Fast Network Embedding Enhancement via High Order Proximity Approximation

Cheng Yang[1], **Maosong Sun**[2*], **Zhiyuan Liu**[2], **Cunchao Tu**[1],
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[1] {cheng-ya14,tcc13}@mails.tsinghua.edu.cn, [2] {sms,liuzy}@tsinghua.edu.cn

Abstract

Many Network Representation Learning (NRL) methods have been proposed to learn vector representations for vertices in a network recently. In this paper, we summarize most existing NRL methods into a unified two-step framework, including proximity matrix construction and dimension reduction. We focus on the analysis of proximity matrix construction step and conclude that an NRL method can be improved by exploring higher order proximities when building the proximity matrix. We propose Network Embedding Update (NEU) algorithm which implicitly approximates higher order proximities with theoretical approximation bound and can be applied on any NRL methods to enhance their performances. We conduct experiments on multi-label classification and link prediction tasks. Experimental results show that NEU can make a consistent and significant improvement over a number of NRL methods with almost negligible running time on all three publicly available datasets. The source code of this paper can be obtained from `https://github.com/thunlp/NEU`.

1 Introduction

A network is an essential data type widely used in our daily lives and academic researches, such as friendship networks in Facebook and citation networks in DBLP [Ley, 2002]. Researchers have made great effort on developing machine learning algorithms for various network applications, e.g. vertex classification [Sen *et al.*, 2008], tag recommendation [Tu *et al.*, 2014], anomaly detection [Akoglu *et al.*, 2015] and link prediction [Liben-Nowell and Kleinberg, 2007]. Most supervised machine learning algorithms applied on these applications require a set of informative features as input [Grover and Leskovec, 2016]. Hand-crafted features may suit the need but require much human effort and expert knowledge. Therefore, representation learning [Bengio *et al.*, 2013] which learns feature embeddings via optimization learning was proposed to avoid feature engineering and improve the flexibility of features. In terms of network analysis, Network Representation Learning (NRL) which aims to learn distributed real-valued embeddings for vertices of a network has attracted much attention in recent years [Perozzi *et al.*, 2014; Tang *et al.*, 2015b; Cao *et al.*, 2015; Grover and Leskovec, 2016].

In this paper, we summarize most existing NRL methods into a unified two-step framework, including proximity matrix construction and dimension reduction. The first step builds a proximity matrix M where each entry M_{ij} encodes the proximity information between vertex i and j. The second step reduces the dimension of the proximity matrix to obtain network embeddings. Different NRL methods employ various dimension reduction algorithms such as eigenvector computation and SVD decomposition. Our analysis of the first step, i.e. proximity matrix construction, shows that the quality of network embeddings can be improved when higher order proximities are encoded into the proximity matrix.

However, an accurate computation of high-order proximities is time-consuming and thus not scalable for large-scale networks. Thus we can only approximate higher order proximity matrix for learning better network embeddings. In order to be more efficient, we also seek to use the network representations which encode the information of lower order proximities as our basis to avoid repeated computations. Therefore, we propose Network Embedding Update (NEU) algorithm which could be applied to any NRL methods to enhance their performances. The intuition behind is that the embeddings processed by NEU algorithm can implicitly approximate higher order proximities with a theoretical approximation bound and thus achieve better performances.

We conduct experiments on multi-label classification and link prediction tasks over three publicly available datasets to evaluate the quality of network embeddings. Experimental results show that the network embeddings learned by existing NRL methods can be improved consistently and significantly on both evaluation tasks after enhanced by NEU. Moreover, the running time of NEU takes less than 1% training time of popular NRL methods such as DeepWalk and LINE, which could be negligible.

Our main contributions are two-fold:

1) We summarize a number of existing NRL methods into a unified framework, i.e. proximity matrix construction and dimension reduction, and conclude that the quality of network

*Corresponding author: sms@tsinghua.edu.cn

embeddings can be enhanced if higher order proximities are encoded into the proximity matrix.

2) We propose NEU algorithm to improve the performances of any network embeddings learned by existing NRL algorithms. The embeddings processed by NEU can implicitly approximate higher order proximities with theoretical bound. Experimental results on multi-label classification and link prediction demonstrate the efficiency and effectiveness of our algorithm.

Related Works Here we give a brief introduction to existing NRL methods and some of which will be thoroughly analyzed in next section. Spectral Clustering [Tang and Liu, 2011] computes top-d eigenvectors of normalized Laplacian matrix as d-dimensional network embeddings. DeepWalk [Perozzi *et al.*, 2014] employs Skip-gram [Mikolov *et al.*, 2013] model, which is originally used in word representation learning, on random walks for NRL. LINE [Tang *et al.*, 2015b] models first-order and second-order proximities between vertices for learning large-scale network embeddings. GraRep [Cao *et al.*, 2015] factorizes different k-order proximity matrices and concatenates the embeddings learned from each proximity matrix. Though GraRep achieves better performance than DeepWalk and LINE, GraRep suffers heavily from inefficiency problem. Besides the above NRL methods that focus on network topology, researchers also explore algorithms to incorporate meta information, e.g. text and label information, into NRL. TADW [Yang *et al.*, 2015] takes text information into consideration under matrix factorization framework and MMDW [Tu *et al.*, 2016] learns semi-supervised network embeddings with max-margin constraints between vertices from different labels. As another semi-supervised NRL method, node2vec [Grover and Leskovec, 2016] further generalizes DeepWalk with Breadth-First Search (BFS) and Depth-First Search (DFS) on random walks. GCN [Kipf and Welling, 2017], DDRW [Li *et al.*, 2016] and Planetoid [Yang *et al.*, 2016] are also proposed for semi-supervised graph embeddings. SDNE [Wang *et al.*, 2016] employs deep neural model for NRL. Other extensions include asymmetric transitivity [Ou *et al.*, 2016], community preserving [Wang *et al.*, 2017] and heterogenous [Tang *et al.*, 2015a; Chang *et al.*, 2015; Huang and Mamoulis, 2017; Xu *et al.*, 2017; Huang *et al.*, 2017] network embeddings. We focus on the most general case where NRL methods use only network topology in this paper.

2 Framework of Existing NRL Algorithms

In this section, we present a unified framework which can cover several representative NRL algorithms include Spectral Clustering [Tang and Liu, 2011], DeepWalk [Perozzi *et al.*, 2014], TADW [Yang *et al.*, 2015], LINE [Tang *et al.*, 2015b] and GraRep [Cao *et al.*, 2015]. First, we clarify the notations and formalize the problem of NRL. Then we introduce the concept of k-order proximity. Finally, we summarize an NRL framework based on proximity matrix factorization and show that the aforementioned NRL methods fall into the category.

Let $G = (V, E)$ be a given network where V is vertex set and E is edge set. The task of NRL is to learn a real-valued representation $r_v \in \mathbb{R}^d$ for each vertex $v \in V$ where d is the embedding dimension. We assume that networks are unweighted and undirected in this paper without loss of generality and define adjacency matrix $\widetilde{A} \in \mathbb{R}^{|V|\times|V|}$ as $\widetilde{A}_{ij} = 1$ if $(v_i, v_j) \in E$ and $\widetilde{A}_{ij} = 0$ otherwise. Diagonal matrix $D \in \mathbb{R}^{|V|\times|V|}$ denotes the degree matrix where $D_{ii} = d_i$ represents the degree of vertex v_i. $A = D^{-1}\widetilde{A}$ is normalized adjacency matrix where the summation of each row equals to 1. Similarly, we also have Laplacian matrix $\widetilde{L} = D - \widetilde{A}$ and normalized Laplacian matrix $L = D^{-\frac{1}{2}}\widetilde{L}D^{-\frac{1}{2}}$.

2.1 K-order Proximity

The (normalized) adjacency matrix and Laplacian matrix characterize the first-order proximity which models the local pairwise proximity between vertices. Note that each off-diagonal nonzero entry of the first-order proximity matrix corresponds to an edge in the network. However, real-world networks are always sparse which indicates that $O(E) = O(V)$. Therefore the first-order proximity matrix is usually very sparse and insufficient to fully model the pairwise proximity between vertices. As a result, people also explore higher order proximity to model the strength between vertices [Perozzi *et al.*, 2014; Tang *et al.*, 2015b; Cao *et al.*, 2015]. For example, the second-order proximity can be characterized by the number of common neighbors between vertices. As an alternative view, the second-order proximity between v_i and v_j can also be modeled by the probability that a 2-step random walk from v_i reaches v_j. Intuitively, the probability will be large if v_i and v_j share many common neighbors. In the probabilistic setting based on random walk, we can easily generalize it to k-order proximity [Cao *et al.*, 2015]: the probability that a random walk starts from v_i and walks to v_j with exactly k steps. Note that the normalized adjacency matrix A is the transition probability matrix of a single step random walk. Then we can compute k-step transition probability matrix as the k-order proximity matrix

$$A^k = \underbrace{A \cdot A \dots A}_{k}, \tag{1}$$

where the entry A^k_{ij} is the k-order proximity between vertex v_i and v_j.

2.2 NRL Framework

Now we have introduced the concept of k-order proximity matrix. In this subsection, we first summarize the NRL framework based on dimension reduction of a proximity matrix and then conduct a theoretical analysis to show that most existing NRL algorithms can be formalized into this framework.

We summarize NRL methods as a two-step framework:

Step 1: Proximity Matrix Construction. Compute a proximity matrix $M \in \mathbb{R}^{|V|\times|V|}$ which encodes the information of k-order proximity matrix where $k = 1, 2 \dots, K$. For example, $M = \frac{1}{K}A + \frac{1}{K}A^2 \cdots + \frac{1}{K}A^K$ stands for an average combination of k-order proximity matrix for $k = 1, 2 \dots, K$. The proximity matrix M is usually represented by a polynomial of normalized adjacency matrix A of degree K and we denote the polynomial as $f(A) \in \mathbb{R}^{|V|\times|V|}$. Here the

degree K of polynomial $f(A)$ corresponds to the maximum order of proximities encoded in proximity matrix. Note that the storage and computation of proximity matrix M does not necessarily take $O(|V|^2)$ time because we only need to save and compute the nonzero entries.

Step 2: Dimension Reduction. Find network embedding matrix $R \in \mathbb{R}^{|V|\times d}$ and context embedding $C \in \mathbb{R}^{|V|\times d}$ so that the product $R \cdot C^T$ approximates proximity matrix M. Here different algorithms may employ different distance functions to minimize the distance between M and $R \cdot C^T$. For example, we can naturally use the norm of matrix $M - R \cdot C^T$ to measure the distance and minimize it.

Spectral Clustering [Tang and Liu, 2011] computes the first d eigenvectors of normalized Laplacian matrix L as d-dimensional network representations. The information embedded in the eigenvectors comes from the first-order proximity matrix L. Note that the real-valued symmetric matrix L can be factorized as $L = Q\Lambda Q^{-1}$ via Eigendecomposition where $\Lambda \in \mathbb{R}^{|V|\times|V|}$ is a diagonal matrix, $\Lambda_{11} \geq \Lambda_{22} \geq \ldots \Lambda_{|V||V|}$ are eigenvalues and $Q \in \mathbb{R}^{|V|\times|V|}$ is the eigenvector matrix.

We can equivalently reduce Spectral Clustering to our NRL framework by setting proximity matrix M as the first-order proximity matrix L, network embedding R as the first d columns of eigenvector matrix Q and context embedding C^T as the first d rows of ΛQ^{-1}.

DeepWalk [Perozzi *et al.*, 2014] generates random walks and employs Skip-gram model for representation learning. DeepWalk learns two representations for each vertex and we denote the network embedding and context embedding as matrix $R \in \mathbb{R}^{|V|\times d}$ and $C \in \mathbb{R}^{|V|\times d}$. As proved by [Yang *et al.*, 2015], DeepWalk implicitly factorizes a matrix $M \in \mathbb{R}^{|V|\times|V|}$ into the product of $R \cdot C^T$, where

$$M = \log \frac{A + A^2 + \cdots + A^w}{w}, \tag{2}$$

and w is the window size used in Skip-gram model. Matrix M characterizes the average of the first-order, second-order, $\ldots$, w-order proximities. DeepWalk algorithm approximates high-order proximity by Monte Carlo sampling based on random walk generation without calculating the k-order proximity matrix directly.

To adopt DeepWalk algorithm to our two-step framework, we can simply set proximity matrix $M = f(A) = \frac{A+A^2+\cdots+A^w}{w}$. Note that we omit the log operation in Eq. (2) because they yield competitive performance as reported by [Yang *et al.*, 2015].

GraRep [Cao *et al.*, 2015] accurately calculates k-order proximity matrix A^k for $k = 1, 2 \ldots, K$, computes a specific representation for each k, and concatenates these embeddings. Specifically, GraRep reduces the dimension of k-order proximity matrix A^k for k-order representation via SVD decomposition. In detail, we assume that k-order proximity matrix A^k is factorized into the product of $U\Sigma S$ where $\Sigma \in \mathbb{R}^{|V|\times|V|}$ is a diagonal matrix, $\Sigma_{11} \geq \Sigma_{22} \geq \ldots \Sigma_{|V||V|} \geq 0$ are singular values and $U, S \in \mathbb{R}^{|V|\times|V|}$ are unitary matrices. GraRep defines k-order network embedding and context embedding $R_{\{k\}}, C_{\{k\}} \in \mathbb{R}^{|V|\times d}$ as the first d columns of $U\Sigma^{\frac{1}{2}}$ and $S^T\Sigma^{\frac{1}{2}}$, respectively. The computation of k-order representation $R_{\{k\}}$ naturally follows our framework. However, GraRep cannot efficiently scale to large networks [Grover and Leskovec, 2016]: though the first-order proximity matrix A is sparse, a direct computation of A^k $(k \geq 2)$ takes $O(|V|^2)$ time which is unacceptable for large-scale networks.

It is straightforward that TADW [Yang *et al.*, 2015] and LINE [Tang *et al.*, 2015b] can also be formalized into our framework and the proofs are omitted due to space limitation.

3 Observation and Problem Formalization

By far, we have shown five representative NRL algorithms can be formulated into our two-step framework, i.e. proximity matrix construction and dimension reduction. In this section, we focus on the first step and study how to define a good proximity matrix for NRL. The study of different dimension reduction methods, e.g. SVD decomposition, will be left as future work.

Table 1: Comparisons among three NRL methods.

	SC	DeepWalk	GraRep
Proximity Matrix	L	$\sum_{k=1}^{K} \frac{A^k}{K}$	$A^k, k = 1 \ldots K$
Computation	Accurate	Approximate	Accurate
Scalability	Yes	Yes	No
Performance	Low	Middle	High

We summarize the comparisons among Spectral Clustering (SC), DeepWalk and GraRep in Table 1 and conclude the following observations.

Observation 1 Modeling higher order and accurate proximity matrix can improve the quality of network representation. In other words, NRL could benefit if we explore a polynomial proximity matrix $f(A)$ of a higher degree.

From the development of NRL methods, we can see that DeepWalk outperforms Spectral Clustering because DeepWalk considers higher order proximity matrices and the higher order proximity matrices can provide complementary information for lower order proximity matrices. GraRep outperforms DeepWalk because GraRep accurately calculates the k-order proximity matrix rather than approximating it by Monte Carlo simulation as DeepWalk did.

Observation 2 Accurate computation of high order proximity matrix is not feasible for large-scale networks.

The major drawback of GraRep is the computation complexity of calculating the accurate k-order proximity matrix. In fact, the computation of high order proximity matrix takes $O(|V|^2)$ time and the time complexity of SVD decomposition also increases as k-order proximity matrix gets dense when k grows. In summary, a time complexity of $O(|V|^2)$ is too expensive to handle large-scale networks.

The first observation motivates us to explore higher order proximity matrix in NRL algorithm but the second observation prevents us from an accurate inference of higher order proximity matrices. Therefore we turn out to study the problem that how to learn network embeddings from approximate higher order proximity matrices efficiently. In order

to be more efficient, we aim to use the network representations which encode the information of lower order proximity matrices as our basis to avoid repeated computations. We formalize our problem below.

Problem Formalization Assume that we have normalized adjacency matrix A as the first-order proximity matrix, network embedding R and context embedding C where $R, C \in \mathbb{R}^{|V| \times d}$. Suppose that the embeddings R and C are learned by the above NRL framework which indicates that the product $R \cdot C^T$ approximates a polynomial proximity matrix $f(A)$ of degree K. Our goal is to learn a better representation R' and C' which approximates a polynomial proximity matrix $g(A)$ with higher degree than $f(A)$. Also, the algorithm should be efficient in the linear time of $|V|$. Note that the lower bound of time complexity is $O(|V|d)$ which is the size of embedding matrix R.

4 Approximation Algorithm

In this section, we present a simple, efficient and effective iterative updating algorithm to solve the above problem.

Method Given hyperparameter $\lambda \in (0, \frac{1}{2}]$, normalized adjacency matrix A, we update network embedding R and context embedding C as follows:

$$\begin{aligned} R' &= R + \lambda A \cdot R, \\ C' &= C + \lambda A^T \cdot C. \end{aligned} \tag{3}$$

The time complexity of computing $A \cdot R$ and $A^T \cdot C$ is $O(|V|d)$ because matrix A is sparse and has $O(|V|)$ nonzero entries. Thus the overall time complexity of one iteration of operation (3) is $O(|V|d)$.

Recall that product of previous embedding R and C approximates polynomial proximity matrix $f(A)$ of degree K. Now we prove that the algorithm can learn better embeddings R' and C' where the product $R' \cdot C'^T$ approximates a polynomial proximity matrix $g(A)$ of degree $K + 2$ bounded by matrix infinite norm.

Theorem Given network and context embedding R and C, we suppose that the approximation between $R \cdot C^T$ and proximity matrix $M = f(A)$ is bounded by $r = ||f(A) - R \cdot C^T||_\infty$ and $f(\cdot)$ is a polynomial of degree K. Then the product of updated embeddings R' and C' from Eq. (3) approximates a polynomial $g(A) = f(A) + 2\lambda A f(A) + \lambda^2 A^2 f(A)$ of degree $K + 2$ with approximation bound $r' = (1 + 2\lambda + \lambda^2)r \leq \frac{9}{4}r$.

Proof Assume that $S = f(A) - RC^T$ and thus $r = ||S||_\infty$.

$$\begin{aligned} &||g(A) - R'C'^T||_\infty = ||g(A) - (R + \lambda AR)(C^T + \lambda C^T A)||_\infty \\ &= ||g(A) - RC^T - \lambda ARC^T - \lambda RC^T A - \lambda^2 ARC^T A||_\infty \\ &= ||S + \lambda AS + \lambda SA + \lambda^2 ASA||_\infty \\ &\leq ||S||_\infty + \lambda ||A||_\infty ||S||_\infty + \lambda ||S||_\infty ||A||_\infty + \lambda^2 ||S||_\infty ||A||_\infty^2 \\ &= r + 2\lambda r + \lambda^2 r. \end{aligned} \tag{4}$$

where the second last equality replaces $g(A)$ and $f(A) - RC^T$ by the definitions of $g(A)$ and S and the last equality uses the fact that $||A||_\infty = \max_i \sum_j |A_{ij}| = 1$ because the summation of each row of A equals to 1.

In our experimental settings, we assume that the weight of lower order proximities should be larger than higher order proximities because they are more directly related to the original network. Therefore, given $g(A) = f(A) + 2\lambda A f(A) + \lambda^2 A^2 f(A)$, we have $1 \geq 2\lambda \geq \lambda^2 > 0$ which indicates that $\lambda \in (0, \frac{1}{2}]$. The proof indicates that the updated embedding can implicitly approximate a polynomial $g(A)$ of 2 more degrees within $\frac{9}{4}$ times matrix infinite norm of previous embeddings. QED.

Algorithm The update Eq. (3) can be further generalized in two directions. First we can update embeddings R and C according to Eq. (5):

$$\begin{aligned} R' &= R + \lambda_1 A \cdot R + \lambda_2 A \cdot (A \cdot R), \\ C' &= C + \lambda_1 A^T \cdot C + \lambda_2 A^T \cdot (A^T \cdot C). \end{aligned} \tag{5}$$

The time complexity is still $O(|V|d)$ but Eq. (5) can obtain higher proximity matrix approximation than Eq. (3) in one iteration. More complex update formulas which explores further higher proximities than Eq. (5) can also be applied but we use Eq. (5) in our experiments as a cost-effective choice.

Another direction is that the update equation can be processed for T rounds to obtain higher proximity approximation. However, the approximation bound would grow exponentially as the number of rounds T grows and thus the update cannot be done infinitely. Note that the update operation of R and C are completely independent. Therefore we only need to update network embedding R for NRL. We name our algorithm as Network Embedding Update (NEU). NEU avoids an accurate computation of high-order proximity matrix but can yield network embeddings that actually approximate high-order proximities. Hence our algorithm can improve the quality of network embeddings efficiently. Intuitively, Eq. (3) and (5) allow the learned embeddings to further propagate to their neighbors. Thus proximities of longer distance between vertices will be embedded.

5 Experiments

We evaluate the qualities of network embeddings on two tasks: multi-label classification and link prediction. We perform our Network Embedding Update algorithm (NEU) over the embeddings learned by baseline methods and report both evaluation performance and running time.

5.1 Datasets

We conduct experiments on three publicly available datasets: Cora[1] [Sen *et al.*, 2008], BlogCatalog and Flickr[2] [Tang and Liu, 2011]. We assume that all three datasets are undirected and unweighted networks.

Cora contains $2,708$ machine learning papers drawn from 7 classes and $5,429$ citation links between them. Each paper has exactly one class label. Each paper in Cora dataset also has text information denoted by a $1,433$ dimensional binary vector indicating the presence of the corresponding words.

[1] http://linqs.cs.umd.edu/projects/ /projects/lbc/index.html.

[2] http://socialcomputing.asu.edu/pages/ datasets.

BlogCatalog contains $10,312$ bloggers and $333,983$ social relationships between them. The labels represent the topic interests provided by the bloggers. The network has 39 labels and a blogger may have multiple labels.

Flickr contains $80,513$ users from a photo sharing website and $5,899,882$ friendships between them. The labels represent the group membership of users. The network has 195 labels and a user may have multiple labels.

5.2 Baselines and Experimental Settings

We consider a number of baselines to demonstrate the effectiveness and robustness of NEU algorithm. For all methods and datasets, we set the embedding dimension $d = 128$.

Graph Factorization (GF) simply factorizes the normalized adjacency matrix A via SVD decomposition to reduce dimensions for network embeddings.

Spectral Clustering (SC) [Tang and Liu, 2011] computes the first d eigenvectors of normalized Laplacian matrix as d-dimensional embeddings.

DeepWalk [Perozzi *et al.*, 2014] has three hyperparameters besides embedding dimension d: window size w, random walk length t and walks per vertex γ. As these hyperparameters increase, the number of training samples and running time will increase. We evaluate three groups of hyperparameters of DeepWalk, i.e. a default setting of the authors' implementation DeepWalk$_{low}$ where $w = 5, t = 40, \gamma = 10$, the setting used in node2vec [Grover and Leskovec, 2016] DeepWalk$_{mid}$ where $w = 10, t = 80, \gamma = 10$ and the setting used in the original paper [Perozzi *et al.*, 2014] DeepWalk$_{high}$ where $w = 10, t = 40, \gamma = 80$.

LINE [Tang *et al.*, 2015b] learns two separate network representations LINE$_{1st}$ and LINE$_{2nd}$ respectively. We use default settings for all hyperparameters except the number of total training samples $s = 10^4|V|$ so that LINE has comparable running time against DeepWalk$_{mid}$.

TADW [Yang *et al.*, 2015] incorporates text information into DeepWalk and we add this baseline for Cora dataset.

node2vec [Grover and Leskovec, 2016] is a semi-supervised NRL method and we use the same hyperparameter setting used in their paper: $w = 10, t = 80, \gamma = 10$. We employ a grid search over return parameter and in-out parameter $p, q \in \{0.25, 0.5, 1, 2, 4\}$ for semi-supervised training.

GraRep [Cao *et al.*, 2015] is only used for the smallest dataset Cora due to its inefficiency [Grover and Leskovec, 2016]. We set $K = 5$ and thus GraRep has $128 \times 5 = 640$ dimensions.

Experimental Settings For Spectral Clustering, DeepWalk, LINE, node2vec, we directly use the implementations provided by their authors. We set the hyperparameters of NEU as follows: $\lambda_1 = 0.5, \lambda_2 = 0.25$ for all three datasets, $T = 3$ for Cora and BlogCatalog and $T = 1$ for Flickr. Here λ_1, λ_2 are set empirically following the intuition that lower proximity matrix should have a higher weight and T is set as the maximum iteration before the performance on 10% random validation set begins to drop. In fact, we can simply set $T = 1$ if we have no prior knowledge of downstream tasks. The experiments are executed on a single CPU for the ease of running time comparison and the CPU type is Intel Xeon E5-2620 @ 2.0GHz.

5.3 Multi-label Classification

For multi-label classification task, we randomly select a portion of vertices as training set and leave the rest as test set. We treat network embeddings as vertex features and feed them into a one-vs-rest SVM classifier implemented by LibLinear [Fan *et al.*, 2008] as previous works did [Tang and Liu, 2009; 2011]. We repeat the process for 10 times and report the average Macro-F1 and Micro-F1 score. Since a vertex of Cora dataset has exactly one label, we only report classification accuracy for this dataset. We normalize each dimension of network embeddings so that the $L2$-norm of each dimension equals to 1 before we feed the embeddings into the classifier as suggested by [Ben-Hur and Weston, 2010]. We also perform the same normalization before and after NEU. The experimental results are listed in Table 2, 3 and 4. The numbers in the brackets represent the performances of corresponding methods after processing NEU. "$+0.1$","$+0.3$","$+1$" and "$+8$" in the time column indicate the additional running time of NEU on Cora, BlogCatalog and Flickr dataset, respectively. As an illustrative example on Cora dataset, NEU takes 0.1 second and improves the classification accuracy from 78.1 to 84.4 for network embeddings learned by TADW when labeled ratio is 10%. We exclude node2vec on Flickr as node2vec failed to terminate in 24 hours on this dataset. We **bold** the results when NEU achieves more than 10% relative improvement. We conduct 0.05 level paired t-test and mark all entries that fail to reject the null hypothesis with *.

Table 2: Classification results on Cora dataset.

	% Accuracy			Time (s)
% Labeled Nodes	10%	50%	90%	
GF	50.8 (**68.0**)	61.8 (**77.0**)	64.8 (**77.2**)	4 (+0.1)
SC	55.9 (**68.7**)	70.8 (**79.2**)	72.7 (**80.0**)	1 (+0.1)
DeepWalk$_{low}$	71.3 (76.2)	76.9 (81.6)	78.7 (81.9)	31 (+0.1)
DeepWalk$_{mid}$	68.9 (**76.7**)	76.3 (82.0)	78.8 (84.3)	69 (+0.1)
DeepWalk$_{high}$	68.4 (**76.1**)	74.7 (80.5)	75.4 (81.6)	223 (+0.1)
LINE$_{1st}$	64.8 (70.1)	76.1 (80.9)	78.9 (82.2)	62 (+0.1)
LINE$_{2nd}$	63.3 (**73.3**)	73.4 (80.1)	75.6 (80.3)	67 (+0.1)
node2vec	76.9 (77.5)	81.0 (81.6)	81.4 (81.9)	56 (+0.1)
TADW	78.1 (84.4)	83.1 (86.6)	82.4 (87.7)	2 (+0.1)
GraRep	70.8 (76.9)	78.9 (82.8)	81.8 (84.0)	67 (+0.3)

5.4 Link Prediction

For the purpose of link prediction, we need to score each pair of vertices given their embeddings. For each pair of network embedding r_i and r_j, we try three scoring functions, i.e. cosine similarity $\frac{r_i \cdot r_j}{||r_i||_2||r_j||_2}$, inner product $r_i \cdot r_j$ and inverse $L2$-distance $1/||r_i - r_j||_2$. We use AUC value [Hanley and McNeil, 1982] which indicates the probability that the score of an unobserved link is higher than that of a nonexistent link as our evaluation metric and select the scoring function with best performance for each baseline. We remove 20% edges of Cora, 50% of BlogCatalog and Flickr as test set and use the remaining links to train network embeddings. We also add three commonly used link prediction baselines for comparison: Common Neighbors (CN), Jaccard Index and Salton Index [Salton and McGill, 1986]. We only report the best performance for DeepWalk$\in\{$DeepWalk$_{low}$, DeepWalk$_{mid}$,

Table 3: Classification results on BlogCatalog dataset.

	% Macro-F1			% Micro-F1			Time (s)
% Labeled Nodes	1%	5%	9%	1%	5%	9%	
GF	6.6 (**7.9**)	9.8 (**11.3**)	10.3 (**12.2**)	17.0 (**19.6**)	22.2 (**25.0**)	23.7 (**26.7**)	19 (+1)
SC	8.4 (**9.3**)	13.1 (**14.8**)	14.5 (**17.0**)	19.4 (20.3)	26.9 (28.1)	29.0 (31.0)	10 (+1)
DeepWalk$_{low}$	11.3 (12.4)	15.9 (17.4)	17.1 (18.6)	24.5 (26.4)	31.0 (33.4)	32.8 (35.1)	100 (+1)
DeepWalk$_{mid}$	11.2 (**13.3**)	16.9 (**19.2**)	18.4 (**20.8**)	24.0 (**27.1**)	31.0 (33.8)	32.8 (35.7)	225 (+1)
DeepWalk$_{high}$	12.4 (13.6)	18.3 (20.1)	20.4 (22.0)	24.9 (26.4)	31.5 (33.7)	33.7 (35.9)	935 (+1)
LINE$_{1st}$	11.1 (12.2)	16.6 (18.3)	18.6 (20.1)	23.1 (24.7)	29.3 (31.6)	31.8 (33.5)	241 (+1)
LINE$_{2nd}$	10.3 (11.2)	15.0 (16.8)	16.5 (**18.3**)	21.5 (**25.0**)	27.9 (**31.6**)	30.0 (**33.6**)	244 (+1)
node2vec	12.5 (13.0)	19.2 (19.8)	21.9 (22.5)	25.0 (27.0)	31.9 (34.5)	35.1 (37.2)	454 (+1)

Table 4: Classification results on Flickr dataset.

	% Macro-F1			% Micro-F1			Time (s)
% Labeled Nodes	1%	5%	9%	1%	5%	9%	
GF	4.3 (**5.2**)	4.9 (**5.4**)	5.0 (5.4)	21.1 (21.8)	22.0 (23.1)	21.7 (23.4)	241 (+8)
SC	8.6 (**10.9**)	11.6 (**14.3**)	12.3 (**15.0**)	24.1 (**29.2**)	27.5 (**34.1**)	28.3 (**34.7**)	102 (+8)
DeepWalk$_{low}$	7.8 (**8.6**)	10.1 (**11.6**)	10.4 (**12.1**)	28.5 (**31.4**)	30.9 (33.5)	31.3 (33.8)	1,449 (+8)
DeepWalk$_{mid}$	8.8 (**9.9**)	12.3 (**14.3**)	13.2 (**15.1**)	29.5 (31.9)	32.4 (35.1)	33.0 (35.4)	2,282 (+8)
DeepWalk$_{high}$	10.5 (**11.6**)	17.1 (17.8)	19.1 (19.8)	31.8 (33.1)	36.3 (36.7)	37.3 (37.6)	9,292 (+8)
LINE$_{1st}$	10.3 (10.7)	16.0 (16.6)	17.6 (18.2)	32.0 (32.7)	35.9 (36.4)	36.8 (37.2)	2,664 (+8)
LINE$_{2nd}$	7.8 (8.5)	13.1 (13.5)	14.7 (15.2)	30.0 (31.0)	34.2 (34.4)	35.1 (35.2)*	2,740 (+8)

DeepWalk$_{high}\}$ and LINE$\in\{$LINE$_{1st}$, LINE$_{2nd}\}$ and omit the results of node2vec as it only yields comparable and even worse performance than the best performed DeepWalk. The experimental results are shown in Figure 1. For each dataset, the three leftmost columns represent the traditional link prediction baselines. Then each pair of columns stands for an NRL method and its performance after NEU.

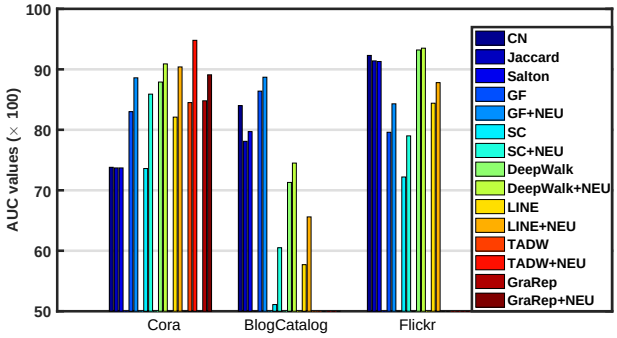

Figure 1: Experimental results on link prediction.

5.5 Experimental Results Analysis

We have four main observations over the experimental results of two evaluation tasks:

(1) NEU consistently and significantly improves the performance of various network embeddings using almost negligible running time on both evaluation tasks. The absolute improvements on Flickr are not as significant as that on Cora and BlogCatalog because Flickr dataset has an average degree of 147 which is much denser than Cora and BlogCatalog and thus the impact of higher order proximity information is diluted by rich first-order proximity information. But for Cora dataset where the average degree is 4, NEU has very significant improvement as high-order proximity plays an important role for sparse networks.

(2) NEU facilitates NRL method to converge fast and stably. We can see that the performances of DeepWalk$_{low}$+NEU and DeepWalk$_{mid}$+NEU are comparable and even better than that of DeepWalk$_{mid}$ and DeepWalk$_{high}$ respectively and the former ones use much less time. Also, DeepWalk encounters the overfitting problem on Cora dataset as the classification accuracy drops when hyperparameters increase. However, the performances of DeepWalk+NEU are stable and robust.

(3) NEU also works for NRL algorithms which don't follow our two-step NRL framework, i.e. node2vec. This observation demonstrates the effectiveness and robustness of NEU.

6 Conclusion

In this paper, we propose a unified NRL framework which covers a number of existing NRL methods. We analyze the first step of the framework, i.e. proximity matrix construction, compare the proximity matrices used in different NRL algorithms and conclude that we can learn better network embeddings if higher order proximities are encoded into the proximity matrix. Then we present Network Embedding Update (NEU) algorithm to improve the performance of any given network embeddings by implicitly approximating higher order proximity matrix. The running time of NEU is almost negligible and the improvement over baseline methods are consistent and significant.

Acknowledgements

This work is supported by the National 973 Program (No.2014CB340501) and the Major Project of the National Social Science Foundation of China (No.13&ZD190).

References

[Akoglu *et al.*, 2015] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 2015.

[Ben-Hur and Weston, 2010] Asa Ben-Hur and Jason Weston. A users guide to support vector machines. *Data mining techniques for the life sciences*, 2010.

[Bengio *et al.*, 2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on PAMI*, 2013.

[Cao *et al.*, 2015] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of CIKM*, 2015.

[Chang *et al.*, 2015] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. Heterogeneous network embedding via deep architectures. In *Proceedings of KDD*, 2015.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *JMLR*, 2008.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. 2016.

[Hanley and McNeil, 1982] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 1982.

[Huang and Mamoulis, 2017] Zhipeng Huang and Nikos Mamoulis. Heterogeneous information network embedding for meta path based proximity. *arXiv preprint arXiv:1701.05291*, 2017.

[Huang *et al.*, 2017] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *Proceedings of WSDM*, 2017.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*, 2017.

[Ley, 2002] Michael Ley. The dblp computer science bibliography: Evolution, research issues, perspectives. In *International symposium on string processing and information retrieval*, 2002.

[Li *et al.*, 2016] Juzheng Li, Jun Zhu, and Bo Zhang. Discriminative deep random walk for network classification. In *Proceedings of ACL*, 2016.

[Liben-Nowell and Kleinberg, 2007] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology*, 2007.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, 2013.

[Ou *et al.*, 2016] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of KDD*, 2016.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of KDD*, 2014.

[Salton and McGill, 1986] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 2008.

[Tang and Liu, 2009] Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *Proceedings of KDD*, 2009.

[Tang and Liu, 2011] Lei Tang and Huan Liu. Leveraging social media networks for classification. *Proceedings of KDD*, 2011.

[Tang *et al.*, 2015a] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of KDD*, 2015.

[Tang *et al.*, 2015b] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of WWW*, 2015.

[Tu *et al.*, 2014] Cunchao Tu, Zhiyuan Liu, and Maosong Sun. Inferring correspondences from multiple sources for microblog user tags. In *Proceedings of SMP*, 2014.

[Tu *et al.*, 2016] Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. Max-margin deepwalk: discriminative learning of network representation. In *Proceedings of IJCAI*, 2016.

[Wang *et al.*, 2016] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of KDD*, 2016.

[Wang *et al.*, 2017] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. *Proceedings of AAAI*, 2017.

[Xu *et al.*, 2017] Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S Yu. Embedding of embedding (eoe): Joint embedding for coupled heterogeneous networks. In *Proceedings of WSDM*, 2017.

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

[Yang *et al.*, 2016] Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of ICML*, 2016.