This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3148284, IEEE Transactions on Knowledge and Data Engineering



# Learning Regularized Noise Contrastive Estimation for Robust Network Embedding

Hao Xiong, Junchi Yan *Senior Member, IEEE*, and Zengfeng Huang

**Abstract**—Skip-gram models are popular in large-scale network embedding for their cost-effectiveness. The objectives of many skip-gram based methods relate to the word2vec model which closely relates to Noise Contrastive Estimation (NCE). Among existing embedding methods, the differences mostly lie in how the node neighborhood is modeled e.g. by different ways of random walk, which leads to different learning strategies. Orthogonal to these efforts, we take a unified view that the NCE based methods commonly involve two basic NCE components in the learning objective. This perspective allows a natural generalization of the objectives by taking different forms of scoring function in the NCE components. We theoretically analyze how the vanilla NCE-based objectives suffer from the slow convergence speed and challenge in first-/second-order proximity preservation. We also prove the fundamental difficulty for NCE methods to capture non-linearity of complex networks. To mitigate such issues, we devise a general distance-based term added to the used NCE term, inspired by its physical meaning. The distance functions include Wasserstein-$k$ distance and Laplacian/Gaussian kernel functions, with relatively little additional time overhead. The effectiveness of our approach is verified both by prototype examples as well as real-world datasets, for the task of node classification and network reconstruction.

**Index Terms**—Network embedding, Noise-contrastive estimation, Score function

## 1 Introduction and Motivation

Graph-like data is ubiquitous such as social networks, knowledge graphs, molecule, transaction networks, etc. Compared with traditional vector-like data as readily handled by a wide range of existing learning methods for classification, regression and ranking, graph can be more complex and one effective way of reusing traditional learning methods for vector-like data is to embed either node or even the whole network into vector representation [1], [2].

Network embedding or namely node embedding has been a hot topic with a line of recent methods, including those based on deep learning [3], matrix factorization [4], [5], [6], [7], graph neural network [8], [9], and skip-gram [10], [11], [12], [13], [14], [15], [16], etc.

Among them, skip-gram based methods have received wide interests for their cost-effectiveness and scalability. Skip-gram is derived from word embedding in natural language processing [17], [18], where each word is represented by an embedding vector through a lookup table. Its basic idea is to build an embedding model which can help predict the nearby words for each word in a sentence. When skip-gram is transferred to network embedding, the motivation becomes to learn a good node embedding model that is able to predict a node's neighbors according to node embedding [10], [11]. The neighborhood relation probability is usually modeled by softmax that requires high computation. As alternatives, hierarchical softmax [17] and negative sampling based on noise contrastive estimation (NCE) [19] are two popular techniques to solve the problem. The NCE-based one is often more welcomed in practice [11], [12], [14], [20] due to its empirical strong performance.

Specifically, NCE-based embedding models aim to maximize the sigmoid of the inner-product between node embeddings and hidden embeddings (also a.k.a. context embedding in [11]) for positive samples, while they do the opposite for negative samples. In each iteration of stochastic gradient descent (SGD), for positive samples, the node embedding will be added by a little step of the same direction of the hidden embedding, and so will the hidden embedding. While for negative samples, the little step will be in the opposite direction of the hidden embedding. In this way, the node embedding and hidden embedding will be moved closer for positive samples while further away for negative samples. However, as we prove in Theorem 1, the convergence speed of vanilla NCE can be quite slow.

One of the consensuses of network embedding is that a good model should preserve proximity between nodes [3], [11], [21], which is also one of the targets of NCE-based embedding models [11]. To be more specific, nodes that are connected (sharing first-order proximity), and nodes that have common neighbors (sharing second-order proximity) are expected to have similar embedding. As mentioned in the last paragraph, NCE-based embedding models achieve to preserve first-order proximity by moving the embedding of connected nodes (positive samples) together. We theoretically give the achievable optimal solutions by iterations of SGD in Theorem 1, by which we find two existing problems of NCE: 1) For positive samples, NCE tends to make the embedding vectors longer, but not make them more similar, which means that the loss of NCE is not suitable enough for proximity preservation; 2) Gradient vanishing may happen in the later stage of training, preventing the embeddings from going more similar. We give a toy example in Fig. 1 to illustrate how the problems may happen. Moreover, we also

*H. Xiong and J. Yan are with Department of Computer Science and Engineering, MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, China. J. Yan is also affiliated with Zhejiang Lab, Hangzhou, China. Z. Huang is with School of Data Science, Fudan University, Shanghai, China.*
*J. Yan is the correspondence author.*
*E-mail: {taxuexh,yanjunchi}@sjtu.edu.cn, huangzf@fudan.edu.cn*

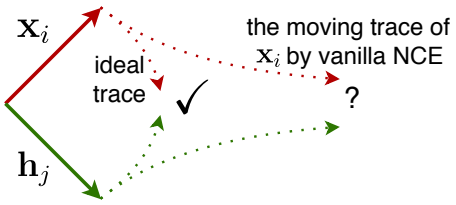



Fig. 1: Illustration of the embedding's updating behavior. Here $\mathbf{x}_i$ and $\mathbf{h}_j$ are the embedding of node $i$ and hidden embedding of node $j$, respectively. In each iteration by NCE, embedding $\mathbf{x}_i$ will add a little step that is of the same direction of $\mathbf{h}_j$, resulting in the upper red dashed line. It has two drawbacks: 1) After some iterations, the convergence will become extremely hard due to gradient vanishing; 2) To make $\mathbf{x}_i$ and $\mathbf{h}_j$ similar enough, $\|\mathbf{x}_i\|$ will become very large. In comparison, we ideally hope the moving trace of the embedding $\mathbf{x}_i$ should be like the lower red dashed line, which makes $\mathbf{x}_i$ and $\mathbf{h}_j$ similar in a straight way.

clarify that the capability of NCE-based embedding models to preserve second-order proximity is only guaranteed with enough neighbors.

Apart from the problems in proximity preserving, we will show in Theorem 3 that a specific type of NCE (referred as NCE-h in the following paper) is more likely to embed nodes in a linear distribution. However, real-world networks are usually of high non-linearity [22]. That makes it very hard to express complex network structures for the NCE-based embedding methods.

The finding inspires us to try to solve the above-mentioned problems by adding some other more advanced distance functions to the loss of vanilla NCE, such as the advanced Wasserstein-$k$ distance and Gaussian/Laplacian kernel functions. The hope is to better maintain the structure and community property after embedding by more effective distance modeling. This idea is orthogonal to the majority of existing skip-gram methods that are exploring different techniques for neighborhood design while keeping the objective unchanged. In fact, a more wide neighborhood can usually improve the performance at the cost of quite more computational overhead.

In a nutshell, the highlight of this paper are as follows:

1) We take a Noise Contrastive Estimation perspective on the majority of skip-gram based models and formally identify two building blocks $O^h_{nce}$ (Eq. 2), $O^x_{nce}$ (Eq. 3)[1] used in the objectives (see Table 1). Orthogonal to the efforts on neighborhood modeling instead of objective re-design as mostly done by the previous methods [12], [13], this perspective allows us to generalize the two basic components which can be readily applied to existing embedding methods.

2) We theoretically study the NCE objective, including analyzing its convergence speed and the achievable optimal solutions, analyzing its capability to preserve nodes' proximity, and the special linearity caused by the negative samples. Based on the observation, we propose to add distance terms who have a common form of gradients to the objective. The approach is also theoretically proved to bear a few nice abilities, including faster convergence speed, stronger ability to preserve nodes' proximity, and better expressiveness on non-linear structures of networks.

3) Experimental results show the improved performance by adding the distance term to existing NCE-based methods over different tasks significantly. By our techniques, even the simplest NCE-based embedding i.e. based on the vanilla $O^h_{nce}$, $O^x_{nce}$ with only direct neighbors can outperform most state-of-the-art methods such as [6], [14] given in Table 1.

1. $O^x_{nce}$ means the objective only contains node embeddings while $O^h_{nce}$ indicates an additional utilization of hidden embeddings.

TABLE 1: Noise Contrastive Estimation perspective on skip-gram based network embedding methods. The two vanilla NCE components $O^h_{nce}$ (Eq. 2), $O^x_{nce}$ (Eq. 3) both consist of an inner product based score term for positive samples plus a noise term for negative ones. They serve as building blocks for the listed embedding methods and can be generalized by a function $F$. This paper sets $F(X) = (X + dist)$ which is motivated for effective gradient descent updating beyond the inner product loss, where $dist$ is a distance function. The bottom two vanilla ones will also be studied in the paper.

| Methods | Objective | Obj. w/ dist. | Neighbor |
|---|---|---|---|
| LINE-1st [11] | $O^x_{nce}$ | $F(O^x_{nce})$ | direct |
| LINE-2nd [11] | $O^h_{nce}$ | $F(O^h_{nce})$ | direct |
| node2vec [12] | $O^h_{nce}$ | $F(O^h_{nce})$ | biased RW |
| struc2vec [13] | $O^h_{nce}$ | $F(O^h_{nce})$ | structural RW |
| VERSE [14] | $O^x_{nce}$ | $F(O^x_{nce})$ | PPR |

*Notation:* ***PPR****: Personalized PageRank.* ***Biased random walk*** *uses two parameters to perform breadth-/depth-first search simultaneously.* ***Structural random walk*** *refers to random walk according to nodes' structural similarity.* ***Direct neighbor*** *in this paper indicates the plainest case that only nodes with 1-hop edges are possible positive samples by sampling.*

## 2 Related Work

Existing works can be divided into NCE-based, matrix factorization-based, and deep network-based ones.

**NCE-based network embedding.** Many recent popular embedding methods relate to the idea of negative sampling (early use in word2vec [17]) for more efficient learning in place of the intractable softmax for large-scale networks. Negative sampling can be regarded as a simplified version of Noise Contrastive Estimation (NCE), which is first proposed in [19] and later used in neural probabilistic language models [23]. In general, the NCE model aims to distinguish the positive samples from background noise.

We show that many skip-gram embedding models are essentially based on NCE. For instance, though in the original paper, DeepWalk [10] models the random-walk sequence by hierarchical softmax, it in fact can also be replaced by negative sampling. The uniform random-walk preserves both local and global structure. On this basis, node2vec [12] introduces biased second-order random-walk to preserve both breadth-first and depth-first structure. It adopts $O^h_{nce}$ (see Eq. 2) as the objective. LINE [11] proposes to model first-order proximity by KL-divergence whose objective is $O^x_{nce}$ (see Eq. 3) and model second-order proximity by $O^h_{nce}$.

TABLE 2: Comparison between RNS [24] and our approach. The motivations, analysis and the approaches are very different and mostly orthogonal to each other.

| | RNS (Robust Negative Sampling) | Ours (Regularized NCE Embedding) |
|---|---|---|
| Motivation | limitations caused by popular neighbor problems | defects in proximity preservation;; limitation of linearly distributed embeddings (NCE-h) |
| Theoretical Analysis | optimal solutions of skip-gram objectives; optimal solutions of skip-gram objectives by negative sampling; | optimal solutions by SGD iterations; convergence speed by SGD iterations; capability to preserve proximity; high linearity of NCE-h embeddings; |
| Approach | an adaptive negative sampler; embedding normalization; | distance regularizers over positive node pairs |
| Input | undirected & unweighted networks | directed & weighted networks; node sequences; |

It only considers 1-hop neighbors while it also yields impressive performance. struc2vec [13] introduces structural random-walk, which adds nodes with similar structure to neighborhoods to preserve structural identity information. It feeds the random-walk sequences into node2vec to learn node embedding. Recently, VERSE [14] proposes to instantiate proximity functions (e.g. Personalized PageRank) and models the proximities by KL-divergence similar to [11]. Specifically, it adopts $O^x_{nce}$ as its objective. Hence one can find existing methods mainly differ in how the neighbor is expanded while the objectives are similar (see Table 1).

Recently, it is pointed out [24] that skip-gram embedding by negative sampling suffers from the so-called 'Popular Neighbor Problem' which may cause poor performance for embedding of high degree nodes and preservation of first-order proximity. While the technique by RNS [24] is relatively trivial: tuning the embedding penalty and the adaptive negative sampler, which cannot guarantee an effective solution. Moreover, due to its unweighted/undirected design, RNS has difficulties in applying other advanced embedding methods that are based on node sequences such as [10], [12], [13]. The main differences between RNS and this paper are listed in Table 2.

**Matrix factorization-based network embedding.** The authors in [7], [25] treat the network embedding task as sparse matrix factorization and [25] proposes to unify a number of shallow models within a matrix factorization framework. HOPE [4] preserves high-order proximity as well as asymmetric transitivity in directed network. AROPE [5] models arbitrary-order proximity by SVD. Most recently, STRAP [26] combines backward push algorithm and Personalized PageRank (PPR) as its transpose proximities, and do matrix factorization under the sparsity constraints. ProNE [6] adopts a two-stage learning procedure, where the first step is sparse matrix factorization and the second is spectral propagation.

**Deep learning-based network embedding.** There are emerging deep network based methods e.g. SDNE [3], which has difficulty in dealing with large-scale networks. GNN-based embedding methods [8] are also loosely related. They are in general more focused on extracting information from node attributes, rather than from node structures as emphasized in network embedding.

## 3 Analysis to the NCE Framework

In this section, we theoretically analyze the defects in the vanilla NCE framework. First, in Sec. 3.1, we give basic background on NCE-based network embedding models including two forms. Then in Sec. 3.2 and Sec. 3.3, we discuss the defects from two aspects, proximity preservation and the limitation of high linearity. Finally in Sec. 3.4, we summarize the defects and give a general sight of regularized NCE that is introduced in the next section. The outline of this section and the following Sec. 4 is shown in Fig. 2. The main notations in this paper are listed in Table 3.

TABLE 3: Main notations and description used in this paper.

| General parameters | |
|---|---|
| $G = (\mathcal{V}, \mathcal{E}, w)$ | network where $\mathcal{V}$ is vertex set, $\mathcal{E}$ is edge set, and $w$ is edge weight |
| $N$ | number of nodes in network $G$ |
| $O^z_{nce}(i,j)$ | objective by NCE-based embedding for a positive sample $(i,j)$ |
| $\mathbb{P}_n$ | distribution of negative samples |
| $L^z$ | the loss function of NCE-based embedding |
| $\mathbf{x}_i$, $\mathbf{h}_j$ | node embedding for node $i$ and hidden embedding for node $j$ |
| $y_{ij}$ | inner-product of node $i$'s node embedding and node $j$'s hidden embedding |
| $.^{(t)}$ | a parameter/solution at $t$-th iteration |
| $.^*$ | a parameter/solution's optimal point |
| $p$ | the probability that an edge forms in a random graph, used in Theorem 3 |
| $A_1$, $A_2$ | two defined events, used in Theorem 3 and Remark 2 |
| $d_i^{in}$, $d_i^{out}$ | in-/out-degree of node $i$, used in Lemma 1 and Theorem 3 |
| $\lambda_{dis}$ | the weight of $\mathbf{x}_i - \mathbf{h}_j$ in the gradient by $O^z_{dis}$, a distance-function-specific scalar variable |
| **Hyper-parameters** | |
| $d$ | embedding dimension |
| $K$ | number of negative samples for each positive sample |
| $\beta$ | weight of the distance term |
| $\gamma$ | parameter of kernel functions |
| $\eta$ | learning rate |

### 3.1 Preliminaries for NCE-based Embedding

Define graph $G = (\mathcal{V}, \mathcal{E}, w)$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ is a set of nodes, $\mathcal{E} = \{(i,j)\}$ is a set of edges, and $w_{ij}$ is the weight of directed edge $(i,j)$. We define $(i,j)$ 'positive' when $w_{ij} > 0$. Also, note that $w_{ij}$ does not have to indicate the connection strengths. It can be re-defined by external algorithms (such as PPR, second-order transition probability, etc., as listed in Table 1). By NCE-based embedding, each node $i$ has node embedding $\mathbf{x}_i$ and hidden embedding $\mathbf{h}_i$ ($\mathbf{x}_i, \mathbf{h}_i \in \mathbb{R}^d$). To predict node $j$ by the context node $i$, the probability $P(j|i)$ is usually defined in softmax as the following equation, where $s(j|i)$ is usually instantiated as inner-product $\mathbf{x}_i^\top \mathbf{h}_j$ [11], [14]:

$$P(j|i) = \frac{\exp(s(j|i))}{\sum_{j'=1}^{N} \exp(s(j'|i))}.$$

Since the full softmax can be intractable for large-scale networks, Noise Contrastive Estimation (NCE) is adopted as an alternative, which aims to distinguish positive data samples from the noise. On the one hand, NCE treats the normalization constants in the softmax as parameters. On the other hand, NCE reduces the number of noise samples

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3148284, IEEE Transactions on Knowledge and Data Engineering



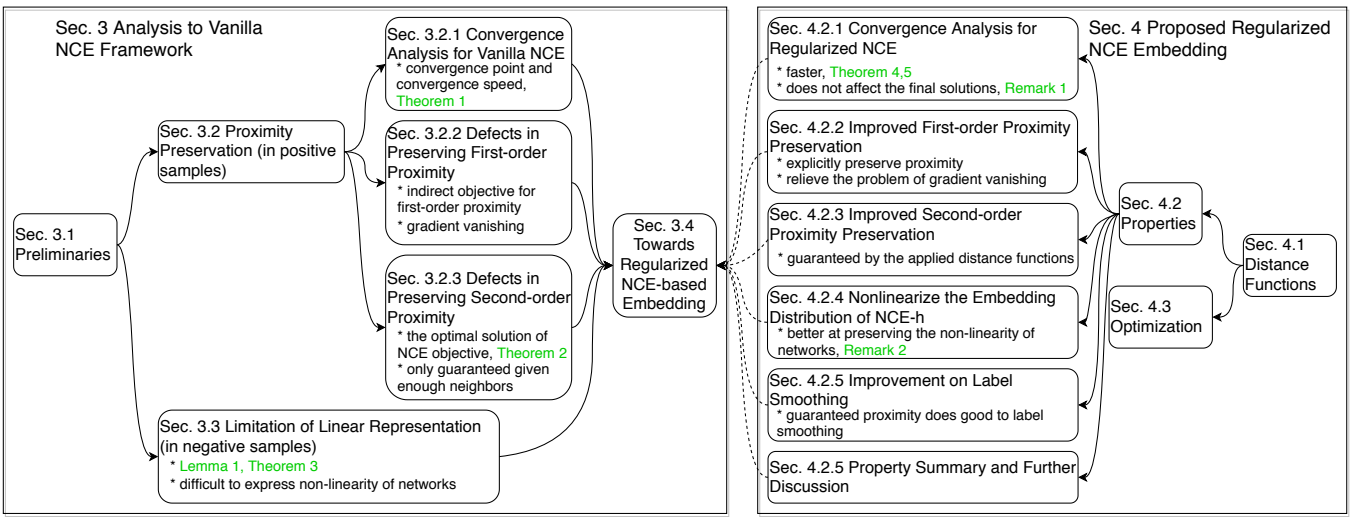



Fig. 2: The outline of the main parts of this paper. In Sec. 3, we discuss the possible defects of the NCE framework in two aspects: proximity preservation in Sec. 3.2 and limited expressiveness of linear embedding in Sec. 3.3. In Sec. 4, we propose the regularized NCE by adding a distance term to vanilla NCE whose properties are analyzed in Sec. 4.2.

to $K$. Denote the modified conditional probability as $\widetilde{P}(j|i)$, then the objective for $(i,j)$ given by NCE is defined as :

$$O(i,j) = -\log \widetilde{P}(j|i) - \underbrace{K\mathbb{E}_{j'\sim\mathbb{P}_n} \log(1-\widetilde{P}(j'|i))}_{\text{noise modeling}}, \quad (1)$$

where $\mathbb{P}_n$ is the context-independent noise distribution. word2vec simplifies the conditional probability as $\widetilde{P}(j|i) = \sigma(\mathbf{x}_i^\top \mathbf{h}_j)$, and proposes negative sampling to optimize the objective which finally becomes Eq. 2 [17]. In advanced NCE-based network embedding methods, a positive data sample can be a pair of connected nodes by edge in $\mathcal{E}$ [11], or a positive node pair in node sequences [12], or generated by specific algorithms [14]. For a positive sample $(i,j)$, negative sampling breaks the connection and substitutes node $j$ with a negative node $j'$ generated by $\mathbb{P}_n$. When $(i,j)$ is sampled as a positive node pair and $(i,j')$ as negative ones, the objective consists of a score function term for the positive sample, plus a noise modeling term for negative samples, which is defined as [11]:

$$O^h_{nce}(i,j) = \underbrace{-\log \sigma(\mathbf{x}_i^\top \mathbf{h}_j)}_{\substack{\text{score function for} \\ \text{positive sample}}} \underbrace{- \sum_{k=1}^{K} \mathbb{E}_{j'\sim\mathbb{P}_n} \log \sigma(-\mathbf{x}_i^\top \mathbf{h}_{j'})}_{\text{noise modeling for negative samples}}, \quad (2)$$

where the score function term usually uses a sigmoid function $\sigma(x) = 1/(1+\exp(-x))$. When hidden embedding $\mathbf{h}$ is set the same as node embedding $\mathbf{x}$, the objective becomes Eq. 3. It is derived from KL-divergence [11] and shares a similar NCE-based form with Eq. 2. It helps preserve first-order proximity when direct neighbors connected by edges are considered (LINE-1st [11]). In the following parts of the paper, we name Eq. 2 as NCE-h and Eq. 3 as NCE-x.

$$O^x_{nce}(i,j) = -\log \sigma(\mathbf{x}_i^\top \mathbf{x}_j) - \sum_{k=1}^{K} \mathbb{E}_{j'\sim\mathbb{P}_n} \log \sigma(-\mathbf{x}_i^\top \mathbf{x}_{j'}). \quad (3)$$

Then the loss of NCE-based embedding is defined as:

$$L^z = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} O^z_{nce}(i,j), z \in \{x,h\}. \quad (4)$$

Hence the optimization objective is to find the embedding that minimize the loss $L^z$:

$$\min_{\{\mathbf{x}_i,\mathbf{h}_i|i\in\mathcal{V}\}} L^z. \quad (5)$$

Recall in Table 1, the two terms, Eq. 2 and Eq. 3, serve as building blocks in lots of skip-gram based embedding methods.

**Gradient analysis.** We then analyze the gradients for objectives Eq. 2 and Eq. 3. Suppose $label = 1$ for positive samples and $label = 0$ otherwise, then the gradient of the updated embedding $\mathbf{x}_i$ is:

$$\frac{\partial O^z_{nce}(i,j)}{\partial \mathbf{x}_i} = \begin{cases} \left(\sigma(\mathbf{x}_i^\top \mathbf{h}_j) - label\right)\mathbf{h}_j & z = h \\ \left(\sigma(\mathbf{x}_i^\top \mathbf{x}_j) - label\right)\mathbf{x}_j & z = x, i \neq j \\ 2\left(\sigma(\mathbf{x}_i^\top \mathbf{x}_i) - label\right)\mathbf{x}_i & z = x, i = j \end{cases}. \quad (6)$$

## 3.2 Proximity Preservation of NCE-based Embedding

In the embedding space, nodes' proximity is measured by the similarity between embeddings. In this part, we give a detailed retrospection over the capability of NCE-based embedding models to preserve nodes' proximity. We first analyze the convergence in Sec. 3.2.1, then the first-/second-order proximity preservation in Sec. 3.2.2 and Sec. 3.2.3. We see that the gradients of $O^h_{nce}$ and $O^x_{nce}$ share a similar form (Eq. 6). So in general, we provide analysis mainly for $O^h_{nce}$ in the following parts of this paper.

### 3.2.1 Convergence Analysis

We theoretically show the convergence of NCE-based model in the following Theorem 1, by analyzing the SGD iterations over a single positive sample. We find that the vanilla NCE-based embedding models can hardly converge at the

optimal solution by SGD. Specifically, we take $(i, j)$ as the positive sample, over which SGD iterations over the embedding $\mathbf{x}_i$ and $\mathbf{h}_j$ are conducted. After infinite iterations, the length of $\mathbf{x}_i$ and $\mathbf{h}_j$ will grow to infinity, and the distance $\|\mathbf{x}_i - \mathbf{h}_j\|$ will decay to zero. Moreover, both of $\mathbf{x}_i$ and $\mathbf{h}_j$ will converge at the direction of $\mathbf{x}_i^{(0)} + \mathbf{h}_j^{(0)}$. However, the convergence speed is very slow (Eq. 9, Eq. 11), making the theoretically optimal solution $\|\mathbf{x}_i - \mathbf{h}_j\| = 0$ unreachable.

**Theorem 1.** ***(For both NCE-h/x)*** *Given an edge $(i, j)$, we update the embedding of nodes $i$ and $j$ based on the NCE objective (Eq. 2 for NCE-h and Eq. 3 for NCE-x), taking $(i, j)$ as the positive sample and neglecting negative samples. Specifically, by the gradients given in Eq. 6, the updating rules are as the follows:*

$$\begin{aligned}\mathbf{x}_i^{(t+1)} &\leftarrow \mathbf{x}_i^{(t)} + \eta\sigma\Big(-\mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)}\Big)\mathbf{h}_j^{(t)},\\ \mathbf{h}_j^{(t+1)} &\leftarrow \mathbf{h}_j^{(t)} + \eta\sigma\Big(-\mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)}\Big)\mathbf{x}_i^{(t)},\end{aligned} \tag{7}$$

*where the superscript $^{(t)}$ denotes the time step, $\eta$ is the learning rate. Supposing that embeddings $\mathbf{x}_i^{(0)}$ and $\mathbf{h}_j^{(0)}$ is initialized with random values by Gaussian distribution, we have the following properties:*

***Property 1.*** *The limit of $\mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)}$ is:*

$$\lim_{t\to\infty} \mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)} = +\infty. \tag{8}$$

***Property 2.*** *The increasing speed of $\mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)}$ follows:*

$$\int_{\mathbf{x}_i^{(0)\top}\mathbf{h}_j^{(0)}}^{\mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)}} \frac{d(\mathbf{x}_i^\top\mathbf{h}_j)}{\sigma(-\mathbf{x}_i^\top\mathbf{h}_j)\mathbf{x}_i^\top\mathbf{h}_j} \sim \int_{\eta^{(0)}}^{\eta^{(t)}} d\eta. \tag{9}$$

***Property 3.*** *The limit of $\|\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)}\|$ is:*

$$\lim_{t\to\infty} \|\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)}\| = 0. \tag{10}$$

***Property 4.*** *The decaying speed of $\|\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)}\|^2$:*

$$\int_{\|\mathbf{x}_i^{(0)}-\mathbf{h}_j^{(0)}\|^2}^{\|\mathbf{x}_i^{(t)}-\mathbf{h}_j^{(t)}\|^2} \frac{d(\|\mathbf{x}_i - \mathbf{h}_j\|^2)}{\|\mathbf{x}_i - \mathbf{h}_j\|^2\sigma(-\mathbf{x}_i^\top\mathbf{h}_j)} \sim -\int_{\eta^{(0)}}^{\eta^{(t)}} d\eta. \tag{11}$$

***Property 5.*** *The limits of normalized embeddings are:*

$$\lim_{t\to\infty} \frac{\mathbf{x}_i^{(t)}}{\|\mathbf{x}_i^{(t)}\|} = \lim_{t\to\infty} \frac{\mathbf{h}_j^{(t)}}{\|\mathbf{h}_j^{(t)}\|} = \frac{\mathbf{x}_i^{(0)} + \mathbf{h}_j^{(0)}}{\|\mathbf{x}_i^{(0)} + \mathbf{h}_j^{(0)}\|}. \tag{12}$$

*Proof.* We prove the five properties as the follows:

**Proof for Property 1.** Considering that the learning rate $\eta \to 0$, we write the updating rules Eq. 7 in the differential form:

$$\begin{aligned}d\mathbf{x}_i &= \sigma(-\mathbf{x}_i^\top\mathbf{h}_j)\mathbf{h}_j d\eta,\\ d\mathbf{h}_j &= \sigma(-\mathbf{x}_i^\top\mathbf{h}_j)\mathbf{x}_i d\eta,\end{aligned}$$

where $d\eta$ denotes the learning step. We define $y_{ij} = \mathbf{x}_i^\top\mathbf{h}_j$, then we have:

$$\begin{aligned}dy_{ij} &= \mathbf{h}_j^\top d\mathbf{x}_i + \mathbf{x}_i^\top d\mathbf{h}_j\\ &= \sigma(-y_{ij})(\mathbf{h}_j^\top\mathbf{h}_j + \mathbf{x}_i^\top\mathbf{x}_i)d\eta\\ &\geq 2\sigma(-y_{ij})y_{ij}d\eta.\end{aligned} \tag{13}$$

After some transformation and do integration, we have:

$$\int_{y_{ij}^{(0)}}^{y_{ij}^{(t)}} \frac{1}{y_{ij}\sigma(-y_{ij})}dy_{ij} > 2\int_{\eta^{(0)}}^{\eta^{(t)}} d\eta.$$

Since the right part is going to infinity, the left part will also go to the positive infinity, and so will $y_{ij}$. Here we end the proof of Property 1 by Eq. 8.

**Proof for Property 2.** From the updating rules in Eq. 7, we can obtain:

$$\mathbf{x}_i^{(t+1)} - \mathbf{h}_j^{(t+1)} = \big(1 - \eta\sigma(-\mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)})\big)(\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)}),$$

so $\|\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)}\|$ is a monotonically decreasing sequence. Since $\|\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)}\| > 0$, the limit of $\|\mathbf{x}_i - \mathbf{h}_j\|$ must exist.

So, after enough steps, the following inequality will hold:

$$\mathbf{h}_j^\top\mathbf{h}_j + \mathbf{x}_i^\top\mathbf{x}_i = \|\mathbf{x}_i - \mathbf{h}_j\|^2 + 2\mathbf{x}_i^\top\mathbf{h}_j < 3\mathbf{x}_i^\top\mathbf{h}_j. \tag{14}$$

Then by Eq. 13, we have:

$$2d\eta \leq \frac{1}{y_{ij}\sigma(-y_{ij})}dy_{ij} < 3d\eta. \tag{15}$$

After integration and omitting the constant term, we will reach Property 2 by Eq. 9.

**Proof for Property 3.** We have proved that the limit of $\|\mathbf{x}_i - \mathbf{h}_j\|^2$ exists in the proof above, then we prove that this limit is zero. Based on Eq. 13, we have:

$$\begin{aligned}d(\|\mathbf{x}_i - \mathbf{h}_j\|^2) &= 2\mathbf{x}_i^\top d\mathbf{x}_i - 2d(\mathbf{x}_i^\top\mathbf{h}_j) + 2\mathbf{h}_j^\top d\mathbf{h}_j\\ &= -2\sigma(-\mathbf{x}_i^\top\mathbf{h}_j)\|\mathbf{x}_i - \mathbf{h}_j\|^2 d\eta\\ &= -2\frac{\|\mathbf{x}_i - \mathbf{h}_j\|^2}{\mathbf{x}_i^\top\mathbf{x}_i + \mathbf{h}_j^\top\mathbf{h}_j}d(\mathbf{x}_i^\top\mathbf{h}_j).\end{aligned} \tag{16}$$

Then combining Eq. 14, after enough training steps, we will have:

$$-\frac{1}{y_{ij}}dy_{ij} \leq \frac{1}{\|\mathbf{x}_i - \mathbf{h}_j\|^2}d\left(\|\mathbf{x}_i - \mathbf{h}_j\|^2\right) < -\frac{2}{3y_{ij}}dy_{ij}.$$

After integration, we have:

$$\ln\frac{y_{ij}^{(0)}}{y_{ij}} \leq \ln\frac{\|\mathbf{x}_i - \mathbf{h}_j\|^2}{\|\mathbf{x}_i^{(0)} - \mathbf{h}_j^{(0)}\|^2} < \frac{2}{3}\ln\frac{y_{ij}^{(0)}}{y_{ij}}. \tag{17}$$

Since we have proved that $\lim_{t\to\infty} y_{ij}^{(t)} = +\infty$ in the proof for Property 1, based on Eq. 17 and Squeeze Theorem, we can prove Property 3 by Eq. 10.

**Proof for Property 4.** Do integration over the second line of Eq. 16, we can reach the conclusion, Eq. 11.

**Proof for Property 5.** From Eq. 7, we have:

$$\mathbf{x}_i^{(t+1)} + \mathbf{h}_j^{(t+1)} = \big(1 + \eta\sigma(-\mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)})\big)(\mathbf{x}_i^{(t)} + \mathbf{h}_j^{(t)}).$$

Based on Property 3, Eq. 10, we have the limit of the normalized embedding for $\mathbf{x}_i^{(t)}$:

$$\begin{aligned}\lim_{t\to\infty}\frac{\mathbf{x}_i^{(t)}}{\|\mathbf{x}_i^{(t)}\|} &= \lim_{t\to\infty}\frac{(\mathbf{x}_i^{(t)} + \mathbf{h}_j^{(t)}) + (\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)})}{\|(\mathbf{x}_i^{(t)} + \mathbf{h}_j^{(t)}) + (\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)})\|}\\ &= \lim_{t\to\infty}\frac{\mathbf{x}_i^{(t)} + \mathbf{h}_j^{(t)}}{\|\mathbf{x}_i^{(t)} + \mathbf{h}_j^{(t)}\|} = \frac{\mathbf{x}_i^{(0)} + \mathbf{h}_j^{(0)}}{\|\mathbf{x}_i^{(0)} + \mathbf{h}_j^{(0)}\|}.\end{aligned}$$

Also, $\lim_{t\to\infty}\mathbf{h}_j^{(t)}/\|\mathbf{h}_j^{(t)}\|$ can be proved in the same way. So far we have finished all the proof. □

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3148284, IEEE Transactions on Knowledge and Data Engineering



**Comments on Theorem 1.** Theorem 1 shows the properties of the convergence of the NCE-based embedding models, and indicates that the minimization of $\|\mathbf{x}_i - \mathbf{h}_j\|$ can be achieved by maximizing $\sigma(\mathbf{x}_i^\top \mathbf{h}_j)$. However, it only holds in ideal situations, i.e. i) the learning rate $\eta$ is tiny enough, ii) the number of iterations is infinite, iii) the embedding $\mathbf{x}_i$ and $\mathbf{h}_j$ will not be affected by other embeddings, and iv) node $j$ will not be sampled as a negative node for $i$. In real-world applications, the case would usually be $\mathbf{x}_i^\top \mathbf{h}_j \ll \infty$ and $\|\mathbf{x}_i - \mathbf{h}_j\| \gg 0$, both far from their convergence points. By vanilla NCE-based embedding, proximity between nodes can hardly be preserved as shown by detailed analysis in the following Sec. 3.2.2 and Sec. 3.2.3.

### 3.2.2 Defects in the First-order Proximity Preservation

First-order proximity describes connection strengths between nodes. $w_{ij} > 0$ indicates the first-order proximity between nodes $i$ and $j$. To preserve the first-order proximity, nodes whose connection strength $w_{ij} > 0$ are expected to have similar $\mathbf{x}_i$ and $\mathbf{h}_j$ ($\mathbf{h}_j$ for NCE-h, $\mathbf{x}_j$ for NCE-x). However, we find that NCE may have difficulties to preserve the first-order proximity.

The most influential factor bringing the difficulties is the convergence speed. As proved by Theorem 1, the convergence speed of $\|\mathbf{x}_i - \mathbf{h}_j\|$ is very slow. Actually, by Eq. 6, the gradient of $\mathbf{x}_i$ is $\partial O^h_{nce}(i,j)/\partial \mathbf{x}_i = \sigma(-\mathbf{x}_i^\top \mathbf{h}_j)\mathbf{h}_j$. When $\mathbf{x}_i^\top \mathbf{h}_j$ is quite large, $\sigma(-\mathbf{x}_i^\top \mathbf{h}_j)$ becomes very small, leading to gradient vanishing before convergence. Hence, though the optimal solutions exist, they can hardly be reached within limited number of training iterations. One can see Fig. 6 which visualizes the updating trace of node embeddings of a positive pair and experimentally shows the difficulties to preserve the first-order proximity.

### 3.2.3 Defects in the Second-order Proximity Preservation

Second-order proximity indicates the similarity of nodes' neighborhoods. The more common neighbors between two nodes, the more similar they are by second-order proximity. Both NCE-h and NCE-x are able to yield similar node embedding for nodes sharing second-order proximity, however, only with enough neighbors.

To start with, we explore the optimal solution of the NCE objective Eq. 5 in Theorem 2. Similar proof can also be found in previous works [24], [25], [27].

**Theorem 2.** *Define $y_{ij} := \mathbf{x}_i^\top \mathbf{h}_j$ for NCE-h and $y_{ij} := \mathbf{x}_i^\top \mathbf{x}_j$ for NCE-x. For the positive node pairs $(i,j)$, the optimal solution $y^*_{ij}$ is:*

*For NCE-h, it is:*

$$y^*_{ij} = \log \frac{w_{ij}}{K\mathbb{P}_n(j)d_i^{out}}, \tag{18}$$

*For NCE-x, it is:*

$$y^*_{ij} = \log \frac{w_{ij} + w_{ji}}{K\left(\mathbb{P}_n(j)d_i^{out} + \mathbb{P}_n(i)d_j^{out}\right)}. \tag{19}$$

*where $d_i^{out} = \sum_{j'=1}^{N} w_{ij'}$ is the out-degree of node $i$.*

*Proof.* To start with, we calculate the partial derivative of the loss $L^{h/x}_{nce}$ with respect to $y_{ij}$:

$$\frac{\partial L^h}{\partial y_{ij}} = -w_{ij}\sigma(-y_{ij}) + K\mathbb{P}_n(j)d_i^{out}\sigma(y_{ij}), \tag{20}$$

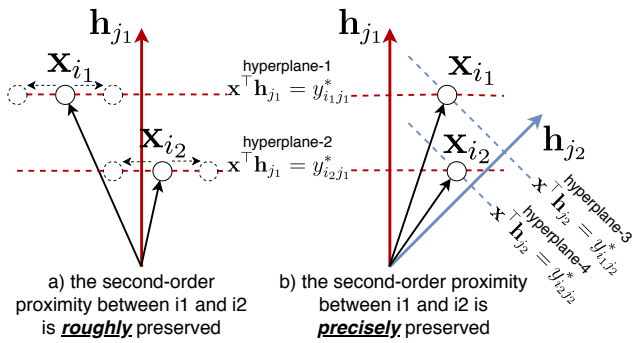



Fig. 3: A 2-D illustration of second-order proximity preservation by vanilla NCE-based embedding model. The dashed lines are the hyperplanes. **a)** nodes $i_1$ and $i_2$ have one common neighbor $j_1$. We can see that the distance $\|\mathbf{x}_{i_1} - \mathbf{x}_{i_2}\|$ is not fixed since $\mathbf{x}_{i_1}$ can be located at any position on hyperplane-1 and so does $\mathbf{x}_{i_2}$ on hyperplane-2. Since $\|\mathbf{x}_{i_1} - \mathbf{x}_{i_2}\|$ is not guaranteed while the distance of the two hyperplanes are determined, we can say that in this case the second-order proximity is 'roughly' preserved. **b)** nodes $i_1$ and $i_2$ have two common neighbors $j_1$ and $j_2$. We can see that $\mathbf{x}_{i_1}$ is located at the intersection of the two hyperplanes and so does $\mathbf{x}_{i_2}$. In this way, the distance $\|\mathbf{x}_{i_1} - \mathbf{x}_{i_2}\|$ is determined. And we say that the second-order proximity is hence 'precisely' preserved. By the two examples, we can claim that for a $d$-dimensional embedding problem, only with no less than $d$ common neighbors can the second-order proximity between two nodes be 'precisely' preserved.

and

$$\begin{aligned}\frac{\partial L^x}{\partial y_{ij}} =& -(w_{ij} + w_{ij})\sigma(-y_{ij}) \\ &+ \left(\mathbb{P}_n(j)d_i^{out} + \mathbb{P}_n(i)d_j^{out}\right)K\sigma(y_{ij}).\end{aligned} \tag{21}$$

By setting $\partial L^x/\partial y_{ij}$ and $\partial L^h/\partial y_{ij}$ to zero, we will reach the optimal solution (denoted with $^*$). □

Then, we take NCE-h as the example to explain how vanilla NCE objectives may fail in preserving nodes' second-order proximity in networks, and the analysis for NCE-x is also similar. According to Theorem 2, we can find that each hidden embedding $\mathbf{h}_j$ defines a class of hyperplanes sharing a common normal vector $\mathbf{h}_j$. If node $i$ connects with node $j$, the embedding $\mathbf{x}_i$ will be located on the hyperplane $\mathbf{x}^\top \mathbf{h}^*_j = y^*_{ij}$. When the two nodes $i_1$ and $i_2$ have a common neighbor $j$, their embedding $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$ are very likely to fall in the same hyperspace $\mathbf{x}^\top \mathbf{h}^*_j > 0$ since the value of $\frac{w_{ij}}{K\mathbb{P}_n(j)d_i^{out}}$ is mostly larger than 1 (for a brief explanation here: since $1/\mathbb{P}_n(j) \sim |\mathcal{V}|$, the number of nodes in the whole network, and $d_i^{out}/w_{ij} \sim |\mathcal{N}^{out}(i)|$, the number of neighbors of node $i$, and $|\mathcal{N}^{out}(i)| \ll |\mathcal{V}|$ in most cases in a large network, we can intuitively assume that the term $\frac{w_{ij}}{K\mathbb{P}_n(j)d_i^{out}}$ is mostly larger than 1 and the logarithm is larger than 0). In this way, the second-order proximity can be roughly preserved, however, still far from precisely. Specifically, we can see that only with more than $d$ neighbors ($d$ is the embedding dimension), the embedding of a node can be determined. Otherwise, the embedding

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3148284, IEEE Transactions on Knowledge and Data Engineering



can be randomly located anywhere on the intersection of the hyperplanes defined by its neighbors. It also means that the distance $\|\mathbf{x}_{i_1} - \mathbf{x}_{i_2}\|$ between $i_1$ and $i_2$ can only be guaranteed when both nodes $i_1$ and $i_2$ have at least common $d$ neighbors. To see this, we give an toy example for $d = 2$ in Fig. 3. The analysis suggests that the capability of vanilla NCE-based embedding to preserve second-order proximity is somehow limited.

### 3.3 The Limitation of High Linearity of NCE-h

In this part, we will show that the vanilla NCE-h based embedding tends to yield node embeddings in a linear distribution, mainly by analysis over negative samples (background noise in NCE). That causes the whole embedding layout to become more like lines in the embedding space, filtering the non-linear information of networks. However, the underlying structure of real-world networks is often highly nonlinear and hence cannot be accurately approximated by embeddings distributed linearly [22].

To prove that, we first introduce the following lemma, which somehow indicates the linearity of node embeddings.

**Lemma 1.** ***(For both NCE-h/x)*** *Given a network $G = (\mathcal{V}, \mathcal{E}, w)$, $\mathcal{V} = \{1, 2, \cdots N\}$, $\forall j \in \mathcal{V}$, the following formulas hold for the optimal solutions denoted with superscript $^*$. Here we use $\mathbb{1}_{condition}$ to denote an indicator function which takes value 1 when the condition holds and 0 otherwise. Then:*

*For NCE-h, we have:*

$$\sum_{i=1}^{N} \mathbb{1}_{w_{ij}=0} d_i^{out} \sigma(y_{ij}^*) \mathbf{x}_i^* = \mathbf{0}. \tag{22}$$

*For NCE-x, we have:*

$$\sum_{i=1}^{N} \mathbb{1}_{w_{ij}w_{ji}=0} \big(\mathbb{P}_n(j) d_i^{out} + \mathbb{P}_n(i) d_j^{out}\big) \sigma(y_{ij}) \mathbf{x}_i^* = \mathbf{0} \tag{23}$$

*Proof.* For NCE-h, by setting $\partial L^h / \partial \mathbf{h}_j = \mathbf{0}$ where $L^h$ is defined in Eq. 4, we have:

$$\sum_{i=1}^{N} \Big( w_{ij} \sigma(-y_{ij}^*) - K \mathbb{P}_n(j) d_i^{out} \sigma(y_{ij}^*) \Big) \mathbf{x}_i^* = \mathbf{0}.$$

Then we put the terms of $\mathbf{x}_i^*$ where $w_{ij} > 0$ on the left side and the terms where $w_{ij} = 0$ on the right side:

$$\begin{aligned} &\sum_{i=1}^{N} \mathbb{1}_{w_{ij}>0} \left( w_{ij} - \big(w_{ij} + K \mathbb{P}_n(j) d_i^{out}\big) \sigma(y_{ij}^*) \right) \mathbf{x}_i^* \\ =& K \mathbb{P}_n(j) \sum_{i=1}^{N} \mathbb{1}_{w_{ij}=0} d_i^{out} \sigma(y_{ij}^*) \mathbf{x}_i^*. \end{aligned} \tag{24}$$

Given the optimal solution for $y_{ij}$ of positive node pairs (Eq. 18), we will find the left-hand side of Eq. 24 be zero and then reach Eq. 22.

For NCE-x, by setting $\partial L^x / \partial \mathbf{x}_j = \mathbf{0}$ ($L^x$ is defined in Eq. 4), we obtain:

$$\begin{aligned} \sum_{i=1}^{N} \Big[ &(w_{ij} + w_{ji}) \sigma(-y_{ij}) \\ &- K \Big( \mathbb{P}_n(j) d_i^{out} + \mathbb{P}_n(i) d_j^{out} \Big) \sigma(y_{ij}) \Big] \mathbf{x}_i^* = \mathbf{0}, \end{aligned}$$

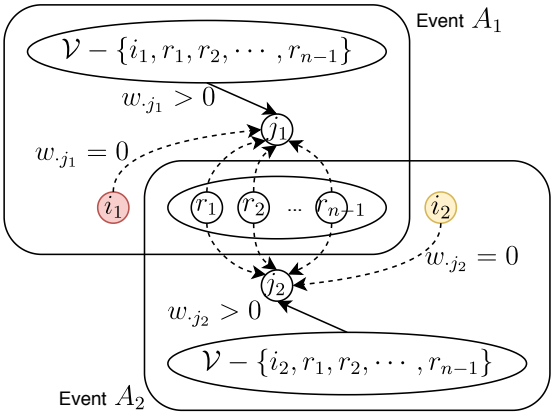



Fig. 4: Illustration of event $A_1$ and $A_2$ in a random graph. By the two events, we can probably find a node $i_2$, whose embedding $\mathbf{x}_{i_2}$ is approximately parallel to $i_1$'s embedding $\mathbf{x}_{i_1}$, even when $i_2$ and $i_1$ do not share any obvious proximity.

By combining the optimal solution of $y_{ij}$ given in Eq. 19, we will reach Eq. 23. □

Then we will prove the following Theorem 3, which is based on analysis over random graphs [28]. It further proves the linear distribution of embedding by vanilla NCE-h.

**Theorem 3.** ***(For NCE-h)*** *Assume $G = (\mathcal{V}, \mathcal{E}, w)$ is a directed random graph [28] of $N$ nodes where edges are created by a given probability $p$. In the optimal solution of Eq. 5, we suppose that $\forall$ node pair $(i, j)$ s.t. $w_{ij} = 0$, it has $\sigma(y_{ij}^*) \in (u - \epsilon, u + \epsilon)$ where $\epsilon \ll u$, then $\forall$ node $i_1 \in \mathcal{V}$, it is possible to find such a node $i_2$, whose embedding $\mathbf{x}_{i_2}$ is almost parallel to $i_1$'s embedding $\mathbf{x}_{i_1}$. Here 'almost parallel' means there exists $\xi$ s.t. $\|\mathbf{x}_{i_1} - \xi \mathbf{x}_{i_2}\|$ is a small value compared with $\|\mathbf{x}_{i_1}\|$ and $\|\mathbf{x}_{i_2}\|$. We denote the expectation of the number of such nodes $i_2$ as $\mathbb{E}\#i_2$, which satisfies the following inequality:*

$$\mathbb{E}\#i_2 \geq \sum_{n=2}^{N-1} \binom{n-1}{N-1} N(N-1)(N-n) p^{2N-2n} (1-p)^{2n}, \tag{25}$$

*where $\binom{\cdot}{\cdot}$ denotes combination number.*

*Proof.* Given $n-1$ random nodes $\{r_1, r_2, \cdots, r_{n-1}\} \subset \mathcal{V}$, we denote the event as $A_1$ that nodes $\mathcal{V} - \{i_1, r_1, r_2, \cdots, r_{n-1}\}$ connect to the same random node $j_1 \in \mathcal{V}$ while nodes $\{i_1, r_1, r_2, \cdots, r_{n-1}\}$ do not. We plot an illustration of the event $A_1$ in Fig. 4, where the weights $w_{\cdot j_1}$ between nodes $\{i_1, r_1, r_2, \cdots, r_{n-1}\}$ and $j_1$ have $w_{\cdot j_1} = 0$, while $w_{\cdot j_1}$ between nodes $\mathcal{V} - \{i_1, r_1, r_2, \cdots, r_{n-1}\}$ and $j_1$ have $w_{\cdot j_1} > 0$. Then we denote an event as $A_2$ that a random node $i_2$ ($i_2 \neq i_1$) and the nodes $\{r_1, r_2, \cdots, r_{n-1}\}$ connect to another random node $j_2$ ($j_2 \neq j_1$). We also plot the illustration of $A_2$ in Fig. 4.

According to Lemma 1, event $A_1$ indicates that:

$$d_{i_1}^{out} \sigma(y_{i_1 j_1}^*) \mathbf{x}_{i_1}^* + \sum_{k=1}^{n-1} d_{r_k}^{out} \sigma(y_{r_k j_1}^*) \mathbf{x}_{r_k}^* = \mathbf{0}. \tag{26}$$

Similarly, event $A_2$ indicates that:

$$d_{i_2}^{out} \sigma(y_{i_2 j_2}^*) \mathbf{x}_{i_2}^* + \sum_{k=1}^{n-1} d_{r_k}^{out} \sigma(y_{r_k j_2}^*) \mathbf{x}_{r_k}^* = \mathbf{0}. \tag{27}$$

By Eq. 23 and Eq. 27, events $A_1$ and $A_2$ jointly indicate:

$$\begin{aligned} & d_{i_1}^{out}\sigma(y_{i_1j_1}^*)\mathbf{x}_{i_1}^* - d_{i_2}^{out}\sigma(y_{i_2j_2}^*)\mathbf{x}_{i_2}^* \\ =& \sum_{k=1}^{n-1} d_{r_k}^{out}(\sigma(y_{r_kj_2}^*) - \sigma(y_{r_kj_1}^*))\mathbf{x}_{r_k}^*. \end{aligned}$$

By triangle inequality, we have:

$$\begin{aligned} & \left\| d_{i_1}^{out}\sigma(y_{i_1j_1}^*)\mathbf{x}_{i_1}^* - d_{i_2}^{out}\sigma(y_{i_2j_2}^*)\mathbf{x}_{i_2}^* \right\| \\ \leq& \sum_{k=1}^{n-1} d_{r_k}^{out}\left(\sigma(y_{r_kj_2}^*) - \sigma(y_{r_kj_1}^*)\right)\left\|\mathbf{x}_{r_k}^*\right\| \\ <& 2\epsilon \sum_{k=1}^{n-1} d_{r_k}^{out}\left\|\mathbf{x}_{r_k}^*\right\|. \end{aligned}$$

In a further step, we have:

$$\left\| \mathbf{x}_{i_2}^* - \frac{d_{i_1}^{out}\sigma(y_{i_1j_1}^*)}{d_{i_2}^{out}\sigma(y_{i_2j_2}^*)}\mathbf{x}_{i_1}^* \right\| < \frac{2\epsilon}{\sigma(y_{i_2j_2}^*)} \sum_{k=1}^{n-1} \frac{d_{r_k}^{out}}{d_{i_2}^{out}} \left\|\mathbf{x}_{r_k}^*\right\|.$$

When $\epsilon \ll u$, the right part of the inequality above can be very small. Hence the embedding vector $\mathbf{x}_{i_1}^*$ and $\mathbf{x}_{i_2}^*$ can be almost parallel. That means, by event $A_1$ and event $A_2$, we can actually find a random node $i_2$ whose embedding $\mathbf{x}_{i_2}$ is highly linear to $\mathbf{x}_{i_1}$.

Then we try to figure out the probability of event $A_1$:

$$P(A_1) = \binom{1}{N}\binom{n-1}{N-1}p^{N-n}(1-p)^n.$$

For event $A_2$, We have the conditional probability:

$$P(A_2|A_1) = \binom{1}{N-n}\binom{1}{N-1}p^{N-n}(1-p)^n.$$

Then we explore the joint probability that the event $A_1A_2$ happens as following:

$$\begin{aligned} P(A_1A_2) &= P(A_1)P(A_2|A_1) \\ &= \binom{n-1}{N-1}N(N-1)(N-n)p^{2N-2n}(1-p)^{2n}. \end{aligned}$$

As all the $n$ satisfying $2 \leq n \leq N-1$ should be considered, the expectation of the number of node $i_2$ holds:

$$\mathbb{E}\#i_2 \geq \sum_{n=2}^{N-1} P(A_1A_2),$$

which is the same as Eq. 25. Here the proof ends. □

Theorem 3 proves that NCE-h is likely to embed nodes into a linear distribution. Also, it indicates that to reduce $\mathbb{E}\#i_2$, one feasible way is to increase $p$, namely making networks denser. For example, among existing NCE-h based embedding methods, DeepWalk [10] uses random walk to connect nodes within a given number of hops.

### 3.4 Towards Regularized NCE-based Embedding

In Sec. 3.1, we give the preliminaries for NCE-based embedding. Then, for positive samples, we theoretically show the convergence speed of NCE and analyze the capability of NCE-based embedding methods to preserve first-/second-order proximity between nodes in Sec. 3.2. In a further step, by analyzing the embeddings of negative samples, we prove that NCE-h may yield node embedding of high linearity, thus possibly limiting the expressiveness of non-linearity of complex real-world networks. In the next section, we propose to add distance functions to NCE-based methods. Specifically, in Sec. 4.2, we elaborate on how distance functions improve the properties of vanilla NCE-based embedding from several aspects, especially overcoming the mentioned defects of NCE-based embedding.

## 4 Proposed Regularized NCE Embedding with Distance Functions

Several drawbacks of vanilla NCE-based embedding models are mentioned in the last section. One of the core problems is that the distances between connected nodes are not guaranteed. To improve the NCE-based embedding objective, we propose to add distance functions in the loss $L^z$. In the rest of this section, we describe the derived models by adding a distance function $O_D^h$ to Eq. 2, and leave those for Eq. 3 in appendix. In fact, the distance forms and the resulting models vary slightly.

**Objective.** We first define the distribution of nodes. The distribution of nodes being sampled for node embedding is denoted as $\mathbb{P}^x$, and nodes being sampled for hidden embedding as $\mathbb{P}^h$. The joint distribution is defined as $\mathbb{P}^{(x,h)}$. We use $P^x, P^h : \mathcal{V} \to \mathbb{R}$ to denote the probability that nodes are sampled for node/hidden embedding. $P^{(x,h)} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ denotes the joint probability that two nodes are sampled as a positive node pair. Empirically, we have $P^x(i) = d_i^{out}/\sum_{j=1}^N d_j^{out}$, $P^h(i) = d_i^{in}/\sum_{j=1}^N d_j^{in}$, and $P^{(x,h)}(i,j) = w_{ij}/\sum_{k=1}^N d_k^{out}$. The introduced distance function for NCE-$z$ based embedding is denoted as $O_{dis}^z : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$.

Then we modify the optimization problem of vanilla NCE-based embedding methods (recall the loss $L^z$ in Eq. 4, Sec. 3.1) to a constrained optimization problem as follows:

$$\min_{\{\mathbf{x}_i, \mathbf{h}_i | i \in \mathcal{V}\}} L^z \quad \text{s.t.} \quad \mathbb{E}_{(i,j)\sim\mathbb{P}^{(x,h)}} O_{dis}^z(i,j) < C_{dis}, \tag{28}$$

by which we hope that the expectation of the distance between connected nodes is a constant $C_{dis}$. With a Lagrange multiplier $\beta$, the final objective is defined as:

$$\min_{\{\mathbf{x}_i, \mathbf{h}_i | i \in \mathcal{V}\}} L^z + \beta \sum_{i,j} P^{(x,h)}(i,j)\left(O_{dis}^z(i,j) - C_{dis}\right),$$

which is equivalent to:

$$\min_{\{\mathbf{x}_i, \mathbf{h}_i | i \in \mathcal{V}\}} L^z + \beta \sum_{i,j} P^{(x,h)}(i,j) O_{dis}^z(i,j). \tag{29}$$

The Lagrange multiplier $\beta$ controls how heavily the NCE-based embedding depends on the distance $O_{dis}^z(i,j)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3148284, IEEE Transactions on Knowledge and Data Engineering



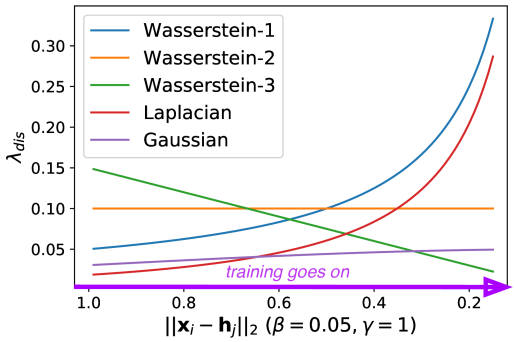



Fig. 5: Comparison of different distance functions used in the NCE-h methods. Note the curve trends are similar in NCE-x cases which are omitted here. A larger value of $x$-axis means that $\mathbf{x}_i$ and $\mathbf{h}_j$ are far away in embedding space. It usually happens in the early training stage. A larger value of $\lambda_{dis}$ means a larger weight on term $(\mathbf{x}_i - \mathbf{h}_j)$ in gradients.

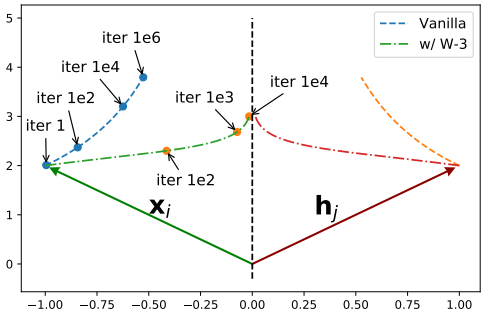



Fig. 6: The comparison between the updating trace of the vector $\mathbf{x}_i$ and $\mathbf{h}_j$ by vanilla NCE and NCE regularized with Wasserstein-3 distance. The initial embeddings are $\mathbf{x}_i^{(0)} = (-1, 2)^\top$ and $\mathbf{h}_j^{(0)} = (1, 2)^\top$. The learning rate of SGD is 0.01. We can see that after 1e4 iterations, regularized NCE almost converges at $\|\mathbf{x}_i - \mathbf{h}_j\| = 0$, while by vanilla NCE, $\mathbf{x}_i$ and $\mathbf{h}_j$ are still somehow far away even after 1e6 iterations. This figure further strengthens the proposed Theorem 1 and Theorem 4, which theoretically give the convergence speed of NCE and regularized NCE respectively.

### 4.1 Distance Functions

We discuss several interesting and popular distance functions, which will be evaluated in experiments.

**Elastic Potential Energy (Wasserstein-2).** Given a stretched spring with stiffness coefficient $k_{stf}$ between node $i$ at point $\mathbf{x}_i$ and $j$ at $\mathbf{h}_j$, to minimize the elastic potential energy of the physical system, the objective for each positive sample $(i, j)$ is given by:

$$O^h_{dis}(i, j) = \frac{1}{2} k_{stf} \|\mathbf{x}_i - \mathbf{h}_j\|^2.$$

**Wasserstein-$k$ Distance.** With determined joint distribution $\mathbb{P}^{(x,h)}$, Wasserstein-$k$ distance [29] is defined as $\left(\mathbb{E}_{(i,j)\sim\mathbb{P}^{(x,h)}} \|\mathbf{x}_i - \mathbf{h}_j\|^k\right)^{1/k}$. Since it can be very hard to calculate the Wasserstein-$k$ distance for a large network, we use the $k$-th power of Wasserstein distance $\sum_{i,j} P^{(x,h)}(i, j)\|\mathbf{x}_i - \mathbf{h}_j\|^k$ which can be more easily dealt with by sampling. Then the distance function is defined as:

$$O^h_{dis}(i, j) = \|\mathbf{x}_i - \mathbf{h}_j\|^k.$$

When $k = 2$, the objective becomes elastic potential with $k_{stf} = 2$. In the following of the paper, we use Wasserstein-2 to refer to the elastic potential energy and discard the parameter $k_{stf}$ as it can be expressed by the parameter $\beta$.

**Gaussian Kernel Function.** To make the function yield a larger value for larger $\|\mathbf{x}_i - \mathbf{h}_j\|$, we define the distance as:

$$O^h_{dis}(i, j) = -\exp(-\gamma\|\mathbf{x}_i - \mathbf{h}_j\|^2)$$

**Laplacian Kernel Function.** Similar with the Gaussian kernel, the distance for Laplacian kernel is defined as:

$$O^h_{dis}(i, j) = -\exp(-\gamma\|\mathbf{x}_i - \mathbf{h}_j\|)$$

**The gradients of distance functions.** One of the common points of the listed distance functions is that the gradients follow a common form:

$$\frac{\partial O^h_{dis}(i, j)}{\partial \mathbf{h}_j} = \lambda_{dis}(\mathbf{x}_i - \mathbf{h}_j), \tag{30}$$

where $\lambda_{dis}$ is a distance-function-specific scalar variable. The main difference of the distance functions lies on the weight, $\lambda_{dis}$, they give to $(\mathbf{x}_i - \mathbf{h}_j)$ in gradients. In earlier learning stage, embedding is desired to be updated with bigger steps. In later stage, besides the decaying effect on learning rate, it is worth to encourage the gradient itself also to decay for convergence. Note for the example in Fig. 5, Wasserstein-3 behaves in this way while Laplacian/Gaussian kernel functions and Wasserstein-1/2 do not.

### 4.2 Properties of Regularized NCE

In this part, we discuss the properties of NCE-based embedding with distance functions. Compared with the properties of the vanilla NCE-based embedding models as discussed in Sec. 3.2 and Sec. 3.3, what we proposed improves the vanilla NCE-based embedding significantly.

#### 4.2.1 Faster Convergence

We theoretically analyze the convergence speed of regularized NCE in Theorem 4. Compared with the convergence speed of vanilla NCE in Eq. 11, Theorem 1, we show that regularized NCE is ideally expected to have faster convergence. Specifically, we use Theorem 4 to theoretically give the convergence speed of regularized NCE, and Remark 1 to claim that distance functions won't influence the final solutions given by vanilla NCE with SGD, and Theorem 5 to prove that under certain constraints, by SGD iterations regularized NCE will yield faster convergence. Moreover, in Fig. 6, we give a toy example for the updating trace of two embedding $\mathbf{x}_i$ and $\mathbf{h}_j$ w/o a distance function to regularize, which further illustrates the following theorems and remarks.

**Theorem 4.** *The convergence speed of* $\|\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)}\|^2$ *follows:*

$$\int_{\|\mathbf{x}_i^{(0)} - \mathbf{h}_j^{(0)}\|^2}^{\|\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)}\|^2} \frac{d\left(\|\mathbf{x}_i - \mathbf{h}_j\|^2\right)}{\|\mathbf{x}_i - \mathbf{h}_j\|^2 \left[\sigma(-\mathbf{x}_i^\top \mathbf{h}_j) + 2\beta\lambda_{dis}\right]} \sim -\int_{\eta^{(0)}}^{\eta^{(t)}} d\eta. \tag{31}$$

*Proof.* **A single updating step.** First, we give the updating step given by regularized NCE-based embedding (recall the updating step given by vanilla NCE as Eq. 7). Combining the gradients of vanilla NCE (Eq. 6) and the gradients of the distance functions (Eq. 30), we can derive the embedding $\mathbf{x}_i^{(t+1)}$ and $\mathbf{h}_j^{(t+1)}$ at time $t+1$ as following (we use $\cdot$ to denote embeddings updated by regularized NCE):

$$\begin{aligned}\mathbf{x}_i^{(t+1)} &\leftarrow \mathbf{x}_i^{(t+1)} + \eta\beta\lambda_{dis}(\mathbf{h}_j^{(t)} - \mathbf{x}_i^{(t)}) \\ &= \mathbf{x}_i^{(t)} + \eta\sigma\Big(-\mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)}\Big)\mathbf{h}_j^{(t)} + \eta\beta\lambda_{dis}(\mathbf{h}_j^{(t)} - \mathbf{x}_i^{(t)}), \\ \mathbf{h}_j^{(t+1)} &\leftarrow \mathbf{h}_j^{(t+1)} + \eta\beta\lambda_{dis}(\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)}) \\ &= \mathbf{h}_j^{(t)} + \eta\sigma\Big(-\mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)}\Big)\mathbf{x}_i^{(t)} + \eta\beta\lambda_{dis}(\mathbf{x}_i^{(t)} - \mathbf{h}_j^{(t)}).\end{aligned} \tag{32}$$

Writing Eq. 32 in a differential form, we can derive:

$$\begin{aligned}d\mathbf{x}_i &= \Big[\sigma(-\mathbf{x}_i^\top\mathbf{h}_j)\mathbf{h}_j + \beta\lambda_{dis}(\mathbf{h}_j - \mathbf{x}_i)\Big]d\eta, \\ d\mathbf{h}_j &= \Big[\sigma(-\mathbf{x}_i^\top\mathbf{h}_j)\mathbf{x}_i + \beta\lambda_{dis}(\mathbf{x}_i - \mathbf{h}_j)\Big]d\eta,\end{aligned}$$

based on which we can derive:

$$\begin{aligned}d(\|\mathbf{x}_i - \mathbf{h}_j\|^2) &= 2(\mathbf{x}_i^\top - \mathbf{h}_j^\top)(d\mathbf{x}_i - d\mathbf{h}_j) \\ &= -2\|\mathbf{x}_i - \mathbf{h}_j\|^2\Big[\sigma(-\mathbf{x}_i^\top\mathbf{h}_j) + 2\beta\lambda_{dis}\Big]d\eta.\end{aligned}$$

One can obtain Eq. 31 by integration with the constant multiplier omitted. The proof ends. □

**Comments on Theorem 4 (why the convergence of regularized NCE is faster than vanilla NCE).** Since it is not very obvious that Eq. 31 shows a faster convergence speed than Eq. 11 (Property 4, Theorem 1) which indicates the convergence speed of $\|\mathbf{x}_i - \mathbf{h}_j\|^2$ by vanilla NCE, here we give some detailed explanations. Comparing Eq. 31 with Eq. 11, we can find that the main difference lies on the denominators where the term $\sigma(-\mathbf{x}_i^\top\mathbf{h}_j)$ in Eq. 11 is substituted with $\sigma(-\mathbf{x}_i^\top\mathbf{h}_j) + 2\beta\lambda_{dis}$ in Eq. 31. By Property 1 in Theorem 1, we know that after enough updating steps $\mathbf{x}_i^\top\mathbf{h}_j$ will be quite large and $\sigma(-\mathbf{x}_i^\top\mathbf{h}_j)$ is close to zero. Approximately, we have $\sigma(-\mathbf{x}_i^\top\mathbf{h}_j) \approx \exp(-\mathbf{x}_i^\top\mathbf{h}_j)$, so we have $\frac{\sigma(-\mathbf{x}_i^\top\mathbf{h}_j)+2\beta\lambda_{dis}}{\sigma(-\mathbf{x}_i^\top\mathbf{h}_j)} \approx 1 + 2\exp(\mathbf{x}_i^\top\mathbf{h}_j)\beta\lambda_{dis}$. So, regularized NCE converges about $1+2\exp(\mathbf{x}_i^\top\mathbf{h}_j)\beta\lambda_{dis}$ times faster than vanilla NCE. Since $\beta$ is a constant and $\lambda_{dis}$ depends on the distance function and $\|\mathbf{x}_i - \mathbf{h}_j\|^2$, how much regularized NCE is faster is different for the distance functions. For Wasserstein-2 distance where $\lambda_{dis}$ is a constant (recall Fig. 5 here), it can be exponentially faster, and for Wasserstein-1 and Gaussian/Laplacian distance, it can be even faster. For Wasserstein-3 distance where $\lambda_{dis}$ decreases as the training procedure goes on, it is easy to prove that the convergence of regularized NCE is still much faster than vanilla NCE.

Beside Theorem 4 which theoretically shows the faster convergence speed of regularized NCE, we also derive Remark 1. It indicates that the regularized distance term won't affect the final solutions for positive samples. Because the proof of Remark 1 is very similar with the proof of properties in Theorem 1, we omit the proof here.

**Remark 1.** *For regularized NCE-based embedding, Property 1, 3, 5 in Theorem 1 still hold.*

In a further step, by the following Theorem 5, we give the constraints of $\beta$, under which the distance functions will give positive effects on the convergence of NCE-based models in each iteration with the same learning rate.

**Theorem 5.** ***(A necessary and sufficient condition for faster convergence for both NCE-h/x)*** *Suppose that at time $t$ a positive node pair $(i,j)$ is sampled, whose embeddings are $\mathbf{x}_i^{(t)}$ and $\mathbf{h}_i^{(t)}$. Given a distance function $O_{dis}^z(\cdot,\cdot)$, the corresponding $\lambda_{dis}$ is defined by Eq. 30. To distinguish the embeddings by regularized NCE from the embeddings by vanilla NCE, we denote the embeddings updated by regularized NCE as $\widetilde{\mathbf{x}}_i^{(t+1)}$ and $\widetilde{\mathbf{h}}_j^{(t+1)}$ at time $t+1$, and the embeddings updated by vanilla NCE as $\mathbf{x}_i^{(t+1)}$ and $\mathbf{h}_i^{(t+1)}$. Then with learning rate $\eta$, then:*

$$\begin{aligned}&\beta < \frac{1}{\eta\lambda_{dis}} + \frac{1}{\lambda_{dis}}\sigma\left(-\mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)}\right) \\ \Leftrightarrow &\widetilde{\mathbf{x}}_i^{(t+1)\top}\widetilde{\mathbf{h}}_j^{(t+1)} > \mathbf{x}_i^{(t+1)\top}\mathbf{h}_j^{(t+1)},\end{aligned} \tag{33}$$

*where the second line indicates a faster speed of convergence.*

*Proof.* By performing vector inner-product for embedding at time $t+1$ according to Eq.32, we have:

$$\begin{aligned}&\widetilde{\mathbf{x}}_i^{(t+1)\top}\widetilde{\mathbf{h}}_j^{(t+1)} = \mathbf{x}_i^{(t+1)\top}\mathbf{h}_j^{(t+1)} \\ &+ \eta\beta\lambda_{dis}\left(1 - \eta\beta\lambda_{dis} + \eta\sigma\Big(-\mathbf{x}_i^{(t)\top}\mathbf{h}_j^{(t)}\Big)\right)\|\mathbf{h}_j^{(t)} - \mathbf{x}_i^{(t)}\|^2.\end{aligned}$$

It is apparent that Eq. 33 holds. The proof ends. □

#### 4.2.2 Improved First-order Proximity Preservation

The improvement of the gradient for first-order preservation is in two folds. First, compared with the vanilla NCE which does not preserve first-order proximity directly, the term of distance functions used in regularized NCE explicitly preserve the proximity. Specifically, the objective in Eq. 28 ensures that the expectation of distance between two nodes in a positive node pair is limited. It is somehow apparent to find that once the expectation of defined distance $\mathbb{E}O_{dis}^z(i,j)$ is determined, the expectation of Euclidean distance $\mathbb{E}\|\mathbf{x}_i - \mathbf{h}_j\|$ is also decided. Hence the preservation of the first-order proximity is guaranteed. We can assume:

$$\mathbb{E}\|\mathbf{x}_i - \mathbf{h}_j\| = C_1, \tag{34}$$

where $C_1$ is a constant. Second, regularized NCE relieves the problem brought by gradient vanishing. According to the updating rules given by regularized NCE (Eq. 32), we can find that when the gradients from NCE loss are close to vanishing, i.e. $\sigma(\mathbf{x}_i^\top\mathbf{h}_j) \to 1$, the whole gradients will not be lost when $\lambda_{dis} > 0$. It indicates the embedding still goes more similar for connected nodes.

#### 4.2.3 Improved Second-order Proximity Preservation

As mentioned in Sec. 3.1, the embedding based on vanilla NCE needs no less than $d$ neighbors to guarantee the second-order proximity preservation. In comparison, an additional distance function guarantees a limited expectation of the Euclidean distance between the node embedding of two nodes who have only one common neighbor. Supposing that two nodes $i_1$ and $i_2$ have a common neighbor $j$, by triangle inequality we have:

$$\|\mathbf{x}_{i_1} - \mathbf{x}_{i_2}\| \le \|\mathbf{x}_{i_1} - \mathbf{h}_j\| + \|\mathbf{x}_{i_2} - \mathbf{h}_j\|.$$

Then according to Eq. 34, the second-order proximity preservation can be guaranteed by:

$$\mathbb{E}\|\mathbf{x}_{i_1} - \mathbf{x}_{i_2}\| \leq 2C_1. \tag{35}$$

#### 4.2.4 Nonlinearize the Embedding Distribution of NCE-h

By adding a distance function, the mentioned linear distribution of embedding by NCE-h in Sec. 3.3 can no longer be found in the embedding by regularized NCE-h. We clarify that by the following Remark 2. The property indicates that regularized NCE-h may be better at preserving the non-linearity of networks compared with vanilla NCE-h.

**Remark 2.** *With the same assumption of Theorem 3, given a node $i_1$, one cannot find such a node $i_2$ whose embedding $\mathbf{x}_{i_2}$ is almost parallel with $\mathbf{x}_{i_1}$ through event $A_1$ and $A_2$ in Theorem 3 by regularized NCE.*

*Proof.* For NCE-h, we set $\partial L^1/\partial \mathbf{h}_j = \mathbf{0}$, then we have:

$$\begin{aligned}&\sum_{i=1}^{N}\Big(w_{ij}\sigma(-y^*_{ij}) - K\mathbb{P}_n(j)d_i^{out}\sigma(y^*_{ij})\Big)\mathbf{x}^*_i\\ =&\beta\sum_{i=1}^{N} w_{ij}\lambda_{dis}(\mathbf{x}^*_i - \mathbf{h}^*_j).\end{aligned}$$

Then we follow the proof of Theorem 3, by event $A_1$ and event $A_2$, and we have:

$$\begin{aligned}&d_{i_1}^{out}\sigma(y^*_{i_1j_1})\mathbf{x}^*_{i_1} - d_{i_2}^{out}\sigma(y^*_{i_2j_2})\mathbf{x}^*_{i_2}\\ =&\sum_{k=1}^{n-1} d_{r_k}^{out}\left(\sigma(y^*_{r_kj_2}) - \sigma(y^*_{r_kj_1})\right)\mathbf{x}^*_{r_k}\\ &+\frac{\beta}{K\mathbb{P}_n(j_1)}\sum_{i=1}^{N} w_{ij_1}\lambda_{dis}(\mathbf{h}^*_{j_1} - \mathbf{x}^*_i)\\ &-\frac{\beta}{K\mathbb{P}_n(j_2)}\sum_{i=1}^{N} w_{ij_2}\lambda_{dis}(\mathbf{h}^*_{j_2} - \mathbf{x}^*_i).\end{aligned}$$

by which we can see that embedding $\mathbf{x}^*_{i_2}$ and $\mathbf{x}^*_{i_1}$ is not 'almost parallel' (defined in Theorem 3). □

#### 4.2.5 Improvement on Label Smoothing

As discussed in Sec. 4.2.2 and Sec. 4.2.3, the added distance term will result in the limited Euclidean distance between the node embedding and hidden embedding of connected nodes. Here we will show the necessity to have a limited Euclidean distance between the embedding of nodes that shares any-order proximity, especially for node classification. Suppose there is a ground-truth label mapping function $\mathcal{M}: \mathbb{R}^d \to \mathbb{R}^c$ which maps node embedding to a label vector which has $c$ classes. When $\mathcal{M}$ is differentiable and $L$-Lipschitz, for any two nodes $i_i$ and $i_2$ we have:

$$\|\mathcal{M}(\mathbf{x}_{i_1}) - \mathcal{M}(\mathbf{x}_{i_2})\| \leq L\|\mathbf{x}_{i_1} - \mathbf{x}_{i_2}\|.$$

That means, if the embedding distance of two nodes is small, their predicted labels by $\mathcal{M}$ will also be similar. Through the first-/second-order proximity preservation as guaranteed by Eq. 34 and Eq. 35, label smoothing between nodes sharing proximity is reached.

**Algorithm 1:** NCE based learning for network embedding via ASGD. As a general framework, it incorporates different forms of distance function and neighbor expansion (see Table 1).

**Input**: vanilla NCE mode $z = \{h, x\}$, embedding dimension $d$, number of negative samples (for each positive sample): $K$, weight $\beta$, network $G$, learning rate $\eta$;

**for all** positive sample $(i, j)$ **do**
  $\mathbf{x}_{err} \leftarrow -\eta\frac{\partial\left(O^z_{nce}(i,j)+\beta O^z_D(i,j)\right)}{\partial \mathbf{x}_i^{(t)}}$;
  $\mathbf{h}_j^{(t+1)} \leftarrow \mathbf{h}_j^{(t)} - \eta\frac{\partial\left(O^z_{nce}(i,j)+\beta O^z_D(i,j)\right)}{\partial \mathbf{h}_j^{(t)}}$; //$O^z_D$ is distance loss;
  **for** $n = 1, 2, \ldots, K$ **do**
    Sample a negative node $j'$;
    $\mathbf{x}_{err} \leftarrow \mathbf{x}_{err} - \eta\frac{\partial O^z_{nce}(i,j')}{\partial \mathbf{x}_i^{(t)}}$;
    $\mathbf{h}_{j'}^{(t+1)} \leftarrow \mathbf{h}_{j'}^{(t)} - \eta\frac{\partial O^z_{nce}(i,j')}{\partial \mathbf{h}_{j'}^{(t)}}$;
  **end for**
  $\mathbf{x}_i^{(t+1)} \leftarrow \mathbf{x}_i^{(t)} + \mathbf{x}_{err}$; //Update $\mathbf{x}_i$ as a batch;
**end for**
**return** node embedding set $\{\mathbf{x}_i\}$;

#### 4.2.6 Further Discussion on Distance Functions

We review and summarize the features of the regularized NCE embedding. We recommend readers to re-read Fig. 2 to see the improvement in different aspects of regularized NCE-based embedding compared with the vanilla one. In the sections before, we discussed about several properties of regularized NCE embedding, including faster convergence (Sec. 4.2.1), improved ability to preserve first-order proximity (Sec. 4.2.2) and second-order proximity (Sec. 4.2.3), better capability in expressing non-linear structures (Sec. 4.2.4), and improvement on label smoothing (Sec. 4.2.5).

The discussed properties are shared commonly by all listed distance functions in this paper but not limited to them. All the distance function whose gradient has the form of Eq. 30 will obtain the privileges. However, we strongly recommend users to use Wasserstein-3 distance in practice. The reasons are in two folds:

- **The Stability of Training.** From Fig. 5, we can see that Wasserstein-3 distance is the only function where the coefficient $\lambda_{dis}$ decreases as the training procedure goes on. A lot of mature optimization skills such as SGD will decay the learning rate for stable training. And the decay of $\lambda_{dis}$ also takes a similar effect during training. In comparison, the $\lambda_{dis}$ of Wasserstein-1 and Laplacian kernel will be very large when the distance $\|\mathbf{x}_i - \mathbf{h}_j\|_2$ close to zero (convergence), which might result in a huge updating step thus ruining the training procedure.
- **The Stability of Faster Convergence.** Note that we give a necessary and sufficient condition for faster convergence in Theorem 5, which indicates that only with $\lambda_{dis} < 1/\beta\eta + \sigma(-\mathbf{x}_i^{(t)}\mathbf{h}_j^{(t)})/\beta$ will the model converge faster. However, we can clearly see that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3148284, IEEE Transactions on Knowledge and Data Engineering



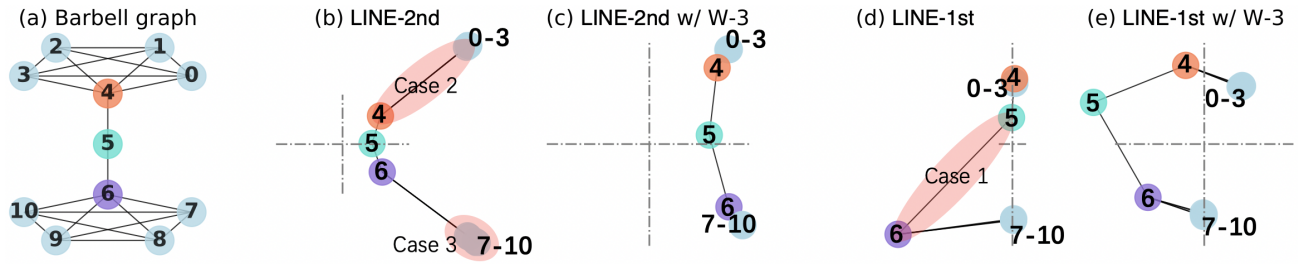



Fig. 7: (a) Barbell graph example. (b)-(e) its embedding by LINE-1st/2nd and LINE-1st/2nd with Wasserstein-3 distance ($\beta = 0.03$). The cross in each sub-figure stands for $x = 0$ ($y \in [-1, 1]$) and $y = 0$ ($x \in [-1, 1]$). The added functions show advantage in preserving structure and nodes' 1st/2nd-order proximity.

TABLE 4: Comparison of distance functions. 'Stability' includes the stability of training and stability of a faster convergence speed.

| Dist. Func. | Stability | Neighbor Preference |
|---|---|---|
| Wasserstein-1 | × | nodes with (possibly) *higher* proximity |
| Wasserstein-2 | ✓ | treat neighbors equally |
| Wasserstein-3* | ✓ | nodes with (possibly) *lower* proximity |
| Laplacian | × | nodes with (possibly) *higher* proximity |
| Gaussian | ✓ | nodes with (possibly) *higher* proximity |

*: experimentally the best on different tasks.

Wassertein-1 distance and Laplacian kernel do not always satisfy the condition.

From Fig. 5 we can see that some of the distance functions (Laplacian/Gaussian kernels, Wasserstein-1 distance) lay more weights on the node pairs $(i, j)$ whose Euclidean distance $\|\mathbf{x}_i - \mathbf{h}_j\|$ is smaller, which means that the functions prefer node pairs with a possibly higher proximity. While for others, we can see that Wasserstein-2 does not have such a preference and Wasserstein-3 prefers node pairs with a possibly lower proximity. We summarize these analyzed features as the pros and cons of the functions in Table 4.

### 4.3 Optimization

The objective Eq. 29 is optimized by asynchronous stochastic gradient decent (ASGD) [30]. We denote node embedding $\mathbf{x}_i$ at time $t$ as $\mathbf{x}_i^{(t)}$ and hidden embedding $\mathbf{h}_j$ as $\mathbf{h}_j^{(t)}$. Each thread updates NCE-based embedding within a common framework as detailed in Alg. 1. Because we have $N \gg (K + 1)$, which means the node number is much greater than batch size of each thread, there are few conflicts between threads. Hence vanilla ASGD is enough in our case and we leave the research on exploring advanced ASGD for network embedding as future work.

## 5 Visualization and Discussion

We do visualization over both simulated graphs and real-world networks, to illustrate the improvement of regularized NCE according to the proposed theorems.

### 5.1 Visualization on Barbell Graph

In this sucsection, we visualize the embedding on Barbell graph in line with [13]. Barbell graph consists of two complete graphs of the same node number connected by a path. Fig. 7(a) gives an example of barbell graph. The barbell graph has some special properties: 1) It has a symmetric structure; 2) It has both a dense part (the complete graph) and a sparse part (the path). 3) Second-order proximity in a barbell graph is varying. According to the summarized three properties, good embedding of a barbell graph should meet the following requirements: 1) the embedding should also has a symmetric distribution; 2) from the view of first-order preservation, the dense part in a graph should be dense in embedding space, and so do the sparse part. 3) to show second-order proximity, nodes with a higher second-order proximity should be closer in the embedding space compared to those share a low second-order proximity. We visualize the example barbell graph's (Fig. 7(a)) 2D embedding in Fig. 7(b)-(e). Specifically, we discuss about the embedding results in Fig. 7 as follows.

#### 5.1.1 Performance of LINE-2nd

Both LINE-2nd and its distance enhanced version can preserve symmetric structure in embedding space. According to Fig. 7(b), LINE-2nd can embed node # 0-3 (the dense part) close and node # 4-6 (the sparse part) relatively distantly so it does well in first-order proximity preserving, so for elastic LINE-2nd in Fig. 7(c). In second-order proximity preserving, we can see significant improvement shown in Fig. 7(c). Node # 0 and 1 share the highest second-order proximity, node # 1 and 4 the second, and node # 4 and 5 do not share second-order proximity. Elastic LINE-2nd achieves the results with node # 0-3 the closest, node # 4 and node # 0-3 the second, and node # 4, 5 the furthest.

#### 5.1.2 Performance of LINE-1st

Fig. 7(d) and Fig. 7(e) show that LINE-1st enhanced by Wasserstein-2 distance can preserve the symmetric structure effectively compared with LINE-1st without a distance function. Moreover, LINE-1st is designed for first-order proximity modeling according to [11], but Fig. 7(d) gives surprising results opposite to its design goal. In Fig. 7(d), node # 6 and 7 should be close just like node # 4 and node 0-3, but as the figure shows, node # 6 is very far from node # 5 as well as node # 7-10. With a distance function, first-order proximity can be naturally preserved as shown in Fig. 7(d). Interestingly, second-order proximity is also well preserved by making node # 0-3 closest, node # 4 and 0-3 the second, and node # 4, 5 the furthest.

TABLE 5: Network statistics for the tasks of network reconstruction and node classification.

| Dataset | YT-C | BC | PPI | GR-QC | DBLP |
|---|---|---|---|---|---|
| $\|\mathcal{V}\|$ | 22,579 | 10,312 | 3,890 | 5,242 | 12,591 |
| $\|\mathcal{E}\|$ | 95,506 | 333,983 | 76,584 | 14,496 | 49,627 |
| #Labels | 47 | 39 | 50 | - | - |
| Avg. Deg. | 8.46 | 64.78 | 39.37 | 5.53 | 7.88 |

### 5.2 Visualization on Real-world Networks

Apart from Barbell graph, we also do visualization for 2-D embedding over a real-world network, YouTube-Cut, whose detailed information is given in Table 5 and Sec. 6.1.1.

In Fig. 8, we plot two nodes' hidden embeddings and the node embeddings of their neighbors with different markers. For LINE-2nd, the distance term: i) reduce the distance between node embedding and hidden embedding for connected nodes (first-order proximity preservation); ii) reduce the distance between node embeddings who have a common neighbor (second-order proximity preservation); iii) break the linearity and devide the two parts from the same area. The observed improvement is all in accord with our theoretical analysis in Sec. 3.2, Sec. 3.3, and Sec. 4.2. For LINE-1st, although the improvement brought by the distance function is not that significant, we will show the improvement by the experimental results in Sec. 6.

In Fig. 9, we plot the node embeddings for three classes of nodes. For LINE-2nd, with distance term the three classes are divided more separately and more non-linear structures are preserved. For LINE-1st, the role of distance functions is not as conspicuous as that in LINE-2nd.

Comparing the normalized embeddings and the unormalized embeddings in Fig. 8 and Fig. 9, we can also see that normalization over embeddings may help to reorganize the whole embedding layout while with little substantial improvement. That is to say, the contribution from distance functions and that from normalization are orthogonal and can be adopted simultaneously.

## 6 Experiments

We evaluate state-of-the-art embedding methods and especially including those NCE-based ones enhanced with our proposed distance modeling module, on the tasks of node classification and network reconstruction. All the experiments run on a single desktop with 128G memory, 4 physical CPU each with 12 cores (Intel(R) Xeon(R) CPU E5-2678 v3 @ 2.50GHz). The default number of threads is 8.

### 6.1 Protocols

#### 6.1.1 Datasets

Experiments are conducted on several real-world benchmarks, whose statistics are given in Table 5.

**YouTube-Cut** [31] (YT-C): a social network where users can add others to the friend list and join interest groups as treated as user labels. It is a large scale network with millions of nodes being very sparse. In line with the protocol in [3], to make it a smaller network that can be handled by the relatively high complexity model, we remove the unlabeled nodes from raw YouTube.

**BlogCatalog** [32] (BC): a social network whose blog is organized by specific categories, which are the labels of a blogger, and bloggers have social connections with each other.

**PPI** [33]: a subgraph of the PPI network for Homo Sapiens. Node labels represent biological states.

**Arxiv GR-QC**[2] [34]: a collaboration network from arXiv and covers scientific collaborations between authors with papers submitted to General Relativity and Quantum Cosmology category.

**DBLP**[3] [35]: a citation network of DBLP. Each node denotes a publication, and each edge represents a citation.

#### 6.1.2 Compared methods

For all methods, we set embedding space's dimension 128. For NCE-based methods, we set 5 negative samples for each sampled positive node pair. Distance weight $\beta$ used in node classification is given in Table 6. For network reconstruction, we give general information that $\beta$ falls in $[0.01, 0.1]$. We set $\gamma = 1$ which is only used in Laplacian/Gaussian kernel functions.

**DeepWalk**[4] [10] combines random walks and the skip-gram language model for embedding. In the implementation, it adopts hierarchical softmax to model node embedding and hidden embedding. We set window size 10, walk length 80 and number of walks for each node 40.

**LINE**[5] [11] models first-order (by NCE-x model, Eq. 3) and second-order proximity (by NCE-h model, Eq. 2) on the adjacency matrix, trains them separately with edge sampling, and concatenates the representations after normalization on node embedding. It can be regarded as 1-hop random walk with both NCE-h and NCE-x. Its full version and two variants, **LINE-1st** and **LINE-2nd**, are studied. The number of samples is set as $3 \times 10^8$.

**node2vec**[6] [12] uses biased random walk to explore both breadth-first and depth-first structure, and train node embedding and hidden embedding by negative sampling (i.e. NCE-h model, Eq. 2). We set $p = 1$ and $q = 1$ in the experiments, and the other parameters for random walks are the same with DeepWalk. Under our settings, node2vec can be regarded as multi-hop uniform random walks with NCE-h model.

**AROPE**[7] [5] models arbitrary-order proximity based on SVD framework. The experiments in [5] show that the third-order proximity yields the best results in most tasks, so we utilize the third-order embedding as the node embedding. We set the weight of different order $[1, 0.1, 0.01]$.

**VERSE**[8] [14] uses node similarity explicitly by Personalized PageRank (PPR), Adajcency Similarity, and SimRank, and learns node embedding by NCE-x. We use the default version with PPR similarity, damping factor $\alpha = 0.85$. We run $10^5$ epochs for each node.

**RNS**[9] [24] proposes a normalization penalty over node embeddings. The number of samples is $3 \times 10^8$.

2. http://snap.stanford.edu/data/ca-GrQc.html
3. http://konect.uni-koblenz.de/networks/dblp-cite
4. https://github.com/xgfs/deepwalk-c
5. https://github.com/tangjianpku/LINE
6. https://github.com/xgfs/node2vec-c
7. https://github.com/ZW-ZHANG/AROPE.git
8. https://github.com/xgfs/verse
9. https://github.com/ShawXh/RNCE/blob/main/rns.cpp

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3148284, IEEE Transactions on Knowledge and Data Engineering



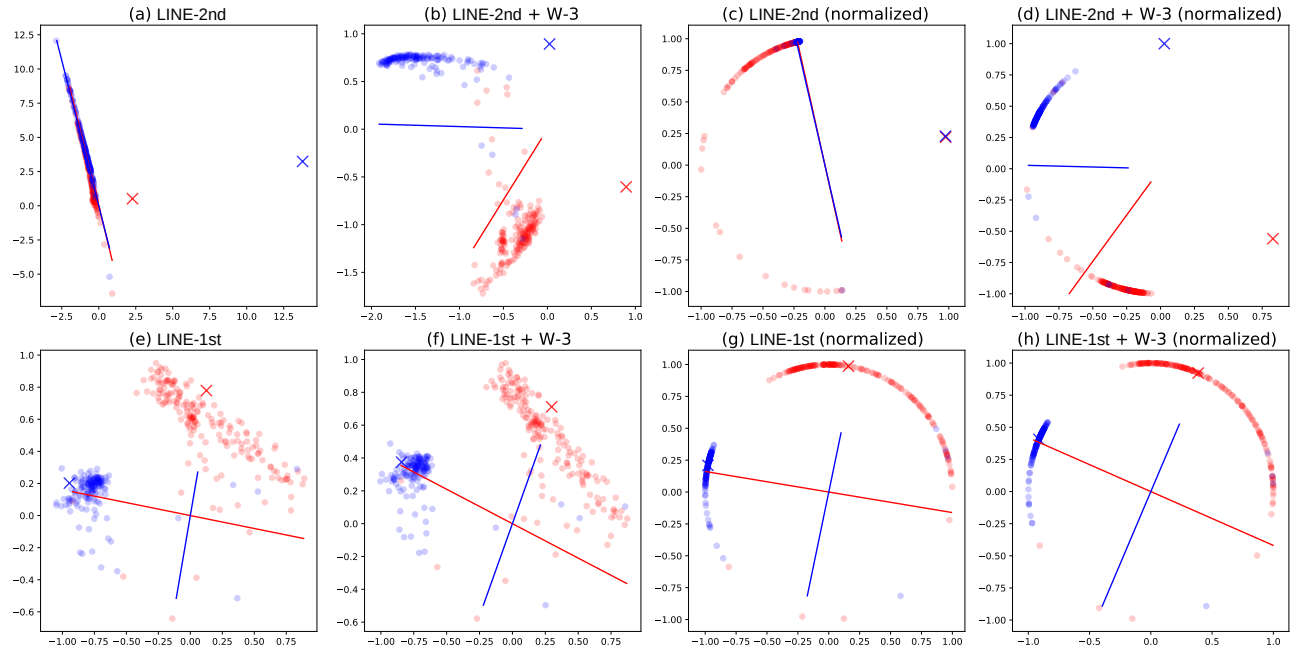



Fig. 8: Results of 2-D embedding by four categories of objectives on YouTube-Cut: (a) LINE-2nd, (b) LINE-2nd w/ Wasserstein-3 distance, (e) LINE-1st, and (f) LINE-1st w/ Wasserstein-3 distance. Fig. (c)(d)(g)(h) are normalized embedding results. The marker '$\times$' denotes hidden embedding of the two nodes while the marker 'o' denotes the embedding of their neighbors. The straight lines denote the hyperplane of $\mathbf{x}$ where $\mathbf{x}^\top \mathbf{y}_j = 0$ with $\mathbf{y}_j$ being the hidden embedding.

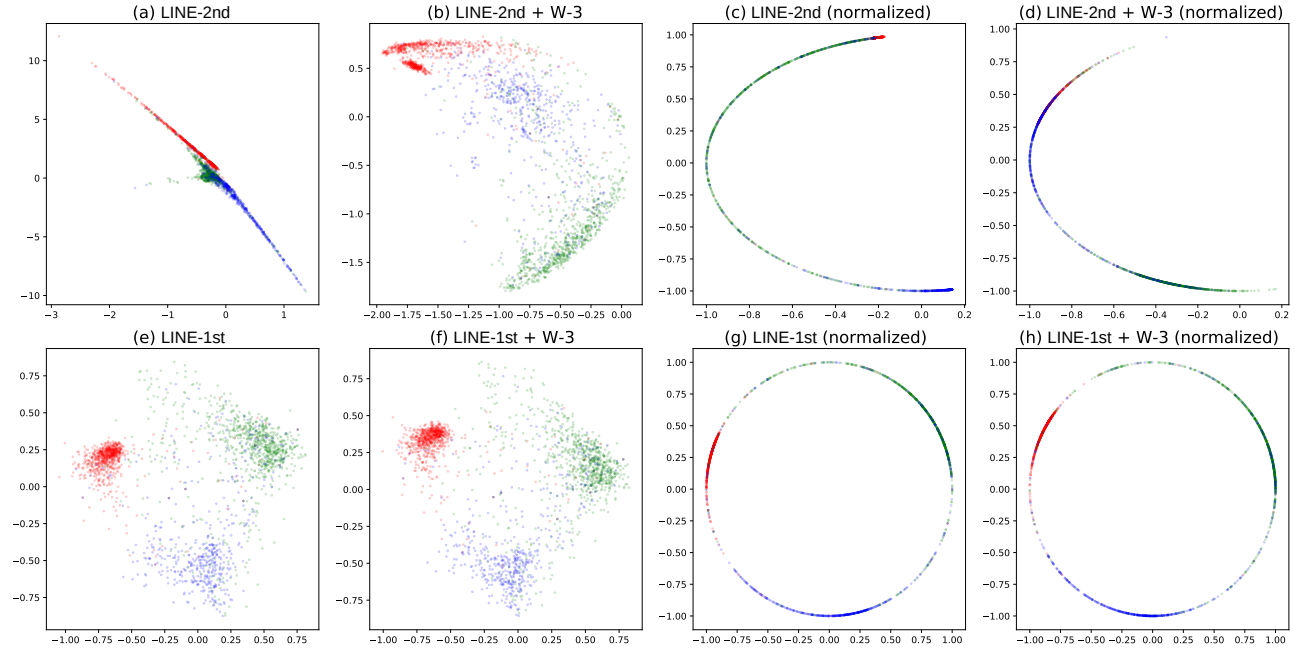



Fig. 9: Results of 2-D embedding on YouTube-Cut. Different colors denote different classes. Note that the embedding results of LINE-2nd are improved by Wasserstein-3 distance, as the nodes of different classes are separated more clearly.

**ProNE**[10] [6] initializes embedding by sparse matrix factorization, and then enhances the embedding via spectral propagation. We run Python version in the repository with the default setting, i.e. the term number of the Chebyshev expansion $k = 10$, $\theta = 0.5$, and $\mu = 0.2$. We use the enhanced embedding [6] as the node embedding result.

10. https://github.com/THUDM/ProNE.git

**STRAP** [26] makes use of the backward push algorithm to efficiently compute the sparse Personalized PageRank (PPR) as its transpose proximity. The experimental results are quoted from [26].

### 6.2 Experiments on Node Classification

As a standard task to evaluate network embedding, multi-label classification assumes each node has one or more

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3148284, IEEE Transactions on Knowledge and Data Engineering



TABLE 6: Node classification on BlogCatalog with different distances. The result of STRAP is quoted from [26]. Underlined denotes those outperforming all the nine baselines in the first nine rows ('no-NCE' stands for the methods not based on NCE, 'none' stands for the NCE-based methods without distance functions). Bold dempets those outperform other methods with the same setting of distance.

| dist. | Methods | Macro-F1 (%) | | | | | Micro-F1 (%) | | | | | $\beta$ | Time (s) | Avg Gain (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 10% | 30% | 50% | 70% | 90% | 10% | 30% | 50% | 70% | 90% | | | |
| no-NCE | DeepWalk | **20.94** | 24.72 | 26.35 | 26.83 | 27.25 | 35.37 | 38.74 | 40.29 | 40.69 | 40.74 | - | 111.3 | - |
| | AROPE | 13.14 | 13.90 | 13.74 | 14.09 | 14.18 | 26.17 | 29.43 | 30.08 | 30.99 | 31.48 | - | 28.8 | - |
| | ProNE | 19.7 | 21.82 | 22.55 | 22.98 | 22.87 | **37.24** | 39.33 | 39.96 | 40.16 | 40.33 | - | 14.4 | - |
| | STRAP* | - | - | - | - | - | 36.42 | **40.29** | **41.59** | **42.68** | **42.55** | - | - | - |
| None | LINE-2nd | 15.03 | 20.09 | 21.89 | 22.84 | 22.80 | 26.54 | 32.79 | 35.87 | 36.42 | 36.80 | - | 69.2 | - |
| | LINE-1st | 17.83 | 19.62 | 19.77 | 19.94 | 19.63 | 34.12 | 36.87 | 37.19 | 37.46 | 37.40 | - | 69.7 | - |
| | LINE | 17.95 | 21.70 | 23.13 | 24.28 | 24.19 | 35.74 | 38.72 | 39.55 | 40.14 | 40.31 | - | 136.1 | - |
| | node2vec | 20.41 | 25.12 | 27.31 | 28.17 | **28.78** | 35.23 | 38.91 | 40.42 | 41.18 | 41.96 | - | 70.2 | - |
| | VERSE | 20.35 | **25.22** | **27.33** | **28.37** | 28.32 | 33.88 | 38.33 | 40.00 | 40.81 | 41.33 | - | 231.8 | - |
| RNS | LINE-2nd | 15.33 | 20.39 | 22.20 | 23.05 | 23.42 | 27.10 | 33.55 | 35.94 | 36.98 | 37.71 | - | 116.06 | +0.45 |
| | LINE-1st | 18.61 | 21.21 | 22.16 | 22.42 | 21.74 | 34.05 | 37.73 | 38.85 | 39.17 | 38.94 | - | 114.92 | +1.51 |
| | LINE | 18.47 | 22.42 | 23.87 | 24.83 | 24.72 | 36.01 | 39.24 | 40.40 | 40.77 | 40.76 | - | 185.48 | +0.58 |
| Wasserstein-1 | LINE-2nd | 13.31 | 17.16 | 19.08 | 20.28 | 21.48 | 32.74 | 36.29 | 37.39 | 37.96 | 38.64 | 0.05 | 84.0 | +0.32 |
| | LINE-1st | 19.77 | 22.18 | 23.04 | 23.39 | 23.26 | 36.06 | 38.97 | 39.71 | 40.12 | 40.21 | 0.05 | 85.4 | +2.67 |
| | LINE | 18.91 | 23.02 | 24.48 | 25.51 | 25.42 | **36.54** | **39.48** | **40.38** | 40.99 | 40.73 | 0.05 | 165.5 | +0.98 |
| | node2vec | **20.72** | **25.49** | **27.38** | **28.10** | 28.06 | 35.54 | 39.11 | 40.35 | **41.07** | **41.33** | 0.01 | 67.7 | +0.58 |
| | VERSE | 20.16 | 24.69 | 26.91 | 27.74 | **28.72** | 33.81 | 38.08 | 39.92 | 40.77 | **41.33** | 0.01 | 271.1 | -0.18 |
| Elastic (W-2) | LINE-2nd | 18.26 | 23.28 | 25.18 | 26.38 | 26.26 | 37.54 | 40.93 | 42.05 | 42.54 | 42.41 | 0.1 | 83.5 | +5.37 |
| | LINE-1st | 21.55 | 23.58 | 24.08 | 24.41 | 24.60 | 38.75 | 40.66 | 41.19 | 41.32 | 41.42 | 0.1 | 83.4 | +4.17 |
| | LINE | 21.36 | 25.04 | 26.09 | 26.84 | 26.90 | **38.85** | **41.48** | **42.16** | **42.60** | **42.52** | 0.1 | 160.0 | +2.81 |
| | node2vec | **22.00** | 25.85 | 27.55 | 28.32 | 28.81 | 38.06 | 39.71 | 41.37 | 41.75 | 41.96 | 0.015 | 70.8 | +1.09 |
| | VERSE | 21.85 | **25.90** | **27.60** | **28.87** | **29.26** | 34.86 | 39.09 | 40.78 | 41.69 | 42.20 | 0.02 | 263.0 | **+0.82** |
| Wasserstein-3 | LINE-2nd | 18.37 | 23.51 | 25.33 | 26.31 | 26.18 | 37.65 | 41.07 | 42.13 | 42.55 | 42.35 | 0.01 | 83.4 | **+5.43** |
| | LINE-1st | **23.25** | 25.25 | 25.90 | 26.16 | 25.94 | **39.59** | **41.65** | 42.10 | 42.31 | 42.35 | 0.02 | 86.2 | **+5.47** |
| | LINE | 21.73 | 25.59 | 27.07 | 28.16 | 28.36 | 39.05 | 41.64 | **42.68** | **43.10** | **43.25** | 0.01/0.02 | 144.9 | **+3.49** |
| | node2vec | 22.00 | 25.85 | 27.36 | **28.49** | 28.26 | 38.61 | 41.03 | 41.82 | 42.37 | 42.31 | 0.002 | 66.7 | **+1.36** |
| | VERSE | 21.50 | **25.91** | **27.52** | 28.42 | **28.71** | 35.97 | 39.38 | 40.75 | 41.30 | 41.56 | 0.01 | 292.9 | +0.71 |
| Laplacian | LINE-2nd | 14.05 | 18.26 | 20.27 | 21.75 | 22.37 | 33.66 | 37.31 | 38.39 | 39.12 | 39.60 | 0.8 | 78.8 | +1.37 |
| | LINE-1st | 18.55 | 20.71 | 21.26 | 21.63 | 21.56 | 34.99 | 37.64 | 38.34 | 38.73 | 38.84 | 0.1 | 81.8 | +1.24 |
| | LINE | 18.41 | 22.44 | 23.93 | 25.01 | 25.14 | **36.03** | **39.07** | 39.98 | 40.48 | 40.64 | 0.8/0.1 | 158.5 | +0.54 |
| | node2vec | **20.58** | **25.35** | **27.15** | 27.90 | 28.47 | 35.51 | **39.07** | **40.53** | **41.07** | **41.55** | 0.005 | 68.2 | +0.27 |
| | VERSE | 20.16 | 25.29 | 27.10 | **28.15** | **28.69** | 33.62 | 38.24 | 40.02 | 40.89 | 41.50 | 0.01 | 266.5 | -0.03 |
| Gaussian | LINE-2nd | 18.04 | 23.16 | 25.01 | 25.93 | 26.19 | 37.39 | **40.86** | **41.93** | **42.22** | **42.36** | 0.7 | 77.8 | +5.20 |
| | LINE-1st | 19.18 | 21.69 | 22.59 | 22.90 | 22.73 | 35.57 | 38.54 | 39.41 | 39.74 | 39.76 | 0.1 | 78.8 | +2.23 |
| | LINE | 20.16 | 24.35 | 25.83 | 26.74 | 27.03 | **37.63** | 40.58 | 41.47 | 41.83 | 41.86 | 0.7/0.1 | 150.0 | +2.18 |
| | node2vec | **20.56** | 25.29 | 27.22 | 27.95 | 27.58 | 35.38 | 38.95 | 40.33 | 41.07 | 41.37 | 0.02 | 68.5 | +0.12 |
| | VERSE | 20.44 | **25.69** | **27.23** | **28.14** | **28.13** | 34.30 | 38.64 | 40.08 | 40.93 | 41.18 | 0.05 | 266.0 | +0.08 |

TABLE 7: Node classification Macro-F1 (%) on YouTube-Cut.

| dist. | Methods | 1% | 3% | 5% | 7% | 9% | $\beta$ |
|---|---|---|---|---|---|---|---|
| no-NCE | DeepWalk | 30.25 | 33.9 | 35.28 | 36.41 | 37.02 | - |
| | ProNE | **31.02** | **35.46** | **38.05** | **39.08** | **39.60** | - |
| | AROPE | 29.18 | 32.90 | 33.80 | 34.77 | 34.95 | - |
| None | LINE | 29.58 | 34.18 | 36.24 | 37.93 | 38.90 | - |
| | node2vec | 23.96 | 32.73 | 35.32 | 36.96 | 38.03 | - |
| | VERSE | 29.81 | 34.45 | 36.11 | 37.21 | 38.10 | - |
| | RNS-LINE | 28.54 | 34.54 | 36.98 | 38.91 | 39.93 | - |
| W-3 | LINE | 32.23 | 37.15 | **39.26** | **40.71** | **41.41** | 0.05 |
| | node2vec | **32.34** | **37.33** | 39.02 | 40.19 | 40.96 | .005 |
| | VERSE | 31.35 | 36.22 | 37.77 | 38.80 | 39.42 | 0.05 |

TABLE 8: Node classification Macro-F1 (%) on PPI.

| dist. | Methods | 10% | 30% | 50% | 70% | 90% | $\beta$ |
|---|---|---|---|---|---|---|---|
| no-NCE | DeepWalk | 13.07 | 16.60 | 18.01 | 19.30 | 20.10 | - |
| | AROPE | 10.84 | 13.62 | 14.35 | 15.53 | 15.39 | - |
| | ProNE | **14.12** | **17.89** | **18.77** | **19.30** | **19.80** | - |
| None | LINE | 12.84 | 16.15 | 17.71 | 18.74 | 18.83 | - |
| | node2vec | 12.18 | 16.52 | 17.99 | 19.05 | 19.32 | - |
| | VERSE | 12.45 | 16.18 | 17.87 | 18.88 | **19.80** | - |
| | RNS-LINE | 12.50 | 16.66 | 18.46 | 19.72 | 20.10 | - |
| W-3 | LINE | **14.25** | **18.67** | **20.02** | **21.30** | **20.98** | 0.05 |
| | node2vec | 14.15 | 18.30 | 19.76 | 20.78 | 20.65 | .005 |
| | VERSE | 13.51 | 17.44 | 18.90 | 19.76 | 19.78 | 0.05 |

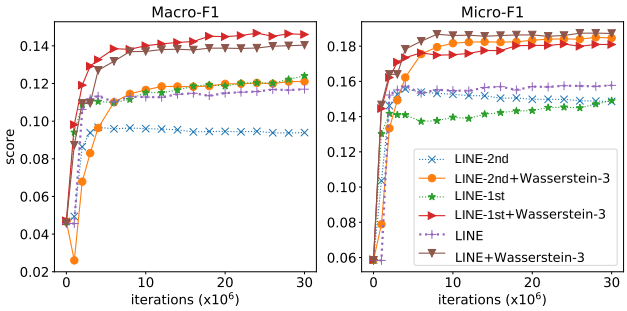
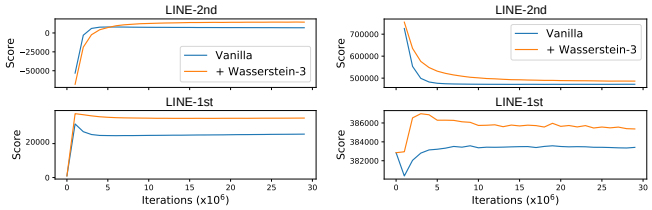



Fig. 10: Performance of LINE and the version enhanced by Wasserstein-3 learning in the first 30 million iterations on PPI with 10% portion of training data.

labels for prediction. After node embedding is learned, LIBLINEAR [36] is adopted to train the one-vs-rest Logistic regression classifiers. Macro-F1 and Micro-F1 serve as the metric for evaluation. We randomly sample a portion of the labeled nodes, whose representations are set as training data and the rest nodes' representation for testing. Specifically, we randomly sample 10% to 90% portion of nodes on BlogCatalog and PPI, and 1% to 9% on YouTube-Cut. For

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3148284, IEEE Transactions on Knowledge and Data Engineering






(a) Positive node pairs (b) Negative node pairs

Fig. 11: The cosine similarity score convergence curve (the higher the better). For positive node pairs, $score = \sum_{i,j=1}^{N} w_{ij} \cos\langle \mathbf{x}_i \mathbf{h}_j \rangle$. For negative node pairs, the score is estimated by $score = \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{E}_{j' \sim \mathbb{P}_n} (1 - \cos\langle \mathbf{x}_i \mathbf{h}_{j'} \rangle)$. From the curve, we can find that with distance function, both LINE-1st/2nd for both positive/negative node pairs converged at a higher score. It indicates that distance functions help the vanilla NCE loss do a better job on distinguishing positive samples from negative ones.

each method, we repeat the procedure for 10 trials and their average are reported.

On BlogCatalog, Table 6 reports the full results for all the baselines as well as NCE-based models with all the mentioned distance functions. On YouTube-Cut (Table 7) and PPI (Table 8) we report only those of NCE-based methods with Wasserstein-3, which yield best performance among all the distance functions. The other distance functions behave similarly on the two datasets compared to on BlogCatalog. In Table 6, all of NCE-based methods are improved by the adoption of distance functions. Among them, Wasserstein-3 is the most brilliant. With Wasserstein-3, even LINE-1st which only utilizes direct neighbors (see Table 1) outperforms about half of the baselines. The overwhelming performance of Wasserstein-3 version is also consistent with our speculation in Sec. 4.2.6.

We also evaluate LINE with Wasserstein-3 on PPI. The learning curve of the first 30 million iterations is shown in Fig. 10. The results indicate that: 1) In the earlier stage, distance function accelerates their convergence; 2) In the middle, distance function helps break the bottleneck of convergence by improving the behavior of gradient descent; 3) In the later stage, distance function keeps fine-tuning to finally reach better performance.

### 6.3 Experiments on Network Reconstruction

Network reconstruction can be used to measure how well a embedding model preserves nodes' origin local structure. The performance relies heavily on the choice of the reconstruction score functions [3], [5], [14]. In line with [3], we reconstruct the network according to the Euclidean distance between node embedding and leave hidden embedding unused, which is exactly required by first-order proximity [37]. We calculate Euclidean distance between all possible node pairs, namely $N(N-1)$ directed node pairs. Then we sort the distances and select a number of node pairs equal to the number of edges in the given network as reconstructions. Finally, precision is calculated with the edges in the raw network as ground-truth.

NCE-based methods are improved significantly with the adoption of the distance functions, suggesting their effectiveness in structure preservation. Similar to the case in node classification, Wasserstein-3 achieves the largest gain compared with other distance functions. Among the NCE-based methods, VERSE basically performs best on this task perhaps due to advanced PPR neighbor modeling. We also make some interesting observations: As a superior version of LINE-2nd by biased random-walk based neighbor extension, node2vec consistently outperforms LINE-2nd. VERSE also mostly outperforms its basis, LINE-1st. It indicates that not only a proper distance function is useful, but also the expansion of neighborhood plays an irreplaceable roles. In fact, the latter has been well studied and showcased in literature (see Table 1).

### 6.4 Further Discussion

We discuss further on the proposed regularized NCE.

#### 6.4.1 *Convergence Analysis*

We plot the convergence curve of cosine similarity between postive pairs and negative pairs in Fig. 11. We see that under the same parameter settings, regularized NCE will converge at a higher point for both positive samples and negative samples, which means that compared with vanilla NCE, regularized NCE does better at distinguishing postive samples from negative samples.

#### 6.4.2 *Time Complexity*

For NCE-based embedding methods, distance functions only incur a linear time cost without additional space overhead, which means both total time complexity and space complexity are not affected by distance functions, as verified by the running time of compared methods in Table 6.

#### 6.4.3 *Parameter Sensitivity*

We also evaluate the effect of distance weight $\beta$ with embedding dimension $d = 128$ on the PPI dataset. Fig. 12 depicts how node classification and network reconstruction behave with varying $\beta$. The results show that both LINE-1st/2nd yield best results with Wasserstein-3 on both tasks, while Wasserstein-2 can achieve comparable results with different $\beta$. The best $\beta$ is about $0.03$ for Wasserstein-3 and around $0.1$ for Wasserstein-2, which actually is able to yield good results for most networks. The results also indicate that a too large $\beta$ may lead to the result that high-order neighbors are embedded too close in the embedding space thus affecting the performance, which is in accordance with Theorem 5.

## 7 Conclusion

We have provided a Noise Contrastive Estimation perspective on skip-gram based network embedding methods, and identify two basic components used in the objectives.

Starting from these two components, we have explored various forms of the score function which is in contrast to existing methods focusing on the modeling of neighborhoods. In particular, the elastic potential energy form bears a clear physical meaning to mitigate the issues in gradient descent based updating procedure by our in-depth analysis.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3148284, IEEE Transactions on Knowledge and Data Engineering



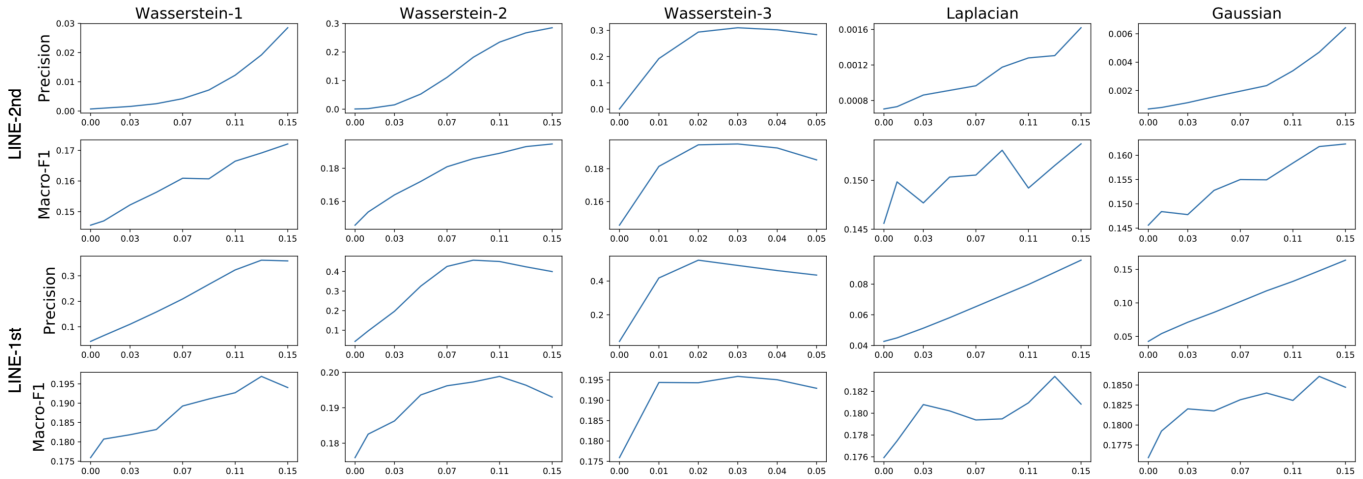



Fig. 12: Performance of LINE-1st/2nd in addition with different distance functions with varying $\beta$, denoted by $x$-axis.

TABLE 9: Network reconstruction precision (%) with average gain (%) on different datasets.

| dist. | Methods | GR-QC | BC | DBLP | PPI | Gain |
|---|---|---|---|---|---|---|
| None | LINE-2nd | 39.73 | <0.01 | 0.02 | 0.07 | - |
| | LINE-1st | **60.61** | 2.50 | 15.70 | 5.24 | - |
| | DeepWalk | 59.53 | 8.46 | 25.95 | 26.96 | - |
| | LINE | 53.73 | 6.22 | 3.10 | 14.46 | - |
| | node2vec | **60.61** | 8.59 | **23.15** | **42.10** | - |
| | AROPE | 14.05 | <0.01 | 1.20 | 0.09 | - |
| | VERSE | 59.95 | **21.32** | 22.94 | 39.34 | - |
| | ProNE | 49.93 | 3.50 | 0.93 | 24.01 | - |
| RNS | LINE-2nd | 48.88 | 0.72 | 8.96 | 0.96 | +4.93 |
| | LINE-1st | 60.46 | 1.95 | 15.30 | 12.4 | +1.52 |
| | LINE | 53.11 | 4.19 | 3.30 | 16.94 | +0.01 |
| Wasserstein-1 | LINE-2nd | 51.74 | <0.01 | 1.90 | 19.00 | +8.21 |
| | LINE-1st | 60.03 | 7.06 | 17.75 | 36.12 | +9.23 |
| | LINE | 56.60 | 5.11 | 2.68 | 15.92 | +0.70 |
| | node2vec | 63.49 | 7.97 | **25.05** | **44.44** | +1.63 |
| | VERSE | **66.16** | **22.74** | 24.34 | **44.44** | +3.53 |
| Elastic (W-2) | LINE-2nd | 60.11 | 3.40 | 2.80 | 22.35 | +12.21 |
| | LINE-1st | 61.60 | 13.42 | 20.59 | 46.13 | +14.42 |
| | LINE | 59.28 | 5.18 | 6.33 | 23.84 | +4.28 |
| | node2vec | **64.44** | 6.40 | 24.47 | **49.07** | +2.48 |
| | VERSE | 62.40 | **29.56** | **25.10** | 45.89 | +4.85 |
| Wasserstein-3 | LINE-2nd | 59.77 | 5.71 | 5.36 | 29.44 | **+15.12** |
| | LINE-1st | 63.08 | 25.62 | 22.67 | **52.40** | **+19.93** |
| | LINE | 59.42 | 7.40 | 9.68 | 28.43 | **+6.86** |
| | node2vec | **63.49** | 10.00 | 24.56 | 50.23 | **+3.46** |
| | VERSE | 61.88 | **26.17** | **25.66** | 46.70 | +4.22 |
| Laplacian | LINE-2nd | 56.07 | <0.01 | 0.05 | 0.60 | +4.23 |
| | LINE-1st | 59.45 | 2.60 | 15.63 | 11.35 | +1.25 |
| | LINE | 54.06 | 5.18 | 2.25 | 15.67 | -0.09 |
| | node2vec | 60.79 | 8.36 | 22.50 | **44.74** | +0.49 |
| | VERSE | **66.87** | **23.11** | **30.10** | 41.98 | +4.63 |
| Gaussian | LINE-2nd | 54.53 | <0.01 | 2.73 | 13.53 | +7.74 |
| | LINE-1st | 60.64 | 3.32 | 14.82 | 20.15 | +3.72 |
| | LINE | 57.41 | 4.73 | 3.34 | 16.30 | +1.07 |
| | node2vec | 63.29 | 8.65 | 25.16 | **45.25** | +1.98 |
| | VERSE | **72.91** | **26.07** | **29.78** | 44.30 | **+7.38** |

Extensive experimental results demonstrate the state-of-the-art performance achieved by our proposed techniques.

For future work, we aim to explore other more direct and efficient asynchronous gradient updating mechanisms.

## ACKNOWLEDGMENT

This research was supported by National Key Research and Development Program of China (2020AAA0107600), and Open Research Projects of Zhejiang Lab (NO. 2021KB0AB04). The authors are thankful to the reviewers' valuable comments to improve the paper.

## REFERENCES

[1] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[2] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[3] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1225–1234.

[4] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1105–1114.

[5] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, and W. Zhu, "Arbitrary-order proximity preserved network embedding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2778–2786.

[6] J. Zhang, Y. Dong, Y. Wang, J. Tang, and M. Ding, "Prone: Fast and scalable network representation learning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 4278–4284. [Online]. Available: https://doi.org/10.24963/ijcai.2019/594

[7] J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, and J. Tang, "Netsmf: Large-scale network embedding as sparse matrix factorization," in *The World Wide Web Conference*. ACM, 2019, pp. 1509–1520.

[8] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.

[9] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[11] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[12] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[13] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 385–394.

[14] A. Tsitsulin, D. Mottin, P. Karras, and E. Müller, "Verse: Versatile graph embeddings from similarity measures," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 539–548.

[15] X. Du, J. Yan, R. Zhang, and H. Zha, "Cross-network skip-gram embedding for joint network alignment and link prediction," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[16] H. Xiong and J. Yan, "Btwalk: Branching tree random walk for multi-order structured network embedding," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Computer Science*, 2013.

[19] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 297–304.

[20] T. Jian, Q. Meng, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *Acm Sigkdd International Conference on Knowledge Discovery & Data Mining*, 2015.

[21] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM, 2015, pp. 891–900.

[22] D. Luo, C. Ding, F. Nie, and H. Huang, "Cauchy graph embedding." 01 2011, pp. 553–560.

[23] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 2012, pp. 419–426.

[24] M. Armandpour, P. Ding, J. Huang, and X. Hu, "Robust negative sampling for network embedding," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3191–3198, 07 2019.

[25] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "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, 2018, pp. 459–467.

[26] Y. Yin and Z. Wei, "Scalable graph embeddings via sparse transpose proximities," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1429–1437. [Online]. Available: https://doi.org/10.1145/3292500.3330860

[27] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," *Advances in neural information processing systems*, vol. 27, pp. 2177–2185, 2014.

[28] P. Erdős and A. Rényi, "On random graphs i," *Publicationes Mathematicae*, vol. 4, pp. 3286–3291, 1959.

[29] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017, pp. 214–223.

[30] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in neural information processing systems*, 2011, pp. 693–701.

[31] L. Tang and H. Liu, "Scalable learning of collective behavior based on sparse social dimensions," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 1107–1116.

[32] ——, "Relational learning via latent social dimensions," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 817–826.

[33] B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bähler, V. Wood *et al.*, "The biogrid interaction database: 2008 update," *Nucleic acids research*, vol. 36, no. suppl_1, pp. D637–D640, 2007.

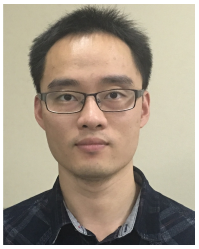
[34] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 2–es, 2007.

[35] M. Ley, "The DBLP computer science bibliography: Evolution, research issues, perspectives," in *Proc. Int. Symposium on String Processing and Information Retrieval*, 2002, pp. 1–10.

[36] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[37] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.


**Hao Xiong** is currently a PhD candidate with Department of Computer Science and Engineering, Shanghai Jiao Tong University, under the supervision of Junchi Yan and Xuemin Lin by a joint PhD program between SJTU and University of New South Wales. Before that, he received the B.E. degree in Cyber Science and Engineering (with honor) from the same university in 2019. He was a research intern in Amazon AI Labs, Shanghai from 2019 to 2020, working on the open-source graph platform DGL. His research interests include machine learning, data mining, and network analysis.



**Junchi Yan** (S'10-M'11-SM'21) is currently an Associate Professor with Department of Computer Science and Engineering, Shanghai Jiao Tong University. Before that, he was a Senior Research Staff Member and Principal Scientist with IBM Research – China where he started his career in April 2011. He obtained the Ph.D. in Electrical Engineering, from Shanghai Jiao Tong University, China in 2015. His research interests are machine learning and computer vision. He serves as Area Chair for ACM-MM 2021/22, CVPR 2021, AAAI 2022, ICML 2022. He is a Senior Member of IEEE.

**Zengfeng Huang** is currently an Associate Professor in the School of Data Science, Fudan University. Before that he was a Research Fellow in CSE, UNSW and a Postdoc in MADALGO, Aarhus University. He obtained his PhD at Hong Kong University of Science and Technology in CSE and B.S. degree in Computer Science from Zhejiang University in 2008. His research interests are algorithmic aspects of data science. His single author paper has been nominated as the outstanding paper in ICML 2018.