\|_{\text{F}}$. The identity matrix is denoted by $\mathbf{I}$.

Notations	Definitions
$n = \|\mathcal{V}\|$	the number of nodes in the network
m	the number of node attribute categories
d	the dimension of embedding representation
$\mathbf{W} \in \mathbb{R}_+^{n\times n}$	the weighted adjacency matrix
$\mathbf{A} \in \mathbb{R}^{n\times m}$	the attribute information matrix
$\mathbf{S} \in \mathbb{R}^{n\times n}$	the node attribute affinity matrix
$\mathbf{H} \in \mathbb{R}^{n\times d}$	the final embedding representation

Table 1: Main symbols and definitions.

We list the main symbols in Table 1. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ be a network, where $\mathcal{V}$ is a set of n nodes, and $\mathcal{E}$ is the set of edges. Each edge $(i,j) \in \mathcal{E}$ is associated with a positive weight $w_{ij} \in \mathbf{W}$. In this paper, we focus on undirected networks. For directed ones, we follow the work of [21] to set w_{ij} and w_{ji} equal to their maximum value. A larger w_{ij} indicates higher similarity or stronger relationship between nodes i and j, and w_{ij} is defined as 0 if there is no edge. The set of adjacent nodes of node i is represented as $N(i)$. Let $\mathbf{A}$ be an $n\times m$ matrix that collects all of the node attributes, where m is the number of node attribute categories, and row $\mathbf{a}_i$ describes the attributes associated with node i.

Based on the terminologies explained above, we formally define the problem of attributed network embedding as follows: *Given a set of n nodes connected by a network $\mathcal{G}$ associated with edge weights $\mathbf{W}$ and node attributes $\mathbf{A}$, we aim to represent each node $i \in \mathcal{V}$ as a d-dimensional vector $\mathbf{h}_i$, such that this embedding representation $\mathbf{H}$ can preserve the node proximity both in topological structure $\mathcal{G}$ and node attributes $\mathbf{A}$. As a result, $\mathbf{H}$ could achieve better performance than original features in terms of advancing other learning tasks.*

3 Accelerated Embedding Framework - AANE

A desirable embedding method for real-world attributed networks satisfies three requirements as follows. First, it needs to be able to handle arbitrary types of edges (e.g., undirected or directed, unweighted or weighted) and node attributes (e.g., discrete or continuous values). Second, it needs to well preserve the node proximity both in network and attribute space. Third, it is important to be scalable since the number of nodes n and dimension of attributes m could be large. We develop an effective and efficient framework AANE that satisfies all of the requirements. In this section, we describe how AANE jointly models network structure and attribute proximity in an effective way. Figure 1 illustrates the basic idea. Given a network with $n = 6$ nodes, it first decomposes attribute affinity $\mathbf{S}$ into the product of $\mathbf{H}$ and $\mathbf{H}^\top$. Meanwhile, it imposes an edge-based penalty into this decomposition such that connected nodes are close to each other in $\mathbf{H}$, and the closeness is controlled by the edge weights in $\mathbf{W}$. The goal is to make more similar nodes in the network space to be closer in $\mathbf{H}$. To accelerate the optimization, a distributed algorithm is proposed to separate the original problem into $2n = 12$ sub-problems with low complexity. The first $n = 6$ sub-problems are designed to be independent of each other, and the same as the last six. So all sub-problems can be assigned to $t = 3$ workers. In the final output, node 1 and node 3 are represented by similar vectors $[0.54, 0.27]$ and $[0.55, 0.28]$, which indicates that they are similar to each other in the original network and feature joint space.

3.1 Network Structure Modeling

In order to render the joint embedding representation $\mathbf{H}$ well-posed, AANE should well preserve the node proximity both in network structure and node attribute space.

To preserve the node proximity in $\mathcal{G}$, the proposed framework is based on two hypotheses [11, 39]. First, a graph-based mapping is assumed to be smooth across edges, especially for the regions of high density [7]. This is in line with the cluster hypothesis [25]. Second, nodes with more similar topological structures or connected by higher weights are more likely to have similar vector representations. To achieve these goals, we propose the following loss function to minimize the embedding differences between connected nodes,

$$\mathcal{J}_{\mathcal{G}} = \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2, \tag{3.1}$$

where rows $\mathbf{h}_i$ and $\mathbf{h}_j$ are vector representations of node i and node j, and w_{ij} is the edge weight between the two. The key idea is that in order to minimize the penalty $w_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2$, a larger weight w_{ij} is more likely to enforce the difference of $\mathbf{h}_i$ and $\mathbf{h}_j$ to be smaller. We employ ℓ_2-norm as the difference metric to alleviate the negative impacts resulted from outliers and missing data. The proposed network structure modeling enjoys several nice properties as follows. First, it is in line with the fused lasso [37] and network lasso [9], which could induce sparsity in the differences of vector representations of similar nodes, and perform continuous edges selection similar to the continuous variable selection in lasso [36]. It is in compliance with the cluster hypothesis in graphs. Second, it is general to different types of networks. It could handle both undirected and directed networks with weighted or unweighted edges. Hence, it can be easily applied to many real-world applications.

3.2 Attribute Proximity Modeling

As indicated by social science theories like homophily and social influences [23, 38], attribute information of nodes is tightly hinged with network topological structure. Node proximity in network space should be consistent with the one in node attribute space. Thus, we investigate how to make the embedding representation $\mathbf{H}$ also well preserve the node attribute proximity. Motivated by the symmetric matrix factorization [17], we propose to approximate attribute affinity matrix $\mathbf{S}$ with the product of $\mathbf{H}$ and $\mathbf{H}^\top$. The basic idea is to enforce the dot product of vector representations $\mathbf{h}_i$ and $\mathbf{h}_j$ to be the same as corresponding attribute similarity $\mathbf{s}_{ij}$. Mathematically, the loss function is defined as

$$\mathcal{J}_A = \|\mathbf{S} - \mathbf{H}\mathbf{H}^\top\|_{\mathrm{F}}^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}(\mathbf{s}_{ij} - \mathbf{h}_i\mathbf{h}_j^\top)^2, \tag{3.2}$$

where affinity matrix $\mathbf{S}$ could be calculated by a representative similarity measure. Specifically, we use cosine similarity in this paper.

3.3 Joint Embedding Representation Learning

We have implemented two loss functions $\mathcal{J}_G$ and $\mathcal{J}_A$ to model the node proximity in network topological structure and node attributes. To make them complement each other towards a unified robust and informative space, we jointly model the two types of information in the following optimization problem,

$$\min_{\mathbf{H}} \ \mathcal{J} = \|\mathbf{S} - \mathbf{H}\mathbf{H}^\top\|_{\mathrm{F}}^2 + \lambda \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2. \tag{3.3}$$

Scalar λ serves as an overall parameter that defines a trade-off between the contributions of network and attribute information. It is also a regularization parameter that balances number of clusters [9, 22]. An intuitive explanation is that when λ is close to 0, the network topology cannot affect the final result $\mathbf{H}$, so each node

can be an isolated cluster. When λ is sufficiently large, the optimal solution will end up with same representations for all nodes, which forms a single cluster. This allows us to tune the number of clusters continuously. This number is not specified in network embedding, and tunability of λ is beneficial in this scenario.

3.4 Accelerated and Distributed Algorithm

The proposed objective function not only jointly models network proximity and node attribute affinity, but also has a specially designed structure that enables it to be optimized in an efficient and distributed manner, i.e., $\mathcal{J}$ is separable for $\mathbf{h}_i$ and can be reformulated as a bi-convex optimization problem. Next, we propose an efficient algorithm to solve it.

If we add a copy $\mathbf{Z} = \mathbf{H}$, then the first term in Eq. (3.3) can be rewritten as

(3.4)
$$\|\mathbf{S} - \mathbf{H}\mathbf{Z}^\top\|_{\mathrm{F}}^2 = \sum_{i=1}^{n} \|\mathbf{s}_i - \mathbf{h}_i\mathbf{Z}^\top\|_2^2 = \sum_{i=1}^{n} \|\mathbf{s}_i^\top - \mathbf{H}\mathbf{z}_i^\top\|_2^2.$$

We can further reformulate Eq. (3.3) into a linearly constrained problem as follows,

(3.5)
$$\begin{aligned} \min_{\mathbf{H}} \quad & \sum_{i=1}^{n} \|\mathbf{s}_i - \mathbf{h}_i\mathbf{Z}^\top\|_2^2 + \lambda \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{h}_i - \mathbf{z}_j\|_2, \\ \text{subject to} \quad & \mathbf{h}_i = \mathbf{z}_i,\ i = 1, \ldots, n. \end{aligned}$$

This indicates that $\mathcal{J}$ is separable for both $\mathbf{h}_i$ and $\mathbf{z}_i$. Since ℓ_2-norm is convex, it is easy to verify that Eq. (3.5) is also bi-convex, i.e., convex w.r.t. $\mathbf{h}_i$ when $\mathbf{z}_i$ is fixed and convex w.r.t. $\mathbf{z}_i$ when $\mathbf{h}_i$ is fixed. Thus, we are able to split the original complex embedding problem into $2n$ small convex optimization sub-problems.

It is infeasible to obtain closed-form solutions for these $2n$ sub-problems. Motivated by the distributed convex optimization technique - Alternating Direction Method of Multipliers (ADMM) [2, 9], we accelerate the optimization by converting it into $2n$ updating steps and one matrix updating step. The proposed solution enjoys several nice properties. First, it enables the n updating steps of $\mathbf{h}_i$ (or $\mathbf{z}_i$) independent of each other. Thus in each iteration, the global coordination can assign tasks to workers and collect the solutions from them without a fixed order. Second, all updating steps have low complexity. Third, it converges to a modest solution fast [2]. We now introduce the details.

To solve the problem, we first formulate the augmented Lagrangian [10] of Eq. (3.5) as follows,

(3.6)
$$\begin{aligned} \mathcal{L} = & \sum_{i=1}^{n} \|\mathbf{s}_i - \mathbf{h}_i\mathbf{Z}^\top\|_2^2 + \lambda \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{h}_i - \mathbf{z}_j\|_2 \\ & + \frac{\rho}{2}\sum_{i=1}^{n}(\|\mathbf{h}_i - \mathbf{z}_i + \mathbf{u}_i\|_2^2 - \|\mathbf{u}_i\|_2^2), \end{aligned}$$

where rows $\mathbf{u}_1, \ldots, \mathbf{u}_n \in \mathbb{R}^d$ are the scaled dual variables, and $\rho > 0$ is the penalty parameter. The minimizer of Eq. (3.5) is then converted to the saddle point of $\mathcal{L}$, which can be found by finding optimal $\mathbf{H}$, $\mathbf{Z}$, and $\mathbf{U}$ iteratively. The corresponding optimization problems in terms of each node i at iteration $k+1$ are written as

(3.7)
$$\begin{aligned} \mathbf{h}_i^{k+1} = \ & \operatorname*{argmin}_{\mathbf{h}_i} (\|\mathbf{s}_i - \mathbf{h}_i{\mathbf{Z}^k}^\top\|_2^2 + \lambda \sum_{j\in N(i)} w_{ij}\|\mathbf{h}_i - \mathbf{z}_j^k\|_2 \\ & + \frac{\rho}{2}\|\mathbf{h}_i - \mathbf{z}_i^k + \mathbf{u}_i^k\|_2^2), \end{aligned}$$

(3.8)
$$\begin{aligned} \mathbf{z}_i^{k+1} = \ & \operatorname*{argmin}_{\mathbf{z}_i} (\|\mathbf{s}_i^\top - \mathbf{H}^{k+1}\mathbf{z}_i^\top\|_2^2 + \lambda \sum_{j\in N(i)} w_{ji}\|\mathbf{z}_i - \mathbf{h}_j^{k+1}\|_2 \\ & + \frac{\rho}{2}\|\mathbf{z}_i - \mathbf{h}_i^{k+1} - \mathbf{u}_i^k\|_2^2), \text{ and} \end{aligned}$$

$$\mathbf{U}^{k+1} = \mathbf{U}^k + (\mathbf{H}^{k+1} - \mathbf{Z}^{k+1}). \tag{3.9}$$

Here we need to find all of the $\mathbf{h}_i^{k+1}$, for $i = 1, \ldots, n$, before calculating $\mathbf{z}_i^{k+1}$. However, the order of solving $\mathbf{h}_i^{k+1}$ is not fixed, since they are independent to each other. If the machine capacity is limited, $\mathbf{s}_i$ can also be calculated separately via equation $\mathbf{s}_i = (\mathbf{a}_i\mathbf{A}^\top) \cdot (\frac{1}{q_i\mathbf{q}})$, where $\mathbf{q}$ is the dot product of each node attribute vector, i.e., $\mathbf{q} = [\sqrt{\mathbf{a}_1 \cdot \mathbf{a}_1}, \ldots, \sqrt{\mathbf{a}_n \cdot \mathbf{a}_n}]$. By taking the derivative of Eq. (3.7) w.r.t. $\mathbf{h}_i$, and setting it to zero, we get an updating rule to approach optimal $\mathbf{h}_i$ as follows,

$$\mathbf{h}_i^{k+1} = \frac{2\mathbf{s}_i\mathbf{Z}^k + \lambda\sum_{j\in N(i)} \frac{w_{ij}\mathbf{z}_j^k}{\|\mathbf{h}_i^k - \mathbf{z}_j^k\|_2} + \rho(\mathbf{z}_i^k - \mathbf{u}_i^k)}{2{\mathbf{Z}^k}^\top\mathbf{Z}^k + (\lambda\sum_{j\in N(i)} \frac{w_{ij}}{\|\mathbf{h}_i^k - \mathbf{z}_j^k\|_2} + \rho)\mathbf{I}}. \tag{3.10}$$

Here we use $\mathbf{h}_i^k$ to estimate the distance $\|\mathbf{h}_i^k - \mathbf{z}_j^k\|_2$. The monotonically decreasing property of this updating rule is proved by the work of Nie et al. [26]. Since the problem in Eq. (3.7) is convex, $\mathbf{h}_i$ should be optimum if and only if $\mathbf{h}_i^k = \mathbf{h}_i^{k+1}$. We stop the updating as they are closed enough. We obtain a similar rule for $\mathbf{z}_i$,

(3.11)
$$\mathbf{z}_i^{k+1} = \frac{2\mathbf{s}_i\mathbf{H}^{k+1} + \lambda\sum_{j\in N(i)} \frac{w_{ij}\mathbf{h}_j^{k+1}}{\|\mathbf{z}_i^k - \mathbf{h}_j^{k+1}\|_2} + \rho(\mathbf{h}_i^{k+1} + \mathbf{u}_i^k)}{2{\mathbf{H}^{k+1}}^\top\mathbf{H}^{k+1} + (\lambda\sum_{j\in N(i)} \frac{w_{ij}}{\|\mathbf{z}_i^k - \mathbf{h}_j^{k+1}\|_2} + \rho)\mathbf{I}}.$$

The distributed algorithm for optimizing the problem in Eq. (3.3) is described in Algorithm 1. Since Eq. (3.5) is bi-convex, it is guaranteed to converge to a local optimal point [2]. To have an appropriate initialization, we set $\mathbf{H}$ as the left singular vectors of $\mathbf{A}_0$ in the first iteration. $\mathbf{A}_0$ is a matrix that collects the first $2d$ columns of $\mathbf{A}$. The optimization is split into $2n$ sub-problems, and we solve them iteratively. In each iteration, the n updating steps of $\mathbf{h}_i$ (or $\mathbf{z}_i$) can be assigned to t workers in a distributed way as illustrated in Figure 1. The termination criterion is that primal residual $\mathbf{r}$ and dual residual $\mathbf{s}$ should be sufficiently small.

Algorithm 1: Accelerated Attributed Network Embedding

Input: $\mathbf{W}$, $\mathbf{A}$, d, ϵ.
Output: Embedding representation $\mathbf{H}$.

$\mathbf{A}_0 \leftarrow$ First $2d$ columns of $\mathbf{A}$;
Set $k = 0$, and $\mathbf{H}^k \leftarrow$ Left singular vectors of $\mathbf{A}_0$;
Set $\mathbf{U}^k = \mathbf{0}$, $\mathbf{Z}^k = \mathbf{H}^k$, and calculate $\mathbf{S}$;
repeat
  Calculate ${\mathbf{Z}^k}^\top \mathbf{Z}^k$;
  `/* Assign these` n `tasks to` t `workers: */`
  for $i = 1 : n$ **do**
    Update $\mathbf{h}_i^{k+1}$ based on Eq. (3.10);
  Calculate ${\mathbf{H}^{k+1}}^\top \mathbf{H}^{k+1}$;
  `/* Assign these` n `tasks to` t `workers: */`
  for $i = 1 : n$ **do**
    Update $\mathbf{z}_i^{k+1}$ based on Eq. (3.11);
  $\mathbf{U}^{k+1} \leftarrow \mathbf{U}^k + (\mathbf{H}^{k+1} - \mathbf{Z}^{k+1})$;
  $k \leftarrow k + 1$;
until $\|\mathbf{r}^k\|_2 \le \epsilon^{pri}$ *and* $\|\mathbf{s}^k\|_2 \le \epsilon^{dual}$;
return $\mathbf{H}$.

3.5 Complexity Analysis As it is typical for ADMM [2], Algorithm 1 tends to converge to a modest accuracy in a few iterations. In the initialization, the time complexity for calculating singular vectors of an n by d matrix $\mathbf{A}_0$ is $\mathcal{O}(d^2 n)$, and we denote the number of operations required to obtain the attribute affinity matrix $\mathbf{S}$ as $\mathcal{T}_{\mathbf{S}}$. In each subtask, the updating time for $\mathbf{h}_i$ should be $\mathcal{O}(d^3 + dn + d|N(i)|)$ since we only need to compute ${\mathbf{Z}^k}^\top \mathbf{Z}^k$ once per iteration. Since $d \ll n$, it can be reduced to $\mathcal{O}(n)$. It is easy to check that the space complexity of each subtask is also $\mathcal{O}(n)$. Therefore, the total time complexity of AANE should be $\mathcal{O}(n\mathcal{N}_{\mathbf{A}} + \frac{n^2}{t})$, where $\mathcal{N}_{\mathbf{A}}$ is the number of nonzero in $\mathbf{A}$.

4 Experiments

In this section, we empirically evaluate the effectiveness and efficiency of the proposed framework AANE. We aim at answering three questions as follows. (1) What are the impacts of node attributes on network embedding? (2) How effective is the embedding representation learned by AANE compared with other learning methods on real-world attributed networks? (3) How efficient is AANE compared with the state-of-the-art methods? We first introduce the datasets used in the experiments.

4.1 Datasets Three real-world attributed networks BlogCatalog, Flickr and Yelp are used in this work. All of them are publicly available and the first two also have been used in previous work [14, 20]. Statistics of the datasets are summarized in Table 2.

Dataset	BlogCatalog	Flickr	Yelp
Nodes ($\lvert\mathcal{V}\rvert$)	5,196	7,564	249,012
Edges ($\lvert\mathcal{E}\rvert$)	171,743	239,365	1,779,803
Attribute (m)	8,189	12,047	20,000
Label (ℓ)	6	9	11

Table 2: Detailed information of the three datasets.

BlogCatalog is a blogger community, where users interact with each other and form a network. Users are allowed to generate keywords as a short description of their blogs. These keywords are severed as node attributes. Users also register their blogs under predefined categories, and we set them as labels. The user with no follower or predefined category has been removed.
Flickr is an online community that people can share photos. Photographers can follow each other and form a network. We employ the tags specified on their images as attribute information. We set the groups that photographers joined as labels.
Yelp[2] is a social networking service, where crowdsourced reviews about local businesses are shared. We employ users' friend relationships to form the network. Bag-of-words model is used to represent users' reviews as attribute information. All local businesses are separated into eleven primary categories, such as Active Life, Fast Food and Services. A user may have reviewed one or several businesses. We use the categories of these businesses as the user's labels.

4.2 Baseline Methods AANE is compared with two categories of baseline methods. To evaluate the contribution of incorporating node attributes, two scalable network embedding methods and a classical method for modeling attributes are used for comparison, i.e., DeepWalk, LINE and PCA. To investigate the effectiveness and efficiency of AANE, we compare it with two state-of-the-art attributed network learning methods, i.e., LCMF and MultiSpec. The detailed descriptions of these methods are listed as follows.

- *DeepWalk* [29]: It involves language modeling techniques to analyze the truncated random walks on a graph. It embeds the walking tracks as sentences, and each vertex corresponds to a unique word.
- *LINE* [34]: It embeds the network into a latent space by sampling both one-hop and two-hop neighbors of each node. It is one of the state-of-the-art scalable network embedding methods.
- *PCA* [16]: It is a classical dimensionality reduction technique. It takes the top d principal components of attribute matrix $\mathbf{A}$ as the learned representation.

[2] `https://www.yelp.com/dataset_challenge/dataset`

		BlogCatalog				Flickr				Yelp 1			
Training Percentage		10%	25%	50%	100%	10%	25%	50%	100%	10%	25%	50%	100%
# nodes for embedding		1,455	2,079	3,118	5,196	2,118	3,026	4,538	7,564	13,945	19,921	29,881	49,802
Macro-average	DeepWalk	0.489	0.548	0.606	0.665	0.310	0.371	0.462	0.530	0.139	0.159	0.215	0.275
	LINE	0.425	0.542	0.620	0.681	0.256	0.331	0.418	0.512	0.165	0.173	0.193	0.227
	PCA	0.691	0.780	0.821	0.855	0.510	0.612	0.671	0.696	0.591	0.599	0.605	N.A.
	LCMF	0.776	0.847	0.886	0.900	0.585	0.683	0.729	0.751	0.589	0.605	0.612	N.A.
	MultiSpec	0.677	0.787	0.847	0.895	0.589	0.722	0.802	0.859	0.461	0.460	N.A.	N.A.
	AANE	**0.836**	**0.875**	**0.912**	**0.930**	**0.743**	**0.814**	**0.852**	**0.883**	**0.630**	**0.645**	**0.656**	**0.663**
Micro-average	DeepWalk	0.491	0.551	0.611	0.672	0.312	0.373	0.465	0.535	0.302	0.310	0.318	0.350
	LINE	0.433	0.545	0.624	0.684	0.259	0.332	0.421	0.516	0.230	0.243	0.264	0.294
	PCA	0.695	0.782	0.823	0.857	0.508	0.606	0.666	0.692	0.667	0.674	0.681	N.A.
	LCMF	0.778	0.849	0.888	0.902	0.576	0.676	0.725	0.749	0.668	0.680	0.686	N.A.
	MultiSpec	0.678	0.788	0.849	0.896	0.589	0.720	0.800	0.859	0.553	0.571	N.A.	N.A.
	AANE	**0.841**	**0.878**	**0.913**	**0.932**	**0.740**	**0.811**	**0.854**	**0.885**	**0.679**	**0.694**	**0.703**	**0.711**

Table 3: Classification performance of different methods on different datasets with $d = 100$.

Training Percentage		10%	25%	50%	100%
# nodes for embedding		69,723	99,605	149,407	249,012
Macro-average	DeepWalk	0.239	0.254	0.266	0.260
	LINE	0.216	0.236	0.259	0.279
	AANE	**0.649**	**0.659**	**0.660**	**0.665**
Micro-average	DeepWalk	0.324	0.345	0.366	0.368
	LINE	0.295	0.313	0.336	0.354
	AANE	**0.698**	**0.709**	**0.711**	**0.714**

Table 4: Performance of different methods on Yelp with $d = 100$. PCA, LCMF and MultiSpec are infeasible.

- *LCMF* [43]: It learns a low-dimensional representation from linkage and content information by carrying out a joint matrix factorization on them.

- *MultiSpec* [18]: It treats network structure and node attributes as two views, and embeds them jointly by co-regularizing spectral clustering hypotheses across two views.

4.3 Experimental Setup Following the widely adopted way of validating network embedding [29, 34], we evaluate AANE and baseline methods on node classification [32, 43]. The task is to predict which category or categories a new node belongs to based on its vector representation and learned classifier. We now introduce the experimental settings in detail.

We employ 5-fold cross validation, i.e., randomly separate the entire nodes into a training group ($\mathbf{W}_{\text{train}}$, $\mathbf{A}_{\text{train}}$, $\mathbf{Y}_{\text{train}}$) and a test group ($\mathbf{W}_{\text{test}}$, $\mathbf{A}_{\text{test}}$, $\mathbf{Y}_{\text{test}}$), where $\mathbf{Y}$ denotes the labels. All edges between training group and test group have been removed. To investigate the performance of a method, we apply it to both groups and learn vector representations $\mathbf{H}$ for all nodes. Since there are multiple label categories, we build a binary SVM classifier for each category based on $\mathbf{H}_{\text{train}}$ and $\mathbf{Y}_{\text{train}}$. At last, we perform the classification based on $\mathbf{H}_{\text{test}}$ and the learned SVM classifiers. Labels in $\mathbf{Y}_{\text{test}}$ serve as ground truth.

The classification performance is measured via two commonly used evaluation criteria, macro-average and micro-average [15]. F-measure is a widely used metric for binary classification. Macro-average is defined as an arithmetic average of F-measure of all ℓ label categories, and Micro-average is the harmonic mean of average precision and average recall.

We follow the suggestions of the original papers to set the parameters of baseline methods. The embedding dimension d is set to be 100. All experimental results are the arithmetic average of 10 test runs. We ran the experiments on a Dell OptiPlex 9030 i7-16GB desktop.

4.4 Effectiveness Evaluation To investigate the impacts of attribute information and how much it can improve the embedding representation, we compare AANE with all baseline methods. We also vary the nodes used for training as {10%, 25%, 50%, 100%} of the entire training group to evaluate the effect brought by different sizes of training data. The classification performance w.r.t. training percentage is presented in Table 3. Since neither of existing attributed network learning methods is practicable on the Yelp dataset, we randomly select 20% of it and set as a new dataset, i.e., Yelp 1, such that we are able to compare the effectiveness of different method on it. Results of AANE on original Yelp dataset are shown in Table 4.

From the results, we have three observations as follows. First, by taking advantage of node attributes, LCMF, MultiSpec, and AANE all achieve significantly better performance than network embedding methods on all datasets. For example, in BlogCatalog results, we can discover that incorporating attribute information allows AANE to achieve 38.7% gain over DeepWalk and 36.3% gain over LINE in terms of Micro-average score. Second, with the proposed formulation, AANE consistently outperforms LCMF and MultiSpec. For instance, on Flickr, AANE achieves 18.2% of improvements than LCMF, which can be explained by the fact that latent features learned by decomposing network matrix and attribute matrix are heterogeneous, and it is challenging to combine them. Third, LCMF and MultiSpec are infeasible on the datasets Yelp. For instance, MultiSpec

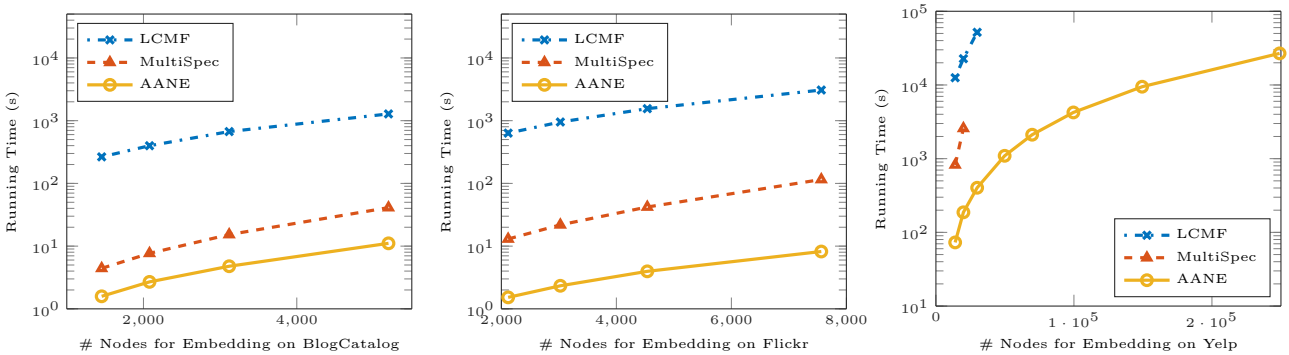

Figure 2: Running time of LCMF, MultiSpec and AANE w.r.t. the number of input nodes on the three datasets.

becomes impracticable as the number of nodes increases to 29,881, due to the high computational and memory requirement of eigen-decomposition.

We further vary the percentage of the training group that is used for training. As shown in Tables 3 and 4, similar observations are made. LCMF, MultiSpec and AANE outperform the two pure network embedding methods. AANE consistently achieves better performance than all baseline methods. We also find that MultiSpec may perform worse than PCA. For instance, on Yelp 1, PCA achieves a gain of 20.6% over MultiSpec when training percentage is 10%. One-tailed t-test results show that AANE is statistically significantly better (with p-value $\ll 0.01$) than all other methods.

4.5 Efficiency Evaluation To study the efficiency of AANE, we compare it with the two attributed network learning methods. The computation time in logarithmic scale as a function of the number of input nodes on three datasets are shown in Figure 2. The blue, red, and yellow dash-dot curves show the performance of LCMF, MultiSpec, and AANE respectively.

Empirical results show that a near optimal solution of $\mathcal{J}$ is enough to guarantee $\mathbf{H}$ to be an informative embedding representation. Therefore, in practice, only a few iterations is required for AANE. From the results in Figure 2, we observe that our method AANE takes much less running time than LCMF and MultiSpec consistently. As the number of input nodes increases, the performance difference also raises up. The third figure shows that LCMF and MultiSpec have larger growth rates than AANE. They become infeasible when the number of nodes is greater than 49,802 and 29,881 respectively. Furthermore, while the three methods are running in single-thread for a fair comparison, AANE could be implemented in multi-thread as illustrated in Figure 1. This strategy could further reduce the computation time of AANE. In summary, all these observations demonstrate the efficiency and scalability of our proposed framework.

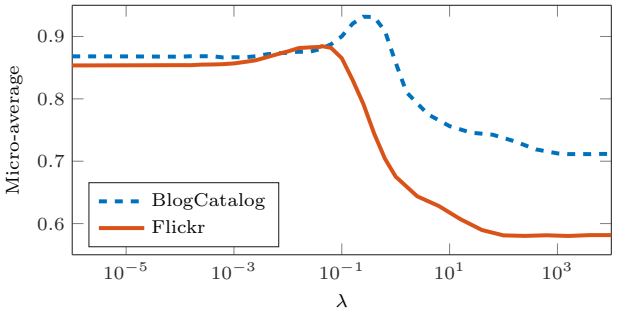

Figure 3: Impact of regularization parameter λ on the proposed framework AANE.

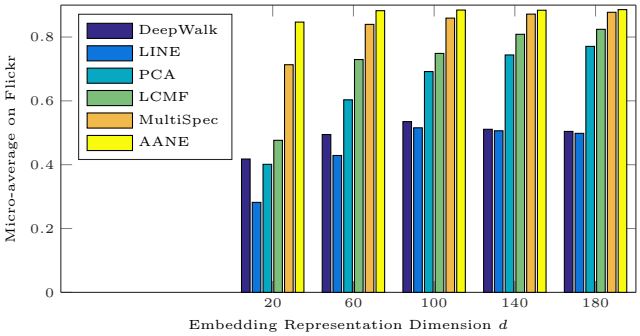

Figure 4: Classification performance on Flickr dataset w.r.t. the embedding dimension d.

4.6 Parameter Analysis We study the impacts of two important parameters, λ and d, in this subsection. As discussed in Section 3.3, λ in AANE balances the contributions of network and node attributes. To investigate the impact of λ, we vary it from 10^{-6} to 10^{3}. Figure 3 shows the classification performance in terms of Micro-average as a function of λ. When $\lambda = 0$, network information has no impact as if all nodes are isolated. As λ increases, AANE starts to cluster nodes according to the topological structure, so the performance keeps improving. As shown in Figure 3, performance on BlogCatalog and Flickr achieves optimal when λ is close to 0.1. The performance decreases when λ is too large, since large λ tends to drive all nodes to have the same vector representation.

To study the impact of embedding dimension, we vary it from 20 to 180 on all datasets. The classification performance in terms of Micro-average w.r.t. d on Flickr is shown in Figure 4. We omit the results on BlogCatalog and Yelp since they are similar. From the results, we discover that observations made above hold undeviatingly as d varies. DeepWalk and LINE are always inferior to AANE and the two attributed network learning methods. AANE consistently outperforms all baseline methods. By increasing d, the performance of all methods first increases and then keeps stable. This shows that the low-dimensional representation performs well in capturing most of the meaningful information.

5 Related Work

Network embedding has become an efficient tool to deal with large-scale networks. Efforts have been devoted from various aspects [8, 29, 34]. Tang and Liu [35] presented an edge-centric clustering scheme to facilitate the learning efficiency and alleviate the memory demand. Ahmed et al. [1] advanced a distributed matrix factorization algorithm to decompose large-scale graphs based on stochastic gradient descent. Tang et al. [34] improved the efficiency of stochastic gradient descent by unfolding a weighted edge into multiple binary edges. Ou et al. [27] designed a Jacobi-Davidson type algorithm to approximate and accelerate the singular value decomposition in the high-order proximity embedding. Wang et al. [40] involved the deep structure to embedding both first and second order proximities of nodes efficiently.

Research has been done to analyze attributed networks in various domains. It becomes increasingly promising to advance the learning performance by jointly exploiting geometrical structures and node attributes [13, 21, 33]. Tsur and Rappoport [38] improved the prediction of the spread of ideas by taking advantage of both content and topological features. Due to the complexity of attributed networks, nodes' properties and dependencies cannot be fully explained via these models. Several algorithms [20, 43] have been proposed to exploit the potential of learning a joint low-rank latent representation. Qi et al. [30] modeled the content information based on pairwise similarity and jointly mapped it along with context link into a semantic latent space by learning structural correlations between semantic concepts. Le and Lauw [19] advocated a probabilistic-based framework for the document network by finding a joint low-dimensional representation for network connectivity and textual content. Chang et al. [4] designed a deep learning method to map the rich linkage and content information into a latent space while capturing the similarities among cross-modal data. Efforts have also been devoted to learning latent representations of networked instances by performing unsupervised feature selection [20, 21]. Xiao et al. [14] explored the potential of incorporating labels into the attributed network embedding. Attributed network analysis is different from multi-view learning [18]. The network structure is more than one angle of view. Its underlying properties are complex, including the connectivity, transitivity [27], first and higher order proximities [40], etc.

Network lasso was formally defined by Hallac et al. [9] as a simultaneous clustering and optimization problem. The key idea is to utilize the ℓ_2-norm distances of adjacent nodes as penalties and enforce nodes in the same cluster to have same variables. The pioneer work of network lasso can also be traced back to the fused lasso [12, 37] and convex clustering [28] problems. Both of them can be viewed as special cases of network lasso and have been well explored. Lindsten et al. [22] demonstrated the equivalence between convex clustering and a convexification of k-means clustering. They utilized the off-the-shelf convex programming software CVX to handle the regularization. This solver is powerful in solving general convex problems but quite inefficient. Thus, Hocking et al. [11] introduced several efficient algorithms for convex clustering with three most commonly used norms, ℓ_1, ℓ_2 and ℓ_∞. Most recently, Chi and Lange [5] exploited the possibility of finding regularization paths of arbitrary norms by using ADMM and AMA (alternating minimization algorithm). However, all these methods can only deal with thousands of nodes, and convex clustering often requires to include distances between every pair of nodes as penalties.

6 Conclusion and Future Work

In this paper, we propose an efficient embedding framework AANE to deal with attributed networks at scale. The proposed framework can effectively model and incorporate node attribute proximity into network embedding in a distributed way. Specifically, we learn a low-dimensional representation based on the decomposition of attribute affinity and the embedding difference between connected nodes. A distributed optimization algorithm is developed to decompose the complex problem into many sub-problems with low complexity, which can be solved by sub-workers in parallel. Experiments on real-world attributed networks demonstrate its effectiveness and efficiency. Our future work in this area will focus on investigating the following questions. (1) How to embed large-scale attributed networks with dynamic topological structure and node attributes? (2) How can we extend our method to a semi-supervised framework? Since in many real-world attributed networks, partial labels are often available.

References

[1] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola. Distributed large-scale natural graph factorization. *WWW*, 2013.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 2011.

[3] B. Cao, N. N. Liu, and Q. Yang. Transfer learning for collective link prediction in multiple heterogenous domains. *ICML*, 2010.

[4] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. *KDD*, 2015.

[5] E. C. Chi and K. Lange. Splitting methods for convex clustering. *JCGS*, 2015.

[6] H.-Y. Chuang, E. Lee, Y.-T. Liu, D. Lee, and T. Ideker. Network-based classification of breast cancer metastasis. *Molecular systems biology*, 2007.

[7] A. El Alaoui, X. Cheng, A. Ramdas, M. J. Wainwright, and M. I. Jordan. Asymptotic behavior of ℓ_p-based Laplacian regularization in semi-supervised learning. *JMLR*, 2016.

[8] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. *KDD*, 2016.

[9] D. Hallac, J. Leskovec, and S. Boyd. Network lasso: Clustering and optimization in large graphs. *KDD*, 2015.

[10] M. R. Hestenes. Multiplier and gradient methods. *J. Optim. Theor. Appl.*, 1969.

[11] T. D. Hocking, A. Joulin, F. Bach, and J.-P. Vert. Clusterpath an algorithm for clustering using convex fusion penalties. *ICML*, 2011.

[12] H. Hoefling. A path algorithm for the fused lasso signal approximator. *JCGS*, 2010.

[13] X. Hu, L. Tang, J. Tang, and H. Liu. Exploiting social relations for sentiment analysis in microblogging. *WSDM*, 2013.

[14] X. Huang, J. Li, and X. Hu. Label informed attributed network embedding. *WSDM*, 2017.

[15] L. Jian, J. Li, K. Shu, and H. Liu. Multi-label informed feature selection. *IJCAI*, 2016.

[16] I. Joliffe. Principal component analysis. *Springer Verlag*, 1986.

[17] D. Kuang, C. Ding, and H. Park. Symmetric nonnegative matrix factorization for graph clustering. *SDM*, 2012.

[18] A. Kumar, P. Rai, and H. Daume. Co-regularized multi-view spectral clustering. *NIPS*, 2011.

[19] T. M. V. Le and H. W. Lauw. Probabilistic latent document network embedding. *ICDM*, 2014.

[20] J. Li, X. Hu, J. Tang, and H. Liu. Unsupervised streaming feature selection in social media. *CIKM*, 2015.

[21] J. Li, X. Hu, L. Wu, and H. Liu. Robust unsupervised feature selection on networked data. *SDM*, 2016.

[22] F. Lindsten, H. Ohlsson, and L. Ljung. Just relax and come clustering!: A convexification of k-means clustering. *LiU E-Press*, 2011.

[23] P. V. Marsden. Homogeneity in confiding relations. *Social Networks*, 1988.

[24] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 2001.

[25] H. Narayanan, M. Belkin, and P. Niyogi. On the relation between low density separation, spectral clustering and graph cuts. *NIPS*, 2006.

[26] F. Nie, H. Huang, X. Cai, and C. Ding. Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization. *NIPS*, 2010.

[27] M. Ou, P. Cui, J. Pei, and W. Zhu. Asymmetric transitivity preserving graph embedding. *KDD*, 2016.

[28] K. Pelckmans, J. De Brabanter, J. Suykens, and B. De Moor. Convex clustering shrinkage. *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*, 2005.

[29] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online Learning of Social Representations. *KDD*, 2014.

[30] G. Qi, C. Aggarwal, Q. Tian, H. Ji, and T. Huang. Exploring context and content links in social media: A latent space method. *TPAMI*, 2012.

[31] A. P. Singh and G. J. Gordon. Relational Learning via Collective Matrix Factorization. *KDD*, 2008.

[32] J. Tang, C. Aggarwal, and H. Liu. Node classification in signed social networks. *SDM*, 2016.

[33] J. Tang, H. Gao, X. Hu, and H. Liu. Exploiting homophily effect for trust prediction. In *WSDM*, 2013.

[34] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. LINE: Large scale Information Network Embedding. *WWW*, 2015.

[35] L. Tang and H. Liu. Scalable learning of collective behavior based on sparse social dimensions. *CIKM*, 2009.

[36] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc.*, 1996.

[37] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *J. R. Stat. Soc.*, 2005.

[38] O. Tsur and A. Rappoport. What's in a hashtag? content based prediction of the spread of ideas in microblogging communities. *WSDM*, 2012.

[39] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

[40] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. *KDD*, 2016.

[41] G. Zhang, Y. Ding, and S. Milojević. Citation content analysis (CCA): A framework for syntactic and semantic analysis of citation content. *JASIST*, 2013.

[42] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. *NIPS*, 2006.

[43] S. Zhu, K. Yu, Y. Chi, and Y. Gong. Combining content and link for classification using matrix factorization. *SIGIR*, 2007.