Arbitrary-Order Proximity Preserved Network Embedding

Ziwei Zhang*
Tsinghua University
zw-zhang16@mails.tsinghua.edu.cn

Peng Cui
Tsinghua University
cuip@tsinghua.edu.cn

Xiao Wang
Tsinghua University
wangxiao007@mail.tsinghua.edu.cn

Jian Pei
JD.com and Simon Fraser University
jpei@cs.sfu.ca

Xuanrong Yao
Tsinghua University
yaoxr17@mails.tsinghua.edu.cn

Wenwu Zhu
Tsinghua University
wwzhu@tsinghua.edu.cn

ABSTRACT

Network embedding has received increasing research attention in recent years. The existing methods show that the high-order proximity plays a key role in capturing the underlying structure of the network. However, two fundamental problems in preserving the high-order proximity remain unsolved. First, all the existing methods can only preserve fixed-order proximities, despite that proximities of different orders are often desired for distinct networks and target applications. Second, given a certain order proximity, the existing methods cannot guarantee accuracy and efficiency simultaneously. To address these challenges, we propose AROPE (arbitrary-order proximity preserved embedding), a novel network embedding method based on SVD framework. We theoretically prove the eigen-decomposition reweighting theorem, revealing the intrinsic relationship between proximities of different orders. With this theorem, we propose a scalable eigen-decomposition solution to derive the embedding vectors and shift them between proximities of arbitrary orders. Theoretical analysis is provided to guarantee that i) our method has a low marginal cost in shifting the embedding vectors across different orders, ii) given a certain order, our method can get the global optimal solutions, and iii) the overall time complexity of our method is linear with respect to network size. Extensive experimental results on several large-scale networks demonstrate that our proposed method greatly and consistently outperforms the baselines in various tasks including network reconstruction, link prediction and node classification.

KEYWORDS

Network Embedding, Arbitrary-Order Proximity, Network Representation Learning

ACM Reference Format:
Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-Order Proximity Preserved Network Embedding. In *KDD 2018: 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219969

*Beijing National Research Center for Information Science and Technology (BNRist)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD 2018, August 19–23, 2018, London, United Kingdom
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219969

1 INTRODUCTION

Network embedding, which has attracted increasing research attention, represents nodes by low-dimensional vectors while preserving the inherent properties and structures of the network. In this way, feature-based machine learning algorithms can be easily applied for network analysis. Prior works have demonstrated that, besides the pairwise edges, the high-order proximities between nodes are of tremendous importance in capturing the underlying structure of the network [3, 8, 13, 21, 23] and thus can provide valuable information for learning the embedding vectors. Therefore, a bunch of methods have been proposed to preserve the high-order proximities in network embedding.

Despite their success, two fundamental problems remain unsolved. First, all the existing methods can only preserve fixed-order proximities. However, embeddings with a certain order proximity do not necessarily perform best on all networks and target applications [3, 24]. For example, in classification tasks, classes with different granularities often require proximities of different orders, i.e. coarser-grained classes require higher-order proximities and finer-grained classes require lower-order proximities [24]. We also empirically verify the effect of the orders, as shown in Figure 5. To incorporate proximities of different orders, the existing methods have to rerun multiple times and compute multiple embeddings for each node separately. For example, DeepWalk [23] sets the order by a hyper-parameter, i.e. the pre-defined window size, and has to rerun the algorithm from scratch each time for a different hyper-parameter. Considering a more general setting where proximities of different orders need to be preserved with different weights, such as combining atomic proximities to form a more delicate proximity measure as in [14], the existing methods confront even more difficulties due to their immutable incorporating of the high-order proximities.

Second, even given a certain order proximity, it is still an open issue for the existing methods to guarantee accuracy and efficiency simultaneously. For example, a series of works [13, 23] adopt random walks to explore the high-order proximities and use Stochastic Gradient Descend (SGD) for optimization. Since the objective function is not convex and the number of iterations for the optimization is usually set small to ensure efficiency, these methods cannot guarantee global optimal solutions [4]. On the other hand, network embedding methods based on matrix factorization [3, 35] or deep learning [34] are known to have efficiency issues. How to guarantee

accuracy and efficiency simultaneously in the case of preserving arbitrary-order proximities is even more challenging.

In this paper, we propose AROPE[1] (arbitrary-order proximity preserved embedding), a novel network embedding method based on SVD framework. We theoretically prove the eigen-decomposition reweighting theorem, which reveals that the intrinsic relationship between proximities of different orders is reweighting and reordering the dimensions. With this theorem, we propose a scalable solution to derive the embedding vectors. Meanwhile, we can shift the embedding vectors between proximities of arbitrary orders, i.e. efficiently getting embedding vectors of any proximity from some base embedding vectors, even in the general setting where different orders of proximities are arbitrarily weighted. Theoretical analysis is provided to guarantee that i) our method has a low marginal cost in shifting the embedding vectors across different orders and weights, ii) given a certain order, our method can get the global optimal solutions, and iii) the overall time complexity of our method is linear with respect to network size.

Extensive experiments on several large-scale networks with millions of nodes and edges are conducted. The experimental results demonstrate that our proposed method greatly and consistently outperforms the state-of-the-art network embedding methods in several applications of network embedding, including network reconstruction, link prediction and node classification.

To summarize, the contributions of our paper are as follows:

- We propose AROPE, a novel network embedding method that supports shifts across arbitrary orders with a low marginal cost.
- We prove the eigen-decomposition reweighting theorem to reveal the intrinsic relationship between proximities of different orders in SVD framework. Theoretical analysis shows that our method can get the global optimal solutions while having a linear time complexity.
- Extensive experimental results demonstrate the efficacy of our method by improving the precision of network reconstruction and link prediction by more than 100% on two large networks.

The rest of this paper is organized as follows. In Section 2, we briefly review related works. We formulate our problem in Section 3. In Section 4, we introduce our proposed method. Experimental results are reported in Section 5. Finally, we summarize in Section 6.

2 RELATED WORK

Network embedding has recently become a paradigm to represent nodes by low-dimensional vectors, aiming to bridge the gap between network analysis and machine learning techniques. Next, we briefly review some representative network embedding methods, and readers are referred to [8] for a comprehensive survey.

Early network embedding methods, also called graph embedding, are studied as a dimension reduction problem [36]. However, these methods focus on the pairwise similarity. How to preserve the high-order proximities becomes an attracting research problem very recently. DeepWalk [23] first proposes using truncated random walks to explore the high-order proximities and utilizes the skip-gram model [20] to derive the embedding vectors. LINE [30] takes a similar idea but sets the walk length as one with an explicit objective function. Node2vec [13] is further proposed with potentially biased random walks for more flexibility. As shown in [4, 25], these random walk based methods are equivalent to factorizing high-order proximity matrices. By using efficient optimization methods such as Stochastic Gradient Descend (SGD) [30], these methods are more scalable than some matrix factorization methods, but cannot guarantee the global optimal solutions.

On the other hand, explicit matrix factorization methods are also adopted for preserving the high-order proximities. GraRep [3] directly applies SVD to high-order proximity matrices and achieves good performance. Non-negative matrix factorization is applied in [35] to preserve the high-order proximity as well as the community structure of the network. However, both methods suffer from scalability issues and cannot be applied to large-scale networks. To address the efficiency problem, [4] introduces a unified framework for preserving high-order proximities and proposes a sparsification technique to speed up the SVD method. Another approximation technique is introduced in matrix factorization in [38]. Despite their remarkable improvements, these methods still cannot guarantee the global optimal solutions. One exception to this dilemma is HOPE [21], which uses generalized SVD to preserve high-order proximities in directed networks with a linear time complexity. However, it requires a special form of the high-order proximities. Actually, the objective function of HOPE is a special case of our method by setting the order as infinity and weights to be exponentially decaying (see Definition 3.1). Except that, HOPE cannot preserve proximities of other orders or shift across different orders.

Deep learning method is also studied in preserving the high-order proximities. SDNE [34] first considers the high non-linearity in network embedding and preserves both the first and the second order proximities using a deep auto-encoder. However, SDNE can only preserve the first two order proximities. Besides, it also has efficiency issues.

How to incorporate side information in network embedding is also explored. For example, [5, 9] introduce metapaths into embedding heterogenous information networks to handle node types. Node attributes and node labels are incorporated into network embedding in [17, 37] and [22, 32, 39] respectively. Dynamic network embedding [19, 41, 42] further considers the evolving characteristic of networks. DHNE [33] extends SDNE to preserve the indecomposability in hyper-networks. In this paper, we focus on the most fundamental case that only the static network structure is available.

To summarize, how to guarantee efficiency and accuracy simultaneously in preserving the high-order proximities largely remains open in the literature. In addition, the existing methods can only preserve fixed-order proximities and cannot shift across orders.

3 PROBLEM FORMULATION

3.1 Notation

Suppose we have a network G with N nodes and M edges. We use $\mathbf{A}$ to denote the adjacency matrix. $\mathbf{A}(i,:)$ and $\mathbf{A}(:,i)$ stand for its i^{th} row and column, respectively. $\mathbf{A}(i,j)$ is the weight of the edge between nodes i and j. In this paper, we mainly consider undirected networks, so $\mathbf{A}$ is symmetric and $\mathbf{A}(i,j) = \mathbf{A}(j,i)$. $\mathbf{A}(i,j)$ is 0 or 1 for unweighted networks, and any non-negative number for weighted networks. $\mathbf{A}^T$ denotes the transpose of $\mathbf{A}$. Throughout the paper, we use bold uppercase characters to denote matrices and bold

[1]The code is available at http://nrl.thumedialab.com/

lowercase characters to denote vectors, e.g. $\mathbf{X}$ and $\mathbf{x}$ respectively. Functions are marked by curlicue, e.g. $\mathcal{F}(\cdot)$.

3.2 Arbitrary-Order Proximity Preserved Network Embedding

Definition 3.1 (High-Order Proximity). Given the adjacency matrix $\mathbf{A}$ of an undirected network, a high-order proximity is defined as a polynomial function $\mathcal{F}(\cdot)$ of $\mathbf{A}$:

$$\mathbf{S} = \mathcal{F}(\mathbf{A}) = w_1\mathbf{A} + w_2\mathbf{A}^2 + ... + w_q\mathbf{A}^q, \tag{1}$$

where q is the order and $w_1, ..., w_q$ are the weights. Note that we refer a proximity of order q as the weighted combination of all the orders from the 1^{st} to the q^{th}, rather than the q^{th} order alone. We allow $q = +\infty$ if the summation converges. We will assume that $w_i \geq 0$ for $\forall\ 1 \leq i \leq q$, following the previous works. When different high-order proximities are referred, we use subscripts to distinguish, e.g. $\mathcal{F}_i(\cdot)$.

The previous work has shown that many state-of-the-art network embedding methods explicitly or implicitly preserve this high-order proximity [4, 25, 38]. It is worth mentioning that some work shows that an additional non-linear wrapping function is important [4], while other work demonstrates that the extra function has a limited effect [38]. We do not consider the wrapping function in this paper and leave it as the future work.

The adjacency matrix $\mathbf{A}$ could be replaced by other variations, such as the Laplacian matrix [1], provided that the substituting matrix is sparse and symmetric. For simplicity, we focus on the adjacency matrix in the rest of the paper, but similar ideas can be straight-forwardly generalized. We also reuse the notation of $\mathcal{F}(\cdot)$ such that when the function is applied to a number, the matrix product in Eq. (1) is replaced by the product of numbers.

To preserve the high-order proximity in a low-dimensional vector space, the widely adopted method is matrix factorization, which minimizes the following objective function:

$$\min_{\mathbf{U}^*, \mathbf{V}^*} \left\| \mathbf{S} - \mathbf{U}^* \mathbf{V}^{*T} \right\|_F^2, \tag{2}$$

where $\mathbf{U}^*, \mathbf{V}^* \in \mathbb{R}^{N \times d}$ are content/context embedding vectors and d is the dimensionality of the space. Without loss of generality, we use $\mathbf{U}^*$ as the content embedding vectors.

From Ercart-Young theorem, the global optimal solution to Eq. (2) can be obtained by truncated SVD [10]. Specifically, denote $[\mathbf{U}, \Sigma, \mathbf{V}]$ as the top-d SVD results of $\mathbf{S}$, where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{N \times d}$ and each column corresponds to one left/right singular vector, and $\Sigma \in \mathbb{R}^{d \times d}$ is a diagonal matrix of singular values in descending order. The embeddings can be obtained by multiplying Σ into $\mathbf{U}, \mathbf{V}$:

$$\mathbf{U}^* = \mathbf{U}\sqrt{\Sigma}, \mathbf{V}^* = \mathbf{V}\sqrt{\Sigma}. \tag{3}$$

However, directly calculating $\mathbf{S}$ and performing SVD will be both time and space consuming. Besides, since different networks and target applications usually require proximities of different orders, how to shift across different orders is also challenging. In the next section, we show a scalable solution to preserving arbitrary-order proximities that supports shifts across orders based on the above formulations.

4 AROPE: THE PROPOSED METHOD

4.1 Problem Transformation

To solve the SVD problem in Eqs. (2)(3), we first transform it into an eigen-decomposition problem. Denote the top-d eigen-decomposition of $\mathbf{S}$ as $[\mathbf{\Lambda}, \mathbf{X}]$, where $\mathbf{\Lambda} \in \mathbb{R}^{d \times d}$ is a diagonal matrix of eigenvalues in descending order of the absolute value, $\mathbf{X} \in \mathbb{R}^{N \times d}$ and each column corresponds to an eigenvector. $[\mathbf{\Lambda}(i,i), \mathbf{X}(:,i)]$, $1 \leq i \leq d$ is also referred as an eigen-pair. Then, SVD and eigen-decomposition are related by the following theorem.

Theorem 4.1. *For any symmetric matrix* $\mathbf{S}$, $\forall\ 1 \leq i \leq d$, *we have:*

$$\begin{cases} \mathbf{U}(:,i) = \mathbf{X}(:,i) \\ \Sigma(i,i) = abs(\mathbf{\Lambda}(i,i)) \\ \mathbf{V}(:,i) = \mathbf{X}(:,i) sign(\mathbf{\Lambda}(i,i)) \end{cases}, \text{and} \tag{4}$$

$$\begin{cases} \mathbf{X}(:,i) = \mathbf{U}(:,i) \\ \mathbf{\Lambda}(i,i) = \Sigma(i,i) sign\left(\mathbf{U}(:,i) \cdot \mathbf{V}(:,i)\right) \end{cases}, \tag{5}$$

where $abs(x) = |x|$ *stands for the absolute value function and* $sign(\cdot)$ *is the sign function, i.e.* $sign(x) = 1$ *if* $x > 0$, $sign(x) = 0$ *if* $x = 0$ *and* $sign(x) = -1$ *if* $x < 0$.

The proof can be found in linear algebra textbooks such as [29]. Using the theorem, we can easily get the results of SVD from the eigen-decomposition by Eq. (4) and vice versa by Eq. (5). Next, we only need to focus on solving the eigen-decomposition of $\mathbf{S}$.

4.2 Eigen-Decomposition Reweighting

To efficiently solve the eigen-decomposition of $\mathbf{S}$, our key finding is that by leveraging the form of arbitrary-order proximities defined in Eq. (1), the eigen-decomposition results of different proximities are highly correlated. Specifically, we have the following theorem:

Theorem 4.2 (Eigen-Decomposition Reweighting). *If* $[\lambda, \mathbf{x}]$ *is an eigen-pair of* $\mathbf{A}$, *then* $[\mathcal{F}(\lambda), \mathbf{x}]$ *is an eigen-pair of* $\mathbf{S} = \mathcal{F}(\mathbf{A})$.

Proof. According to the definition of an eigen-pair:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}. \tag{6}$$

Then, we can easily get:

$$\mathbf{A}^2\mathbf{x} = \mathbf{A}\lambda\mathbf{x} = \lambda\mathbf{A}\mathbf{x} = \lambda^2\mathbf{x}. \tag{7}$$

By definition, $\left[\lambda^2, \mathbf{x}\right]$ is an eigen-pair of $\mathbf{A}^2$. By using the definition of $\mathbf{S} = \mathcal{F}(\mathbf{A})$ and repeating the above process, we have:

$$\mathbf{S}\mathbf{x} = \mathcal{F}(\mathbf{A})\mathbf{x} = \left(w_1\lambda + w_2\lambda^2 + ... + w_q\lambda^q\right)\mathbf{x} = \mathcal{F}(\lambda)\mathbf{x}, \tag{8}$$

which completes the proof. □

The theorem shows that, without performing the eigen decomposition on $\mathbf{S}$, we can get the eigen-decomposition results of $\mathbf{S}$ from the eigen-decomposition results of $\mathbf{A}$ by replacing λ with $\mathcal{F}(\lambda)$. In fact, the theorem reveals the intrinsic relationship between proximities of different orders. If we regard each eigenvector as a "coordinate" of the nodes in the network and each eigenvalue as a "weight" of the coordinate, then, preserving proximities of different orders is equivalent to reweighting the dimensions.

Another issue is that after the eigen-decomposition reweighting, the order of the eigenvalues may change, i.e. the top-d eigen-decomposition of $\mathbf{S}$ is not necessarily the reweighting of the top-d eigen-decomposition of $\mathbf{A}$.

To tackle that problem, we can prove that the top-d eigen decomposition of any $\mathbf{S}$ is guaranteed to be the reweighting of the top-l eigen-decomposition of $\mathbf{A}$, where $l = \mathcal{L}(\mathbf{A}, d)$ is a function of the network and d. Specifically, denote $\lambda'_1, \lambda'_2, ..., \lambda'_d$ as the top-d eigenvalues of $\mathbf{S} = \mathcal{F}(\mathbf{A})$ in descending order of the absolute value. From Theorem 4.2, we have:

$$\exists\, p_i \; s.t. \; \lambda'_i = \mathcal{F}(\lambda_{p_i}), \forall 1 \le i \le d, \tag{9}$$

i.e. p_i is the order of λ'_i before the high-order transformation $\mathcal{F}(\cdot)$. As a result, we only need the top $p_i, 1 \le i \le d$ eigen-decomposition of $\mathbf{A}$ to get $\lambda'_1, ..., \lambda'_d$ for $\mathbf{S}$. We can prove the following theorem:

Theorem 4.3. *If $\mathcal{F}(\cdot)$ satisfies Definition 3.1, then:*

$$1 \le p_i \le l \quad \forall 1 \le i \le d, \tag{10}$$

where l satisfies that the top-l eigenvalues of $\mathbf{A}$ have d positive, i.e.

$$l = \mathcal{L}(\mathbf{A}, d) = \min\; l' \quad s.t. \; \sum_{j=1}^{l'} \mathbb{I}\left(\lambda_j > 0\right) = d, \tag{11}$$

where $|\lambda_1| \ge |\lambda_2| \ge ... \ge |\lambda_l|$ are the top-l eigenvalues of $\mathbf{A}$ in descending order of the absolute value and $\mathbb{I}(\cdot)$ is 1 when the condition is satisfied and 0 other wise.

Proof. We prove the theorem by showing that any eigenvalue except the top-l cannot be larger in absolute value than the d positive eigenvalues after $\mathcal{F}(\cdot)$. Assuming $|\lambda_i| \ge |\lambda_j|$ and $\lambda_i > 0$, we have:

$$\begin{aligned}|\mathcal{F}(\lambda_i)| &= \left|w_1\lambda_i + ... + w_q\lambda_i^q\right| = w_1|\lambda_i| + ... + w_q|\lambda_i|^q \\ &\ge w_1\left|\lambda_j\right| + ... + w_q\left|\lambda_j\right|^q \ge \left|w_1\lambda_j + ... + w_q\lambda_j^q\right| = \left|\mathcal{F}(\lambda_j)\right|.\end{aligned} \tag{12}$$

The equation is resulted from the fact that all $w_i \ge 0$.

Using Eq. (11), we know that

$$\begin{aligned}&\exists\, 1 \le c_1 < c_2 ... < c_d = l \\ &s.t.\; \lambda_{c_1} \ge \lambda_{c_2} \ge ... \ge \lambda_{c_d} > 0,\end{aligned} \tag{13}$$

i.e. $c_i, 1 \le i \le d$ are indexes for the positive eigenvalues.

For $\forall j > l, \forall 1 \le i \le d$, we have:

$$\left|\lambda_{c_i}\right| \ge \left|\lambda_j\right| \;\; and \;\; \lambda_{c_i} > 0. \tag{14}$$

From Eq. (12), $\left|\mathcal{F}(\lambda_{c_i})\right| \ge \left|\mathcal{F}(\lambda_j)\right|$, which leads to the theorem. □

We also show that Theorem 4.3 is tight in the following corollary.

Corollary 4.4. *For any $\mathbf{A}$, $p_d = l$ is achievable in Eq. (10), i.e. the top-l eigen-decomposition of $\mathbf{A}$ is a tight requirement for calculating the top-d eigen-decomposition of $\mathbf{S}$.*

Proof. Consider $q = +\infty$ for $\mathcal{F}(\cdot)$ and let the weights be exponentially decaying, i.e. $w_i = \beta^i$ and $\beta > 0$ is a constant. To ensure that the high-order proximity function converges, we assume that β is smaller than the inverse of the spectral radius of $\mathbf{A}$ [29].

Then, for $\forall\; 1 \le i \le d$, we have:

$$\lambda'_i = \mathcal{F}(\lambda_{p_i}) = \beta\lambda_{p_i} + \beta^2\lambda_{p_i}^2 + ... = \frac{\beta\lambda_{p_i}}{1 - \beta\lambda_{p_i}}, \tag{15}$$

Algorithm 1 AROPE: ARbitrary-Order Proximity preserved Embedding

Require: Adjacency Matrix $\mathbf{A}$, Dimensionality d, Different High-Order Proximity Functions $\mathcal{F}_1(\cdot), ..., \mathcal{F}_r(\cdot)$
Ensure: Embedding vectors $\mathbf{U}_i^*, \mathbf{V}_i^*$ for $\mathcal{F}_i(\cdot), 1 \le i \le r$
1: Calculate the top-l eigen-decomposition $[\mathbf{\Lambda}, \mathbf{X}]$ of $\mathbf{A}$
2: **for** i in 1:r **do**
3: Calculate the reweighted eigenvalues $\mathbf{\Lambda}' = \mathcal{F}_i(\mathbf{\Lambda})$
4: Sort $\mathbf{\Lambda}'$ in descending order of the absolute value and select the top-d
5: Calculate the top-d SVD results using Eq. (4)
6: Return $\mathbf{U}_i^*, \mathbf{V}_i^*$ using Eq. (3)
7: **end for**

where p_i and λ'_i are defined in Eq. (9). It is easy to see that $\mathcal{F}(\lambda_{p_i})$ grows monotonously with λ_{p_i}. As a result, we have:

$$p_i = c_i, \forall 1 \le i \le d, \tag{16}$$

where c_i are indexes for the positive eigenvalues of $\mathbf{A}$ defined in Eq. (13). Since $c_d = l$, this completes the proof. □

The above theorems suggest that to get the top-d eigen decomposition of any $\mathcal{F}(\cdot)$, we need to calculate the top-l eigen-decomposition of $\mathbf{A}$. Then, we can reweight and reorder dimensions, and use the top-d after reweighting to derive the embedding vectors.

As for the relationship between d and l, for the simplest case of random networks, i.e. the Erdos Renyi model, we have an expected relation of $l \approx 2d$ for sufficiently large networks due to the Wigner Semicircle Law [11]. For random networks with a power-law distribution, the same relation can be induced under mild conditions [7]. For real networks, the relationship depends on the individual network structure, but we verify that $l \le 2d$ is satisfied for all the networks which we experiment on.

An important observation is that since the top-l eigen decomposition of $\mathbf{A}$ is shared by arbitrary-order proximities, we can shift between proximities of different orders with a low marginal cost by pre-computing the eigen-decomposition.

4.3 The Framework and Complexity Analysis

We show our algorithm framework in Algorithm 1. Our proposed method, AROPE, can get the global optimal solutions to arbitrary-order proximities by utilizing the eigen-decomposition reweighting theorem. In addition, our method can deal with the general case that proximities of different orders are arbitrarily weighted. A simplified version of our method is to assign equal weights or exponentially decaying weights to different orders. One may also concern about the numerical stabilities of our method, especially when the eigenvalues are very similar. In practice, we find that out method is quite stable, potentially due to the success of established numerical methods in solving the eigen-decomposition problem.

From the algorithm, we can see that the time complexity of line 1 is $O\left(T(Nl^2 + Ml)\right)$ by using iterative approaches [16], where N is the number of nodes, M is the number of edges and T is the number of iterations. The time complexity of each loop from line 3 to line 6 is $O\left(l + Nd\right)$. So the total complexity is $O\left(T(Nl^2 + Ml) + r(l + Nd)\right)$,

which exhibits two merits. First, the complexity is linear with respect to the network size (M and N respectively), so our method is scalable and can be applied to large-scale networks. Second, since $l \geq d$ and usually $T >> r$, the first term dominates and it takes low additional time to compute multiple arbitrary-order proximities, i.e. our method supports shifts across orders with a low marginal cost.

4.4 Special Cases of the Proposed Method

Next, we show that our method incorporates many commonly used high-order proximities as special cases.

4.4.1 Common Neighbors and Propagation. A simple and widely studied proximity measure in network analysis is the common neighbors [18], i.e. the structure of neighbors of neighbors for nodes is reflected. Our method can preserve the common neighbors if we set the order as $q = 2$ and weights to be $w_1 = 0, w_2 = 1$, i.e.

$$\mathbf{S} = \mathcal{F}(\mathbf{A}) = \mathbf{A}^2. \tag{17}$$

To generalize the common neighbors, some propagation based proximities are proposed, which consider higher-order structures of the network. Our method incorporates a commonly used propagation based proximity in [14] by setting $\mathbf{S}$ as:

$$\mathbf{S} = \mathcal{F}(\mathbf{A}) = w_2\mathbf{A}^2 + w_3\mathbf{A}^3, \tag{18}$$

where w_2, w_3 are weights of the propagation.

4.4.2 Katz Proximity. Another commonly used proximity measure is the Katz proximity [15], which is an ensemble of proximities of different orders with exponentially decaying weights. Our method can preserve the Katz proximity defined as:

$$\mathbf{S} = \mathcal{F}(\mathbf{A}) = \sum_{i=1}^{\infty} \beta^i \mathbf{A}^i, \tag{19}$$

where β is a constant that controls how fast the weights decay. To ensure that the proximity converges, β must be smaller than the inverse of the spectral radius of $\mathbf{A}$ [29].

4.4.3 Eigenvector Centrality. Another special case of our method is to set the dimensionality as $d = 1$, i.e. we only analyze the effect of the first dimension. We find that the first dimension is actually highly related to the eigenvector centrality [2], a widely applied approach to measure the importance of nodes in the network. We specify this case in the following theorem.

Theorem 4.5. *For any $\mathcal{F}(\cdot)$, the first dimension of embedding vectors of AROPE is proportional to the eigenvector centrality of nodes, i.e. $\mathbf{U}^*(:,1) \propto \mathbf{ec}$, where $\mathbf{ec}$ is a vector of the eigenvector centrality.*

Proof. By definition, the eigenvector centrality is the eigenvector of the adjacency matrix corresponding to the largest eigenvalue, i.e. $\mathbf{ec} = \mathbf{X}(:,1)$. Then, we only need to prove that this eigenvector is proportional to the first dimension of our embedding vectors.

Using Theorem 4.2 and Eqs. (3)(4), we have:

$$\mathbf{U}^*(:,1) = \mathbf{U}(:,1)\sqrt{|\lambda_1'|} = \mathbf{X}(:,p_1)\sqrt{|\mathcal{F}(\lambda_{p_1})|}. \tag{20}$$

From [26], we know that $\lambda_1 \geq d_{ave} > 0$, where d_{ave} is the average degree of the network. Then, from Theorem 4.3, we have $p_1 = 1$, which leads to the result. □

Table 1: The Statistics of Datasets

Dataset	# Nodes	# Edges	Average Degree
BlogCatalog	10,312	667,966	64.8
Flickr	80,513	11,799,764	146.6
Youtube	1,138,499	5,980,886	5.3
Wiki	1,791,486	50,888,414	28.4

The theorem shows that the first dimension of our embedding vectors contains all the information of the eigenvector centrality regardless of what high-order proximity is used. In other words, we can use our embedding vectors to measure the importance of nodes. Interestingly, this is connected to the structural identity of nodes, which is a recently emerging topic in network embedding [27]. Although we do not show such aim in designing our method, the eigenvector centrality turns out to be a special case of our method.

5 EXPERIMENTS

5.1 Experimental Setting

To comprehensively evaluate our proposed method, we conduct extensive experiments on the following four real networks:

- BlogCatalog[2], Flickr[2], Youtube[2] [31]: they are online social networks where nodes represent users and edges represent relationships between users, so the networks are undirected.
- Wiki[3] [40]: this is a network of Wikipedia hyperlinks. Each node represents a page and each edge represents a hyperlink between two pages. We treat the edges as undirected since we only consider undirected networks in this paper.

All statistics of the datasets are summarized in Table 1. We compare our method with the following state-of-the-art methods:

- DeepWalk [23][4] uses truncated random walks and the skip-gram model to learn embedding vectors. We use two settings of parameters: one suggested by the paper and one recommended in the implementation of the authors, and report the best results.
- LINE [30][5] explicitly preserves the first two order proximities, denoted as LINE_{1st} and LINE_{2nd} respectively. For brevity, we exclude the results of concatenating them because they show no obvious improvement. We use the default settings for all parameters except the total number of training samples, which we conduct a line search for the optimal value.
- Node2vec [13][6] generalizes DeepWalk by adopting potentially biased random walks. We vary the bias parameters p, q from $\{0.5, 1, 2\}$ and use the default settings for other parameters.
- SDNE [34][7] adopts a deep auto-encoder to preserve the first two order proximities. We use the default neural network structure and parameters in the implementation of the authors.
- NEU [38][8] proposes enhancing other network embedding vectors by approximating a higher-order proximity. We use the

[2] http://socialcomputing.asu.edu/datasets
[3] http://snap.stanford.edu/data/wiki-topcats.html
[4] https://github.com/phanein/deepwalk
[5] https://github.com/tangjianpku/LINE
[6] https://github.com/aditya-grover/node2vec
[7] https://github.com/suanrong/SDNE
[8] https://github.com/thunlp/NEU

results of DeepWalk as their base embeddings and use other parameters suggested by the paper.

We exclude some other network embedding methods, such as GraRep [3] and M-NMF [35], for their scalability issues, and exclude HOPE [21] for being a special case of our method (see Section 2). We also exclude the results of Node2vec and SDNE on Wiki because the former runs out of memory even with the 384GB memory provided, and the latter fails to terminate in one week.

For our method, we present two variations:

- AROPE: we vary the order q from $\{1, 2, 3, 4\}$ with a grid search for the weights. Note that we can efficiently adjust these hyper-parameters by shifting across different orders and weights.
- AROPE-F: we also vary the order q from $\{1, 2, 3, 4\}$, but fix the weights as $w_i = 0.1^i$, i.e. a simplified version where we greatly shrink the searching space for hyper-parameters and reduce the effect of weights on different orders.

All hyper-parameters of our method and the baselines are tuned using a small validation set, which we set as 10% in our experiments. For all the methods, we uniformly set the dimensionality as $d = 128$ unless stated otherwise.

All our experiments are conducted in a single PC with 2 I7 processors and 48GB memory, except for Node2vec, which we run in a server with 384GB memory because of its intensive memory usage.

5.2 Preserving the High-Order Proximity

Since our objective function is to preserve high-order proximities, we first test the effectiveness and efficiency of our method in this task. As other network embedding methods do not explicitly minimize the objective function in Eq. (2), we compare our method with the following two baselines:

- SVD. We directly calculate the high-order proximities and perform SVD. It has the optimal accuracy but is not scalable.
- SPE [28] is a scalable approximation to preserve the high-order proximities by calculating partial SVD on a submatrix of randomly chosen landmark nodes. We use a landmark size of 1600 as suggested by the paper.

For the high-order proximities, we vary the order q from 2 to 6 and set the weights to be $w_i = 0.1^i$, i.e. the same as AROPE-F. Note that this is simply an illustrating showcase for the task and we support other orders and weights as well. To compare different methods, we report the ratio of the objective function value in Eq. (2) to the theoretical lower bound calculated by SVD, and the running time of each method. We only use BlogCatalog in this task because SVD is very time consuming. The average results of 5 runs are reported.

The results in Figure 1 show that AROPE gets the global optimal solutions to all proximities as SVD. In comparison, SPE has much worse performance and the error grows larger as the order becomes higher. From the efficiency perspective, AROPE reports the minimum running time and the improvement is especially significant when the order becomes high, as AROPE reduces the running time by orders of magnitude and the increase in running time of AROPE is barely noticeable. These results demonstrate that AROPE can achieve accuracy and efficiency simultaneously in preserving arbitrary-order proximities.

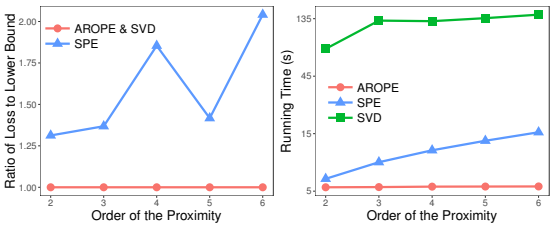

Figure 1: Results of preserving the high-order proximities on BlogCatalog. AROPE gets the global optimal solutions as SVD while being much more efficient.

5.3 Network Reconstruction

One primal objective for network embedding is to reconstruct the network [34]. Specifically, we train embedding vectors and rank pairs of nodes according to their similarities, i.e. the inner product of two embedding vectors. The highest ranking pairs of nodes are used to reconstruct the network. For the evaluation metric, we use precision defined as:

$$Precision@N_p = \frac{1}{N_p} \sum_{i=1}^{N_p} \delta_i, \tag{21}$$

where $\delta_i = 1$ means the i^{th} reconstructed pair is correct (i.e. the edge exists in the network), $\delta_i = 0$ otherwise and N_p is the number of evaluated pairs. For Youtube and Wiki, the number of possible pairs of nodes $\frac{N(N-1)}{2}$ is too large to evaluate, so we sample 1% for evaluation, as in [21].

Figure 2 shows that our proposed methods, AROPE and AROPE-F, consistently outperform the baselines when the number of reconstruction pairs varies from one hundred to one million. On all networks, for the top ten thousand evaluated pairs, AROPE achieves more than 90% precision. AROPE slightly outperforms AROPE-F because of the flexibility in adjusting the weights, but AROPE-F still greatly outperforms the baselines. The improvement over the baselines is more substantial on Youtube and Wiki because reconstruction usually becomes more challenging when the network grows larger and the number of possible pairs grows quadratically. Node2vec, as a generalization of DeepWalk, surprisingly has even worse performance. SDNE performs well on BlogCatalog, the smallest network, but has unsatisfactory results on the large-scale network Youtube. All methods converge to roughly the same point when the number of reconstruction pairs grows sufficiently large since nearly all of the true edges in the evaluated pairs are correctly recovered.

The above results show that our method can better preserve the network structure by using the SVD framework and preserving arbitrary-order proximities, which lays the foundation for applying our method to other applications of network embedding.

5.4 Link Prediction

Link prediction, aiming to predict which pairs of nodes are likely to form edges, is a typical application of network embedding. In our experiments, we randomly hide 30% of the edges for testing and train embedding vectors on the rest of the network. Then, we rank pairs of nodes in a similar way as network reconstruction and evaluate the highest ranking pairs on the testing network. We

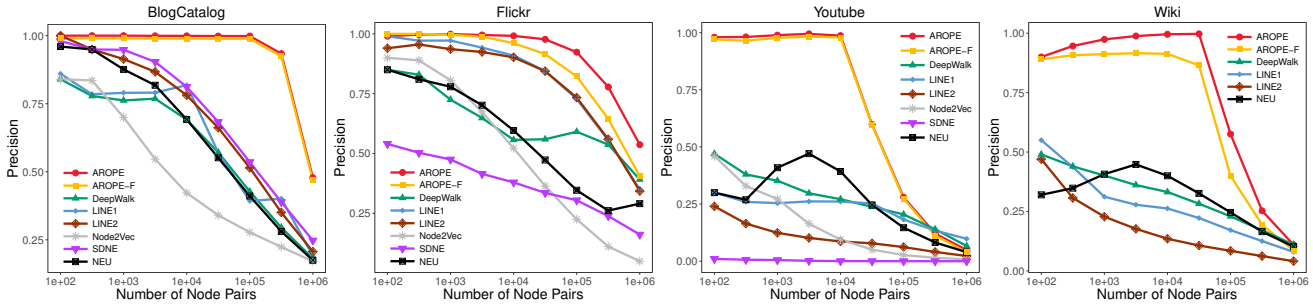

Figure 2: Network reconstruction results. The experimental results show that our proposed method has much higher precision, suggesting that we can reconstruct the given network significantly better. The optimal order of AROPE and AROPE-F for network reconstruction on these networks is 1, 2, 1, 2 (left to right) respectively.

also adopt precision defined in Eq. (21) as the measurement. The average results of 5 runs are reported in Figure 3.

The results show that our proposed method outperforms the baselines significantly and consistently. Similarly to the results of network reconstruction, the improvement is more substantial on Youtube and Wiki as we achieve more than 100% improvements for the top one thousand predictions. The precision scores for all the methods are lower on Youtube than other networks because Youtube is much sparser and prediction is more challenging. But we still achieve significantly better results in that case. The results demonstrate that by better preserving arbitrary-order proximities, we can greatly improve the link prediction performance, verifying the generalization ability of our embedding vectors.

We also conduct experiments of node recommendation, which shows similar patterns as link prediction. We exclude the results here due to the space limit.

5.5 Node Structural Role Classification

Recently, how to preserve the structural role of nodes in network embedding attracts some research attention [27], which has important applications such as measuring node centrality or influence maximization [6]. Since we show that the eigenvector centrality is a special case of our method in Section 4.4.3, our method implicitly preserves the structural role of nodes.

To validate that, we conduct experiments on two air-traffic networks[9] from Brazilian and European as in [27]. The networks constitute 131 nodes and 2006 edges, and 399 nodes and 11986 edges respectively, where nodes indicate airports and edges correspond to commercial airlines. The nodes are assigned a ground-truth label ranging from 1 to 4 to indicate the level of activities of the corresponding airports.

We randomly split the nodes into a training set and a testing set. Then, we train a one-vs-all logistic regression with L2 regularization [12] using the content embedding vectors on the training set, and test on the testing set. We use accuracy, i.e. the percentage of nodes whose labels are correctly classified, to evaluate the performance since the label size is balanced. The benchmark accuracy of random guessing is also added. For all embedding methods, we set the dimensionality as $d = 16$ due to the limited size of the network. The average results of repeating the process 20 times are reported.

From Figure 4, we can see that our method consistently outperforms the baselines on both networks. The results demonstrate that our method indeed captures the structural role of nodes by using the SVD framework, which is an additional benefit besides preserving the arbitrary-order proximities. Other network embedding methods for preserving the proximities, to the contrary, are less competent in this task. Here we exclude the results of comparing with other structural-based methods because they cannot be applied to proximity-based tasks (such as link prediction).

5.6 Parameter Analysis

5.6.1 The Order of the Proximity. In AROPE, one important parameter is the order of the proximity q. Here, we analyze the effect of q. Specifically, we vary q from $\{1, 2, 3, 4, +\infty\}$ with a grid search for the best weights for each order. For brevity, we only report three typical results: network reconstruction on Flickr, link prediction on BlogCatalog and node structural role classification on Brazilian Flight in Figure 5.

The results show that the high-order proximities ($q > 1$) outperform the pairwise proximity ($q = 1$) in most cases, demonstrating the effectiveness of the high-order proximities in network embedding. However, the optimal q varies greatly on different networks and target applications. For example, $q = 3$ has the best results on BlogCatalog for link prediction, and $q = 4$ shows a promising accuracy on Brazilian Flight for node classification. For network reconstruction on Flickr, different measurements require different orders. These results demonstrate that proximities of different orders are critical to real applications and it is usually hard to predetermine the order. For our method, since it provides the flexibility of shifts across arbitrary orders with a low marginal cost, we can efficiently adjust the order by using a small validation set.

5.6.2 The Number of High-order Proximities. In Section 4.3, we proved that our method can calculate embedding vectors for multiple arbitrary-order proximities with a low additional cost to support shifts across orders. Now we verify this point by experiments. Specifically, we record the running time of our method when the number of proximities increases, i.e. shifts across orders.

[9] https://github.com/leoribeiro/struc2vec

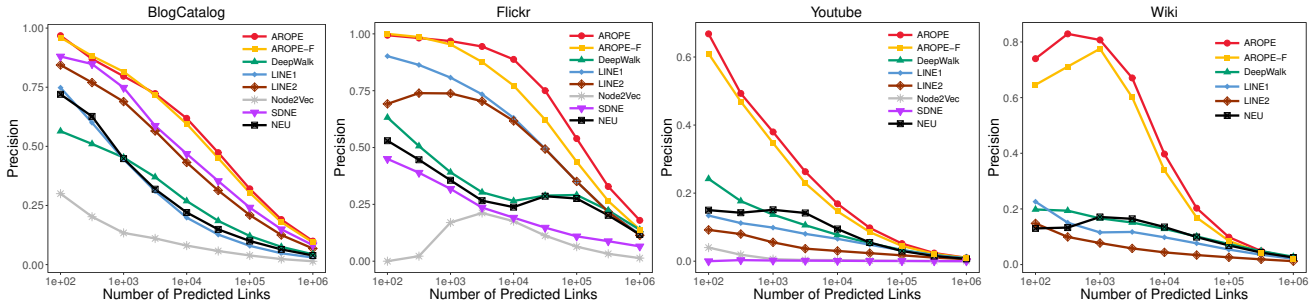

Figure 3: Link prediction results. The results show that our proposed method has significantly higher precision in link prediction, showing the superior performance of our method in network inference tasks. The optimal order of AROPE and AROPE-F for link prediction on these networks is 3, 4, 2, 3 (left to right) respectively.

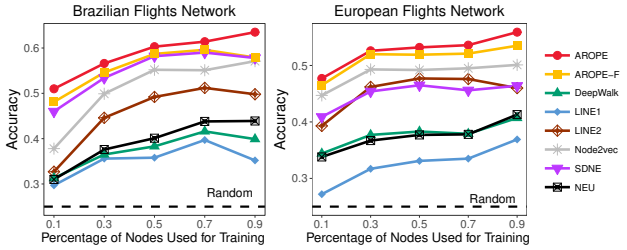

Figure 4: Results of node structural role classification on two flights networks. The results show that our method has higher accuracies, demonstrating better performance in capturing the structural role of nodes on these two networks.

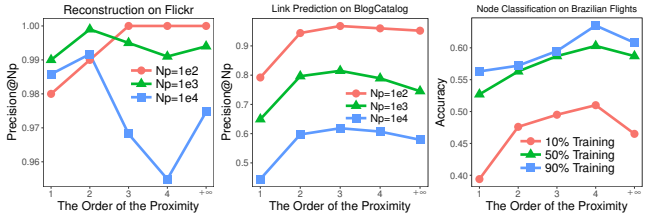

Figure 5: Analyzing the task performance as a function of the order of the proximity q. The results show that the high-order proximities ($q > 1$) usually outperforms the pairwise proximity ($q = 1$). However, the optimal q varies greatly on different networks and applications, suggesting the importance of shifts across different orders.

The results in Figure 6 show that the running time increases very slightly when the number of proximities increases, suggesting that we can get embedding vectors for multiple arbitrary-order proximities with a low marginal cost. In this way, we can easily shift across different orders and weights instead of starting from scratch each time as the baselines.

5.6.3 Scalability Analysis. Here we conduct experiments to verify the scalability of our method. We apply our method to synthetic networks of different sizes generated by random graphs, i.e. the Erdos Renyi model [11], and record the running time. The results

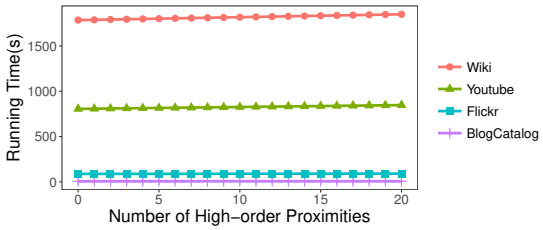

Figure 6: The running time vs. the number of high-order proximities. The results show that the running time increases very slightly with respect to the number of proximities.

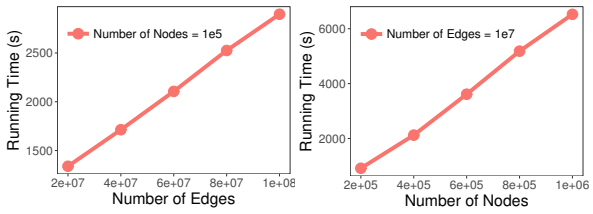

Figure 7: The scalability analysis of AROPE on synthetic networks. The results show that our method has a linear time complexity with respect to the number of nodes and number of edges respectively. In addition, our method is extremely efficient, which can be applied to real large-scale networks.

of fixing the number of nodes (as one hundred thousand) or fixing the number of edges (as ten million) while varying the other are plotted in Figure 7 respectively.

The figures show that the running time grows linearly with the number of nodes and the number of edges respectively, verifying the scalability of our method. It is worth noting that even for the largest network with one million nodes and ten million edges, AROPE spends less than two hours in a single PC, i.e. our method is extremely efficient and can be applied to real large-scale networks.

6 CONCLUSION

In this paper, we study the problem of preserving arbitrary-order proximities in network embedding. By proving and utilizing the

eigen-decomposition reweighting theorem, we extract the intrinsic relationship between proximities of different orders and propose AROPE, a scalable solution to derive the embedding vectors. Theoretical analysis shows that i) our method supports shifts across arbitrary orders with a low marginal cost, ii) given a certain order, our method can get the global optimal solutions, and iii) the overall time complexity of our method is linear with respect to the network size. Extensive experimental results demonstrate the efficacy of our method in several applications of network embedding. One future direction is to generalize this framework to directed networks and incorporate side information such as node contents and node labels.

ACKNOWLEDGMENTS

This work was supported in part by National Program on Key Basic Research Project (No. 2015CB352300), National Natural Science Foundation of China (No. 61772304, No. 61521002, No. 61531006, No. 61702296), National Natural Science Foundation of China Major Project (No. U1611461), the research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, and the Young Elite Scientist Sponsorship Program by CAST. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Mikhail Belkin and Partha Niyogi. 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*. 585–591.
[2] Phillip Bonacich. 2007. Some unique properties of eigenvector centrality. *Social Networks* 29, 4 (2007), 555–564.
[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM.
[4] Siheng Chen, Sufeng Niu, Leman Akoglu, Jelena Kovačević, and Christos Faloutsos. 2017. Fast, Warped Graph Embedding: Unifying Framework and One-Click Algorithm. *arXiv:1702.05764* (2017).
[5] Ting Chen and Yizhou Sun. 2017. Task-Guided and Path-Augmented Heterogeneous Network Embedding for Author Identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 295–304.
[6] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data mining*. ACM, 199–208.
[7] Fan Chung, Linyuan Lu, and Van Vu. 2003. Spectra of random graphs with given expected degrees. *Proceedings of the National Academy of Sciences* (2003).
[8] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2017. A Survey on Network Embedding. *arXiv preprint arXiv:1711.08752* (2017).
[9] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 135–144.
[10] Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika* (1936).
[11] László Erdős, Antti Knowles, Horng-Tzer Yau, Jun Yin, et al. 2013. Spectral statistics of Erdős–Rényi graphs I: local semicircle law. *The Annals of Probability* 41, 3B (2013), 2279–2375.
[12] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* 9, Aug (2008), 1871–1874.
[13] 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.
[14] Ramanthan Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. 2004. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*. ACM, 403–412.
[15] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.
[16] Richard B Lehoucq and Danny C Sorensen. 1996. Deflation techniques for an implicitly restarted Arnoldi iteration. *SIAM J. Matrix Anal. Appl.* (1996).
[17] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed Network Embedding for Learning in a Dynamic Environment. In *Proceedings of the 26th ACM International Conference on Information and Knowledge Management*.
[18] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology* 58, 7 (2007), 1019–1031.
[19] Jianxin Ma, Peng Cui, and Wenwu Zhu. 2018. DepthLGP: Learning Embeddings of Out-of-Sample Nodes in Dynamic Networks. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.
[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv:1301.3781* (2013).
[21] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric Transitivity Preserving Graph Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.
[22] 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*. 1895–1901.
[23] 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.
[24] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. 2017. Don't Walk, Skip!: Online Learning of Multi-scale Network Embeddings. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. ACM, 258–265.
[25] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 459–467.
[26] Juan G Restrepo, Edward Ott, and Brian R Hunt. 2007. Approximating the largest eigenvalue of network adjacency matrices. *Physical Review E* 76, 5 (2007), 056119.
[27] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning Node Representations from Structural Identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 385–394.
[28] Han Hee Song, Tae Won Cho, Vacha Dave, Yin Zhang, and Lili Qiu. 2009. Scalable proximity estimation and link prediction in online social networks. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*. ACM.
[29] Gilbert Strang. 2006. *Linear Algebra and Its Applications* (4 ed.). Brooks Cole.
[30] 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*. 1067–1077.
[31] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data mining*. ACM, 817–826.
[32] Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. 2016. Max-Margin DeepWalk: Discriminative Learning of Network Representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.
[33] Ke Tu, Peng Cui, Xiao Wang, Fei Wang, and Wenwu Zhu. 2018. Structural Deep Embedding for Hyper-Networks. *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.
[34] 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.
[35] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.
[36] S. Yan, D. Xu, B. Zhang, H. J. Zhang, Q. Yang, and S. Lin. 2007. Graph embedding and extensions: a general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 1 (2007), 40.
[37] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. 2015. Network Representation Learning with Rich Text Information. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*.
[38] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. 2017. Fast Network Embedding Enhancement via High Order Proximity Approximation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*.
[39] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *International Conference on Machine Learning*. 40–48.
[40] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local Higher-Order Graph Clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 555–564.
[41] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu. 2018. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.
[42] Dingyuan Zhu, Peng Cui, Ziwei Zhang, Jian Pei, and Wenwu Zhu. 2018. High-order Proximity Preserved Embedding For Dynamic Networks. *IEEE Transactions on Knowledge and Data Engineering* (2018).